DISSERTATION

submitted to the
Combined Faculties for the Natural Sciences and for Mathematics
of the Ruperto-Carola University of Heidelberg, Germany

for the degree of
Doctor of Natural Sciences

Put forward by
M.Sc. Philipp Johannes Kreyenberg
born in Herrenberg

Oral examination: November 6, 2019

FLOW FIELD ESTIMATION OF ACTIVE SOLUTE TRANSPORT
–
INFORMATION TRANSFER FROM SYNTHETIC DATA TO HELE-SHAW CELL EXPERIMENTS USING CONVOLUTIONAL NEURAL NETWORKS

Referees:

Prof. Dr. Kurt Roth
Prof. Dr. Werner Aeschbach

Flow Field Estimation of Active Solute Transport: Variable density groundwater flow associated with active solute transport is understood reasonably well. Nevertheless, predictions are operationally still difficult due to joint effects of nonlinear processes and uncertain boundary conditions. Gaining deeper insight into the dynamics of these groundwater systems therefore relies on the availability of accurate and dense measurements of the complete system state and parameters. Often, such measurements are hard to come by, hence our information is incomplete. Recent deep learning methods in conjunction with numerical simulation of the physical processes to create large training datasets enable the information transfer to real world problems. To demonstrate this, I chose a laboratory experiment on density-driven active solute transport observed in a Hele-Shaw cell, where high resolution measurements of the solute concentration distribution are available. With the use of deep convolutional neural networks I was able to estimate the otherwise inaccessible flow fields and to identify the influence of background flow for this experiment without explicit knowledge of the boundary conditions. The situation of missing data, as encountered here, is typical also for other hydrological systems, from soil-vegetation-atmosphere interactions to catchment dynamics and groundwater recharge. Hence, I believe that the approach has wide applicability.

Schätzung von Strömungsfeldern für Aktiven Stofftransport: Die Strömung von Grundwasser mit variabler Dichte im Zusammenhang mit aktivem Stofftransport ist hinreichend gut verstanden. Dennoch sind Vorhersagen aufgrund von kombinierten Auswirkungen nichtlinearer Prozesse und ungewisser Randbedingungen in der Praxis nach wie vor schwierig. Um einen tieferen Einblick in die Dynamik dieser Grundwassersysteme zu erhalten, müssen genaue und gut aufgelöste Messungen des gesamten Systemzustands und der Parameter verfügbar sein. Oft ist es schwierig solche Messungen zu erhalten, weshalb unsere Informationen über die Systeme unvollständig sind. Aktuelle Deep-Learning-Methoden in Verbindung mit numerischer Simulation der physikalischen Prozesse zur Generierung großer Trainingsdatensätze ermöglichen den Informationstransfer hin zu realen Problemen. Um dies zu demonstrieren, habe ich ein Laborexperiment zu dichtegetriebenem aktiven Stofftransport in einer Hele-Shaw-Zelle gewählt, in dem hochaufgelöste Messungen der Konzentrationsverteilung des gelösten Stoffes verfügbar sind. Unter Verwendung von Deep-Convolutional-Neural-Networks konnte ich, ohne die genaue Kenntnis der Randbedingungen, die sonst unzugänglichen Strömungsfelder schätzen und den Einfluss der Hintergrundströmung für dieses Experiment identifizieren. Die Situation fehlender Daten, wie sie in diesem Beispiel auftritt, ist auch für andere hydrologische Systeme typisch, von Wechselwirkungen zwischen Boden, Vegetation und Atmosphäre über die Dynamik von Wassereinzugsgebieten bis hin zur Grundwasserneubildung. Deshalb glaube ich, dass der Ansatz eine breite Anwendbarkeit hat.

Contents

1 Introduction

Based on *Kreyenberg et al.* [2019].

Gaining deeper insight into hydrological systems is challenging and relies on the availability of accurate measurements that are dense in space and time. More and more of such data become available with increasing deployment of e.g., satellite-based sensors or embedded sensor networks. However, some relevant system quantities, for instance local flow velocities, remain difficult to measure. On the other hand, simulations offer the advantage of detailed information, also of the quantities that are difficult to measure. They are often based on a good physical understanding, but the presence of nonlinear processes and multiscale heterogeneities typically impedes accurate predictions.

Advances can be made by closing the information gap of missing system quantities with consistent information transfer from simulation of relevant physical processes to the real world. Associated with this is the evaluation of the representation of relevant physical processes in the simulation. As elucidated in *Marçais and de Dreuzy* [2017], *Shen* [2018], and *Shen et al.* [2018] the progress and increasing availability of modern deep learning algorithms combined with the increasing availability of measured data open new possibilities to address these challenges.

To explore these possibilities I focus on one exemplary problem: flow field estimation of density-driven active solute transport observed in a small scale laboratory experiment within a Hele-Shaw cell, where high-resolution measurements of the solute concentration distribution are available.

Density-driven active solute transport is a relevant process for geological storage of anthropogenic CO_2 [*Weir et al.*, 1995; *Lindeberg and Wessel-Berg*, 1997; *Ennis-King and Paterson*, 2003, 2005]. Capturing atmospheric CO_2 and storing it in 1 to 3 km deep geological brine formations is, among others, one of the most promising techniques to mitigate climate change [*IPCC*, 2005]. At the prevalent conditions in these depths the supercritical CO_2, being trapped underneath impermeable cap rock, overlies the resident brine. The CO_2 dissolves into the brine leading to a local density increase at the interface. Eventually, this gives rise to density-driven instabilities drastically shortening the time scale of the mixing process in contrast to pure diffusion [*Ennis-King and Paterson*, 2003; *Hassanzadeh et al.*, 2005; *Yang and Gu*, 2006; *Farajzadeh et al.*, 2007; *Pruess and Zhang*, 2008; *Kneafsey and Pruess*, 2010; *Pau et al.*, 2010]. Density-driven flow is a key process in several other settings beyond CO_2-sequestration. Examples include the description of water dynamics beneath saline lake formations [*Wooding et al.*, 1997a, b], toxic and radioactive waste disposal [*Kolditz et al.*, 1998], and saltwater intrusion into exploited coastal aquifers [*Diersch and Kolditz*, 2002].

To investigate the dynamics of density-driven flow several experimental studies using optical observation of CO_2 and brine analogous solutions in Hele-Shaw cells have been

conducted at the laboratory scale [*Fernandez et al.*, 2002; *Oltean et al.*, 2004; *Kneafsey and Pruess*, 2010; *Backhaus et al.*, 2011; *Kneafsey and Pruess*, 2011; *Faisal et al.*, 2013; *Slim et al.*, 2013; *Faisal et al.*, 2015; *Ecke and Backhaus*, 2016; *Rasmusson et al.*, 2017; *Thomas et al.*, 2018]. As shown by *Thomas et al.* [2015] the use of color indicators often fails to completely visualize the flow patterns in Hele-Shaw cell experiments. Using a colored solute in water to introduce the density contrasts, simultaneously allows the accurate visualization of the solute concentration distribution with high resolution light transmission measurements [*Slim et al.*, 2013]. Contrary to the dense measurements of the concentration distribution, the flow field that describes the movement of the fluid remain experimentally inaccessible.

Optical flow estimation is a classical task in computer vision with the aim to estimate the motion of objects given two consecutive images. Typical applications are in autonomous driving [*Janai et al.*, 2017] and action recognition [*Simonyan and Zisserman*, 2014b]. The introduction of supervised deep learning using convolutional neural networks (CNNs) to the field of optical flow estimation in conjunction with training on synthetic data [*Dosovitskiy et al.*, 2015] has led to a paradigm shift [*Ilg et al.*, 2017]. CNNs with an encoder-decoder architecture to estimate motion showed state-of-the-art results on benchmark datasets, while enabling the estimation in real time.

de Bezenac et al. [2017] applied an adapted encoder-decoder CNN to a related system, the prediction of synthetically generated sea surface temperature data described by convection-diffusion. For this example they showed that the method can learn the underlying processes such that it is competitive with a numerical assimilation method. *Zhu and Zabaras* [2018] used encoder-decoder CNNs as surrogate models for uncertainty quantification in modeling steady-state single phase flow in heterogeneous media. With Bayesian treatment of the CNN by adopting the variational inference method of *Liu and Wang* [2016] and *Liu* [2017] they showed improved scalability to high-dimensional problems with limited training data. *Mo et al.* [2019] adopted the network architecture of *Zhu and Zabaras* [2018] and extended the model as a surrogate for uncertainty quantification of transient multiphase flow in heterogeneous media. Whereas these studies focus on replacing the forward models of related physical systems using an encoder-decoder CNN, in this work I use similar deep learning methods to aim at the estimation of missing system quantities. Generally these challenges are subject to methods like inverse modeling and data assimilation.

In this work, I explore the information transfer from synthetically generated data, representing the process understanding, to a laboratory experiment with missing flow field data using recent deep learning methods. For the information transfer to be coherent this requires (i) the physical processes that occur in the laboratory experiment to be represented completely and faithfully in the physical model and therefore in the synthetic data and (ii) the experimental and the synthetic data to have the same, in my case image-like, structure. In a first step, I generated a set of synthetic training data through numerical simulation of density-driven active solute transport. Based on FlowNet by *Dosovitskiy et al.* [2015] and FlowNet 2.0 by *Ilg et al.* [2017] and analogously to *de Bezenac et al.* [2017], *Zhu and Zabaras* [2018], and *Mo et al.*

[2019] I trained encoder-decoder CNNs, adaptations of FlowNet2-s and FlowNet2-SD [*Ilg et al.*, 2017], on the synthetic training data in an end-to-end fashion. In the encoder high-level features in the input concentration fields are extracted and transferred to a coarse abstract representation that is refined in the decoder to reconstruct the output flow fields. The trained CNN was then used to estimate flow fields from concentration measurements of a synthetic test dataset. This test dataset was again obtained from numerical simulation, but with the concentration boundary condition being modified to test the generalization of the method. In a next step, I applied the CNN to concentration measurements of a Hele-Shaw cell experiment. This way I estimated the flow fields that were otherwise inaccessible for density-driven flow and assessed the representation of the relevant physical transport processes in the numerical simulation.

With this approach the CNNs implicitly learn, relying purely on synthetic data, the phenomenology of a class of physical processes on a broad parameter spectrum. This incorporated representation of the physical processes can then be directly utilized to estimate missing system quantities on measurements. In this sense, I see this as a complementary approach to inverse modeling and data assimilation of physical processes. These methods are based on accurate models of the processes, where the missing system quantities are reconstructed by calibrating the forward model on the measured data.

The remainder of this dissertation is organized as follows. Chapter 2 introduces the theoretical background of active solute transport. Chapter 3 presents the numerical experiments conducted over a large parameter range to generate the synthetic datasets that represent the relevant transport processes. Chapter 4 introduces the laboratory experiment that is the target for the information transfer with respect to the flow field estimation. Chapter 5 provides deeper insight into the background of the utilized deep learning methods and details the CNNs that I trained on the numerical experiments to incorporate the process representation and used to estimate the flow fields in the laboratory experiment. In Chapter 6, I present and discuss the results of the flow field estimation on a synthetic test case and the laboratory experiment, before I draw a conclusion and give an outlook in Chapter 7.

2 | Active Solute Transport

The transport of a solute in a fluid can proceed passively by being convected purely due to externally imposed dynamics of the fluid, or actively due to influences of the solute on the flow resulting in additional driving forces. In this work, I consider active solute transport in porous media saturated with water. The presence of the solute considered here alters the density and may lead to instabilities that induce convection of the water.

This chapter introduces the mathematical description of the physical processes important to active solute transport. These involve the dynamics of pure water in porous media (Section 2.1) that are transferred to Hele-Shaw cells as an experimental model of the flow domain. In combination with dispersion, which arises from molecular diffusion and convective mixing due to the pore geometry, this describes the transport of a solute (Section 2.2). In both sections, the presentation is based on *Roth* [2017], where a more in-depth introduction to these topics is found. Accounting for the influences on the density due to the solute concentration results in the coupling to the dynamics of the water, where density-driven instabilities arise from unstable layering due to solute concentration gradients (Section 2.3).

2.1 Fluid Dynamics in Porous Media

A porous medium generally defines a medium that is composed of voids embedded in a solid matrix. In nature, soils can be described as such media that are composed of irregular solid grains of various sizes with pore space in between. In multiphase systems, the pore space can be occupied by multiple fluids, such as air, oil, and water. Here, I solely consider single phase water flow, where the pore space is completely saturated. Interconnectedness of the pore space is an important property to allow for water flow. These interconnected pores define the microscopic flow domain that can have quite complicated architectures. Therefore, when observing the flow at larger scales, the detailed description in the individual pores is intractable. Employing an upscaling from the microscopic pore scale to the macroscopic continuum scale by spatially averaging the system quantities overcomes this issue. Nonetheless, for this upscaling to be valid the existence of an Representative Elementary Volume (REV) is required. The REV is the minimal volume for which the averaged quantities become independent of variations in its shape and size. In this formulation the porosity ϕ is defined by the ratio of the pore space volume V_{pore} and the total volume V_{tot}:

$$\phi = \frac{V_{\text{pore}}}{V_{\text{tot}}}. \tag{2.1}$$

For saturated porous media the water content θ is equal to the porosity: $\phi = \theta$.

The water dynamics under isothermal conditions are described by the conservation of mass and the conservation of momentum. The following outlines the upscaling of the microscopic quantities to achieve the continuum formulation under the assumption of water to be an uniform and incompressible Newtonian fluid.

2.1.1 Conservation of Mass

The mass balance describes the rate of change of mass in the water saturated pore space V_w resulting from the flow across the pores at its boundary ∂V_w:

$$\frac{\partial}{\partial t}\int_{V_w}\rho_w^\mu \mathrm{d}V = -\int_{\partial V_w}\rho_w^\mu \mathbf{u}^\mu \cdot \mathrm{d}\mathbf{A} = -\int_{V_w}\nabla\cdot[\rho_w^\mu \mathbf{u}^\mu]\mathrm{d}V, \tag{2.2}$$

where, denoted with superscripted μ as the microscopic quantities, ρ_w is the water density and $\mathbf{u}$ is the velocity vector. The minus originates from setting the area element $\mathrm{d}\mathbf{A}$ to point outward and Gauss' theorem has been used in the second equality. Eq. (2.2) can be reformulated to

$$0 = \frac{\partial}{\partial t}\int_{V_w}\rho_w^\mu \mathrm{d}V + \nabla\cdot\int_{V_w}[\rho_w^\mu \mathbf{u}^\mu]\mathrm{d}V = \frac{\partial}{\partial t}\|V_w\|\langle\rho_w^\mu\rangle + \nabla\cdot\left[\|V_w\|\langle\rho_w^\mu \mathbf{u}^\mu\rangle\right], \tag{2.3}$$

where $\|V_w\|$ is the volume of V_w and the spatial averaging $\langle\dots\rangle$ must be supported by a REV, meaning that the pore space V_w must be embedded in an macroscopic volume element V that qualifies as REV. Dividing Eq. (2.3) by $\|V\|$, the volume of V, and recognizing that according to Eq. (2.1) $\theta = \phi = \|V_w\|/\|V\|$ for the saturated pore space, this yields the macroscopic mass balance, which formulates the continuity equation on this scale:

$$\frac{\partial}{\partial t}[\phi\rho_w] + \nabla\cdot[\rho_w \mathbf{j}_w] = 0, \tag{2.4}$$

wherein the macroscopic quantities result from the spatial averaging. Accordingly, the density is defined as $\rho_w = \langle\rho_w^\mu\rangle$ and the introduced macroscopic water flux as

$$\mathbf{j}_w = \phi\langle\mathbf{u}^\mu\rangle. \tag{2.5}$$

For the derivation of Eq. (2.4) from Eq. (2.3) to be valid $\langle\rho_w^\mu \mathbf{u}^\mu\rangle = \langle\rho_w^\mu\rangle\langle\mathbf{u}^\mu\rangle$ must hold. This is true for water flow in saturated porous media, as it can be argued that the correlation of ρ_w^μ and $\mathbf{u}^\mu$ is negligible (cf. *Roth* [2017]).

Assuming the porous media to be incompressible, the porosity ϕ and accordingly the water content θ are constant over time. In addition, under the assumption of the Oberbeck-Boussinesq approximation, the continuity equation can be simplified to

$$\nabla\cdot\mathbf{j}_w = 0, \tag{2.6}$$

where, according to *Oberbeck* [1879] and *Boussinesq* [1903], the variation of the density is neglected unless it is coupled to the influence of gravity.

2.1.2 Conservation of Momentum

The conservation of linear momentum of flow in the pores of the porous medium can be described by the stationary Stokes equation:

$$\mu_w \nabla^2 \mathbf{u}^\mu = \nabla p^\mu - \rho_w \mathbf{g}, \tag{2.7}$$

where μ_w is the dynamic viscosity of water, p^μ the pressure, and $\mathbf{g}$ the acceleration of gravity. For Eq. (2.7) to be a valid description, several assumptions have to be made: (i) The water is assumed to be incompressible and its temperature is set constant justifying constant ρ_w and μ_w. (ii) The external forcing of the system has to be slow when measured on the inherent time scale, allowing for the time-independent formulation. (iii) Inertia has to be negligible, which holds for small pores as the effect of the water-matrix interface is very strong eliminating turbulences in the flow.

In this setting, the velocity $\mathbf{u}^\mu$ is parallel to $-\nabla^2 \mathbf{u}^\mu$. This is because $\mathbf{u}^\mu$ adjusts such that the viscous term balances the driving potential gradient, left and right hand side of Eq. (2.7), respectively. This yields

$$\mathbf{u}^\mu = -\kappa(\mathbf{x}) \nabla^2 \mathbf{u}^\mu = -\frac{\kappa(\mathbf{x})}{\mu_w} [\nabla p^\mu - \rho_w \mathbf{g}] \tag{2.8}$$

with the introduced linear scaling $\kappa(\mathbf{x})$ that incorporates the complicated pore space geometry. Upscaling of Eq. (2.8) supported by a plane REV results in the continuum formulation:

$$\mathbf{j}_w = -\phi \frac{1}{\mu_w} \langle \kappa [\nabla p^\mu - \rho_w \mathbf{g}] \rangle \tag{2.9}$$

with the macroscopic water flux $\mathbf{j}_w = \phi \langle \mathbf{u}^\mu \rangle$ and the averaging $\langle \dots \rangle$ as before.

Note that the stationary Stokes equation is linear, meaning that if $\{\mathbf{u}, p\}$ solves Eq. (2.7) then also $\{\alpha \mathbf{u}, \alpha p\}$ does. This implies that the magnitudes of the microscopic pressure gradient ∇p^μ is proportional to the macroscopic pressure gradient ∇p. Yet, due to the pore space geometry, they need not to be parallel. This relation can be described by $\nabla p^\mu = \mathbf{a}(\mathbf{x}) \nabla p$, where $\mathbf{a}(\mathbf{x})$ is a second rank, symmetric tensor. Further recognizing that κ is independent of $\mathbf{u}$ and ∇p renders the averaging and thus the upscaling to be valid. This leads to Darcy's empirical flux law:

$$\mathbf{j}_w = -\frac{\mathbf{k}}{\mu_w} [\nabla p - \rho_w \mathbf{g}], \tag{2.10}$$

where the permeability $\mathbf{k}$, again a second rank, symmetric tensor, absorbs κ and $\mathbf{a}(\mathbf{x})$: $\mathbf{k} = \phi \langle \kappa \mathbf{a} \rangle$. Therefore, $\mathbf{k}$ relates the response of the flow to the forcing by also representing the anisotropy of the medium. In Eq. (2.10) the volumetric water flux is also denoted as the Darcy velocity $\mathbf{u} = \mathbf{j}_w$ representing the flow field.

2.1.3 Hele-Shaw Cells as Models of Porous Media

Hele-Shaw cells [*Hele-Shaw*, 1898] are simplified physical models of porous media that make it possible to easily observe the fluid flow. They consist of two parallel glass

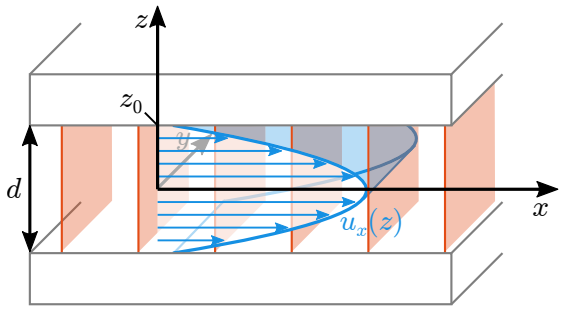

Figure 2.1: Illustration of the parabolic Hagen-Poiseuille velocity profile (blue) between two parallel plates (gray). The driving is due to a large scale pressure field whose isosurfaces are depicted in red. Modified from *Roth* [2017].

plates, separated by a small gap of width d that serves as the flow domain. To be a valid model for a porous medium d is required to be so small that the influence of the fluid-glass interface on the dynamics is large and the flow can be described by the stationary Stokes equation (Eq. (2.7)). In this case, the same governing equations as for the continuum formulation for porous media arise by averaging over the gap. For Hele-Shaw cells, the water saturated gap d embedded between the glass plates results in a porosity of $\phi = 1$ and the requirement of small d combined with the construction inevitably restricts the observations to quasi 2-dimensional flow.

Conservation of Mass The mass balance in the gap of a Hele-Shaw cell directly describes the conservation of mass. It takes the form

$$\frac{\partial \rho_w}{\partial t} + \nabla \cdot [\rho_w \mathbf{u}] = 0, \tag{2.11}$$

where, because of the relation in Eq. (2.5) and $\phi = 1$, the quantities in the continuum scale correspond to the averaged microscopic ones and $\mathbf{u} = \langle \mathbf{u}^\mu \rangle$ is the macroscopic velocity. Again, invoking the Oberbeck-Boussinesq approximation this reduces to

$$\nabla \cdot \mathbf{u} = 0. \tag{2.12}$$

Conservation of Momentum As depicted in Fig. 2.1, consider horizontal oriented glass plates to allow for a more compact formulation by omitting the influence of gravity. For small d, the conservation of linear momentum is then given by the stationary Stokes equation:

$$\mu \nabla^2 \mathbf{u} = \nabla p, \tag{2.13}$$

where the driving pressure gradients are parallel to the glass plates. Let the gap width $d = 2z_0$ and choose the coordinate system such that the x- and y-directions are parallel to the glass plates, which are situated at z_0 and $-z_0$ respectively. With the constraint

that $\mathbf{u}(z_0) = \mathbf{u}(-z_0) = \mathbf{0}$ the parabolic Hagen-Poiseuille velocity profile (cf. Fig. 2.1) establishes between the glass plates:

$$\mathbf{u}(z) = -\frac{1}{2\mu}\nabla p[z_0^2 - z^2], \tag{2.14}$$

which is a solution of Eq. (2.13). Integrating over the gap width d in direction of z yields Darcy's empirical flux law for the specific geometry of a Hele-Shaw cell:

$$\mathbf{u} = -\frac{d^2}{12\mu}\nabla p. \tag{2.15}$$

For the horizontal orientation of the glass plates the pressure gradient drives the flow. In stationary flows of isotropic fluids it is admissible to replace $-\nabla p$ by the sum of all driving forces. Reintroducing the influence of gravity, when the glass plates are oriented vertically, leads to

$$\mathbf{u} = -\frac{d^2}{12\mu}[\nabla p - \rho_w \mathbf{g}]. \tag{2.16}$$

Comparison to Eq. (2.10) identifies that the permeability $\mathbf{k}$ reduces to a scalar $k = d^2/12$ for Hele-Shaw cells.

2.2 Solute Transport

This section introduces the passive transport of a solute added to the water. Here, the solute is considered to act as a passive, conservative tracer, meaning that its mass in the fluid phase is conserved and it does not alter the properties of water. Generally, this holds for nonreactive solutes and small concentrations. In this case, the movement of the solute particles can be considered to be superposed to the dynamics of a pure fluid. For larger concentrations, however, the solute can affect the density of the fluid, leading to a coupling to the gravitational driving. These effects of active solute transport are discussed in Section 2.3.

Passive solute transport is governed by three processes. Molecular diffusion smooths any solute concentration gradients, while convection with the mean flow redistributes the solute. On the continuum scale, the combined effects then result in dispersion. Analogously to the dynamics of the fluid, first the concepts are introduced considering the microscopic pore scale that are then upscaled to achieve the formulation on the continuum scale.

2.2.1 Molecular Diffusion

At the molecular scale the solute particles are diffusively transported by Brownian motion. This resembles an undirected random walk of the molecules that effectively

attenuates any given gradient of the solute concentration ∇C. The resulting transport in the fluid is described by Fick's first law [*Fick*, 1855]:

$$\mathbf{j}_s = -D_m \nabla C \tag{2.17}$$

that relates the flux of solute mass $\mathbf{j}_s$ to the concentration gradient ∇C with the constant of proportionality D_m being the molecular diffusion coefficient. Hence, D_m is assumed to be isotropic, but depends on the properties of the solute and the fluid. Combining Eq. (2.17) with the conservation of solute mass:

$$\frac{\partial C}{\partial t} + \nabla \cdot \mathbf{j}_s = 0 \tag{2.18}$$

yields Fick's second law that is also referred to as diffusion equation:

$$\frac{\partial C}{\partial t} - D_m \nabla^2 C = 0, \tag{2.19}$$

which describes the development of the concentration in time due to its gradients. With a typical value of a diffusion coefficient being $D_m = 10^{-10}\,\mathrm{m^2 s^{-1}}$, purely diffusive transport is only sufficiently fast over very small distances.

2.2.2 Dispersion

Flow in porous media and generally in thin flow domains results in heterogeneous velocity fields. In longitudinal direction the velocities are altered due to the pore geometry. Given a stationary driving, velocities are higher in small diameter pores, while slower in large diameter pores. In transversal direction the velocities are affected by the boundaries. Close to the water-solid interfaces friction leads to slow flow that influences the velocities across the entire pore cross-section. Consequently, the interplay of molecular diffusion with the latter effect can result in further dispersion of the solute. This is known as Taylor-Aris dispersion. In combination with diffusive and convective mixing at the junctions of the pores this leads to the effect of hydrodynamic dispersion.

Taylor-Aris Dispersion Consider the flow in a single pore, for instance a small gap between parallel glass plates (Fig. 2.1), where the x-direction is chosen to point in the direction of the mean flow. In this geometry the velocity field is given by the Hagen-Poiseuille profile as in Eq. (2.14). Now consider an initial pulse of solute particles located at a distinct cross-section perpendicular to the mean flow. According to the velocity field, the particles get spread in the longitudinal direction, populating the parabolic profile. Due to molecular diffusion, the particles also move in transverse direction and, therefore, jump from stream lines in the center of the pore, where the flow is fast, to streamlines closer to the boundaries, where the flow is slow and vice versa. For sufficiently long time scales the particles are able to travel across the gap multiple times, enabling them to encounter the entire velocity field in transverse

direction. The repeated acceleration and deceleration, relative to each other, spreads the particles in the longitudinal direction. For the geometry of cylindrical pores this effect was initially described by *Taylor* [1953] and later extended to arbitrary cross-sections by *Aris* [1956] and is therefore referred to as Taylor-Aris dispersion.

Although dispersion emerges from a different process than molecular diffusion, its formal description is the same, except for an adapted effective diffusion coefficient:

$$D_{\text{eff}} = D_m + \alpha \frac{\bar{u}^2 r_0^2}{D_m} \tag{2.20}$$

with the mean velocity $\bar{u}$, the characteristic length scale r_0 of the cross-sectional area, and a dimensionless parameter α that represents the pore geometry. Note that the additional second term is proportional to $1/D_m$. This means that for smaller values of D_m and therefore slower mixing in transverse flow direction, the particles experience stronger dispersion in the direction of the mean flow. Accounting for the transport along with the mean flow, the solute flux in x-direction is then given by

$$j_s = \bar{u}C - D_{\text{eff}} \frac{\partial C}{\partial x}, \tag{2.21}$$

where $\bar{u}C$ is the convective flux.

This description in Eqs. (2.20) and (2.21) reflects that mixing due to dispersion is only apparent in moving fluids. For absent flow, the effective dispersion coefficient D_{eff} reduces to the coefficient of molecular diffusion D_m that always contributes to the mixing.

While Taylor-Aris dispersion describes the mixing process in single pores, additional mixing occurs at the junctions of the pore network. Considering the two situations illustrated in Fig. 2.2, mixing can range from purely diffusive mixing to dominantly convective mixing.

Diffusive Mixing The charactersitic length l of the pore junction (cf. Fig. 2.2 (left)) defines the contact length of fluid that originates from different pores. This determines the available time $l/\bar{u}$ for solute particles to diffuse in transverse flow direction, where $\bar{u}$ is the mean velocity in the junction. Conceptually, this process is equivalent to Taylor-Aris dispersion in a short pore, where the contact time is too short to allow diffusion across the entire characteristic radius r_0. Assuming the pore network exhibits a characteristic distance between pore junctions Λ, the ratio l/Λ gives the fraction of the fluid travel distance for which diffusive mixing does occur. With decreasing l and increasing Λ the effective mixing in transverse flow direction is reduced. Similar to Taylor-Aris dispersion, an effective dispersion coefficient arises:

$$D_{\text{eff}} = \alpha' \frac{\Lambda}{l} \frac{\bar{u}^2 r_0^2}{D_m}, \tag{2.22}$$

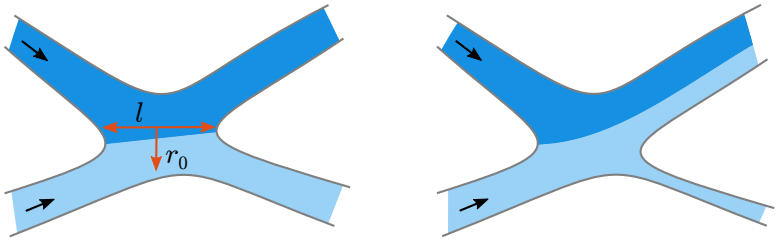

Figure 2.2: Mixing processes at pore junctions of fluid with high solute concentration (darker blue) and fluid with low concentration (lighter blue). Purely diffusive mixing (left): The fluids are in contact over the length l. Within the time of contact that depends on l and the flow velocity, solute particles can diffuse into the low concentration fluid. Dominantly convective mixing (right): The fluid with low concentration gets partially merged into a pore with high solute concentration. Subsequently, the two fluids mix due to Taylor-Aris dispersion. Modified from *Roth* [2017].

that is larger for a slower effective transverse mixing process. Here again, α' is a dimensionless parameter that represents the geometry of the junction, $\bar{u}$ the mean velocity, and D_m the molecular diffusion coefficient.

Convective Mixing A fundamentally different process contributing to dispersion of the solute is convective mixing in pore junctions (Fig. 2.2 (right)) that is referred to as hydromechanic dispersion. Assuming laminar flow and neglecting molecular diffusion, the pore geometry exclusively determines the fraction κ of the fluid that gets merged with fluid of different solute concentration. The number of junctions the fluid needs to encounter for complete mixing is then given by $1/\kappa$. Again, assuming that the characteristic distance between the junctions is described by Λ, the travel distance for complete mixing is Λ/κ and the time needed to do so is $\Lambda/\kappa\bar{u}$. In this case the dispersion coefficient is proportional to $\bar{u}$ and given by

$$D_{\text{eff}} = \beta\Lambda\bar{u} = \lambda\bar{u}, \tag{2.23}$$

where the constant β incorporates κ and the dispersivity λ describes the characteristic extent of the structures that contribute to the mixing.

The dispersive processes being present for solute transport at the microscopic pore scale are manifold. Their representation involves the effects of molecular diffusion, the coupling of the latter to the flow field, and the influence of the geometry of the pore network itself. Mainly depending on the mean flow velocity, their individual contribution can vary greatly. Commonly, the combination of the introduced processes is described by the summarizing term of hydrodynamic dispersion.

To this point only dispersion in the direction of the flow has been considered leading to D_{eff} being represented as a scalar quantity. Dispersion, however, is generally different in longitudinal and transverse direction and is therefore described by the tensor D_{eff}. For transport in an isotropic medium and under the assumption of purely hydromechanic dispersion *Bear* [1961] and *Scheidegger* [1961] provide a theoretical description. In this case, the components of D_{eff} are given by

$$D_{ij} = [\lambda_l - \lambda_t]\frac{u_i u_j}{|\mathbf{u}|} + \lambda_t |\mathbf{u}| \delta_{ij} \tag{2.24}$$

with longitudinal and transverse dispersivities λ_l and λ_t, mean velocity $\mathbf{u}$ with components u_i and u_j, and Kronecker's delta δ_{ij}.

2.2.3 Conservation of Solute Mass

Like the description of the dynamics of pure fluids it rapidly becomes unfeasible to describe solute transport at the pore space, when considering larger spatial scales. Therefore, the same approach of upscaling the microscopic description to arrive at the macroscopic continuum formulation is introduced. Analogously, the upscaling needs to be supported by the existence of an averaging volume V that has to be an REV, not only for the pore space and the water content, but in addition for the solute concentration. The solute mass contained in the averaging volume then is

$$\int_{V_w} C_w^\mu \mathrm{d}V = \|V_w\| \langle C_w^\mu \rangle, \tag{2.25}$$

where the concentration C_w is contained in the fraction V_w of V that is occupied with water with volume $\|V_w\|$. Again, the microscopic quantities are identified with superscripted μ and the spatial averaging is denoted by $\langle \dots \rangle$. Dividing by the volume $\|V\|$ of V and identifying $\theta = \phi = \|V_w\|/\|V\|$ yields the macroscopic total concentration:

$$C_{\text{tot}} = \frac{\|V_w\|}{\|V\|} \langle C_w^\mu \rangle = \phi \langle C_w^\mu \rangle = \phi C_w, \tag{2.26}$$

where the relation to the macroscopic concentration in the water phase C_w is given by the volume fraction ϕ for a saturated porous medium.

As already given in Eq. (2.21), the solute transport at the pore scale is composed of the convective and the dispersive transport. The solute flux $\mathbf{j}_s^\mu$ is then described by

$$\mathbf{j}_s^\mu = \mathbf{u}^\mu C_w^\mu - \mathsf{D}_{\text{eff}}^\mu \nabla C_w^\mu, \tag{2.27}$$

where $\mathbf{u}$ is the velocity field, C_w the solute concentration in the fluid, and D_{eff} the hydrodynamic dispersion tensor. With this, we can set the balance for the solute mass to be

$$\frac{\partial}{\partial t} \int_{V_w} C_w^\mu \mathrm{d}V = -\int_{\partial v_w} \mathbf{j}_s^\mu \cdot \mathrm{d}\mathbf{A} = -\int_{V_w} \nabla \cdot [\mathbf{u}^\mu C_w^\mu - \mathsf{D}_{\text{eff}}^\mu \nabla C_w^\mu] \mathrm{d}V, \tag{2.28}$$

where Gauss' theorem has been utilized in the second equality. Again dividing by $\|V\|$ and spatial averaging results in

$$\phi \frac{\partial C_w}{\partial t} + \nabla \cdot [\phi \langle \mathbf{u}^\mu C_w^\mu \rangle] - \nabla \cdot [\phi \langle \mathsf{D}_{\text{eff}}^\mu \nabla C_w^\mu \rangle] = 0, \tag{2.29}$$

where the correlation terms $\langle \mathbf{u}^\mu C_w^\mu \rangle$ and $\langle \mathsf{D}_{\text{eff}}^\mu \nabla C_w^\mu \rangle$ emerge. For the discussion of these terms and the argumentation how the involved quantities can be rendered independent the reader is referred to *Roth* [2017] at this point.

For a macroscopic uniform porous medium and slow flow velocities, D_{eff} can be approximated to be independent of the location $\mathbf{x}$. With the continuity equation, as presented in Eq. (2.6), Eq. (2.29) simplifies to

$$\phi \frac{\partial C_w}{\partial t} + \mathbf{u} \cdot \nabla C_w - \phi \mathsf{D}_{\text{eff}} \nabla^2 C_w = 0, \tag{2.30}$$

where $\mathbf{u} = \mathbf{j}_w = \phi \langle \mathbf{u}^\mu \rangle$ is now the Darcy velocity representing the flow field. Eqs. (2.29) and (2.30) both formulate the convection-dispersion equation that represents the solute transport on the macroscopic continuum scale.

2.3 Density-Driven Instabilities

The description of passive solute transport is based on the assumption that the solute does not affect the flow (cf. Section 2.2). In general this is not true, as the solute can alter the properties of the fluid. In the following, the influence on the density of water ρ_w is considered and the resulting flow dynamics in porous media is introduced.

Given a heterogeneous solute distribution, unstable fluid layering can occur, meaning that denser fluid is situated above lighter fluid. This situation can lead to Rayleigh-Taylor like instabilities that, given their cause, are referred to as density-driven instabilities in this work. Let such a system be initially at rest and without any additional external driving. Due to the instabilities, convection sets in and regions of downwelling and upwelling develop. Hence, the transport is actively driven by the presence of the solute itself and causes a regime shift from purely diffusive to convective transport. For instance, this is very important to the application of carbon dioxide sequestration, where the time scale of the transport process may reduce from thousands to hundreds of years [*Hassanzadeh et al.*, 2005].

2.3.1 Dynamics

Consider a 2-dimensional system with the initial condition as depicted in Fig. 2.3. The flow domain represents a water saturated porous medium at the macroscopic continuum scale as introduced in Section 2.1. Let the upper and lower boundary be impermeable to the flow, while the upper boundary is in contact with a solute-bearing layer. Representing the concentration boundary condition with $C_{\max}$, there the solute can infiltrate into the flow domain, increasing the density of water at the very top. The

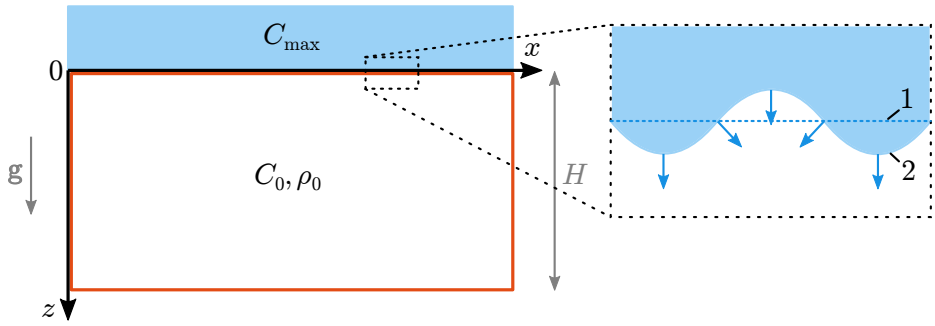

Figure 2.3: Initial condition for the density-driven instability: The flow domain of height H (red rectangle) represents a water saturated porous medium with C_0 and ρ_0. The system is at rest. The solute-bearing layer (blue) with $C_{\max} > C_0$ above the domain sets the upper concentration boundary condition. Detail: Illustration of a small perturbation of an exemplary wavelength being present at the boundary that displaces the interface from position 1 to position 2. The blue arrows indicate the direction of subsequent smoothing due to molecular diffusion.

dynamics of this system are described by the combination of the continuity equation (Eq. (2.6)), Darcy's empirical flux law (Eq. (2.10)), and the convection dispersion equation (Eq. (2.30)):

$$\nabla \cdot \mathbf{u} = 0, \tag{2.31}$$

$$\mathbf{u} = -\frac{k}{\mu_w}[\nabla p - \rho_w(C_w) g \mathbf{e}_z], \tag{2.32}$$

$$\phi \frac{\partial C_w}{\partial t} = -\mathbf{u} \cdot \nabla C_w + \phi \mathsf{D}_{\text{eff}} \nabla^2 C_w, \tag{2.33}$$

where the medium is assumed to be isotropic, resulting in scalar permeability k, and the gravitational acceleration is split into the scalar g and the unit vector in z-direction $\mathbf{e}_z$. Because of the 2-dimensional formulation, the Darcy velocity is $\mathbf{u} = (u_x, u_z)^\top$ and the gradients are defined by $\nabla = (\partial/\partial x, \partial/\partial z)^\top$. Coupling Eq. (2.32) to Eq. (2.33), the water density ρ_w now depends on the solute concentration C_w. Hence, the flow is driven by the presence of the solute. For small variation in C_w this relation is linearly approximated by

$$\rho_w(C_w) = \rho_0 + \Delta\rho_w \tilde{C}_w \tag{2.34}$$

with the density difference $\Delta\rho_w = \rho_w(C_{\max}) - \rho_w(C_0)$ and the reduced concentration

$$\tilde{C}_w = [C_w - C_0]/[C_{\max} - C_0]. \tag{2.35}$$

Instability Onset With the interface initially being perfectly flat, the weight of the denser fluid above is perfectly supported by the fluid beneath. In this case the transport

would be purely diffusive. However, the natural presence of small perturbations at the interface arising from thermal fluctuations and vibrations can lead to an onset of convection. The illustration of an exemplary perturbation wavelength is given in the detail of Fig. 2.3. In general, a broad wavelength spectrum of perturbations is present promoting the onset of convection. Let's consider the 1-dimensional formulation of Darcy's flux law in direction of gravity to get insight into the amplification of the initial perturbations:

$$\frac{\partial p}{\partial z} = -\frac{\mu_w}{k} u_z + \rho_w(C_w) g. \tag{2.36}$$

We can derive the resulting driving due to an pressure change δp when considering a small displacement δz of the interface from position 1 with $\rho_w(C_{\max})$ to position 2 with $\rho_w(C_0)$ to be

$$\delta p = \frac{\partial p_2}{\partial z}\delta z - \frac{\partial p_1}{\partial z}\delta z = \left[\rho_w(C_0) - \rho_w(C_{\max})\right] g\, \delta z, \tag{2.37}$$

where the initial flow velocity is assumed to be $u_z = 0$. Since $\rho_w(C_0) < \rho_w(C_{\max})$, this results in $\delta p/\delta z < 0$, which represents an increased driving force. Hence, the initial displacement promotes the onset of convection.

In contrast to the amplification of the perturbations, molecular diffusion can stabilize the situation by smoothing out the perturbations sufficiently fast before they can grow. According to Eq. (2.17) and as depicted in the detail of Fig. 2.3, net diffusive transport is perpendicular to the interface. Due to the geometry, the diffusive flux into regions where the interface is concave is effectively enhanced and may lead to the vanishing of the perturbation.

The interplay between the small perturbations promoting convection and the smoothing due to molecular diffusion determines the onset of the instability and is governed by three aspects: (i) for small distances, diffusive transport is sufficiently fast to smooth out the perturbation, (ii) convective growth of the perturbations is stronger for shorter wavelengths, and (iii) driving of convection is larger for larger density gradients. The combination of these aspects defines whether the situation gets unstable and, if it does, a critical wavelength λ_{crit} of the perturbation that experiences the strongest amplification. Subsequently, macroscopic convection patterns develop according to λ_{crit}. Conducting a linear stability analysis of the system, the relation

$$\lambda_{\text{crit}} = \frac{2\pi\mu_w D_m}{0.07 k \Delta\rho_w g} \tag{2.38}$$

can be deduced for the regime where convection actually does occur [*Riaz et al.*, 2006].

Temporal development After the onset, the system is subject to a transient relaxation process, where the dense fluid is transported toward the bottom boundary and the flow domain eventually saturates. As there is no solute sink at the bottom this process is different from the more commonly studied asymptotic regime of Rayleigh-Bénard convection observed for heat transport in free fluids [*Bénard*, 1900; *Rayleigh*,

1916] and its porous medium analog [*Horton and Rogers Jr*, 1945; *Lapwood*, 1948]. There, the heat source at the bottom and the sink at the top lead to a dynamic equilibrium in the system's final state.

According to a number of numerical and experimental studies the system encounters several temporal regimes [e.g., *Slim et al.*, 2013; *Slim*, 2014; *Kreyenberg*, 2015; *Ecke and Backhaus*, 2016; *Thomas et al.*, 2018]. At first, small density fingers arising from the unstable perturbation wavelength grow linearly. When they have grown sufficiently large, they start to interact nonlinearly through coupling of their surrounding flow fields. The flow velocities along the fingers points downward due to the negative buoyancy of the denser fluid, while the velocities between them points upward to balance the downward flow. Irregularities in the rising of less dense fluid can push fingers closer together, causing them to merge. Subsequently, the merged fingers form larger plumes that dominate the temporal development of their environment. Because their seeding points at the upper boundary are now spaced coarsely, new small fingers form similarly to the initial finger formation at the onset. The larger plumes then feed on the reinitialized fingers as the dominant flow field pushes them toward the plumes. This process persists until the fastest plumes reach the bottom boundary and saturation of the flow domain begins.

2.3.2 Dimensionless Formulation

To get a deeper understanding of the system's development according to the boundary condition and the water and solute properties, it is useful to introduce a dimensionless formulation of Eqs. (2.31) to (2.33). To achieve this, the system is scaled by its characteristic length scale L_c, typical velocity U_c, and intrinsic time scale T_c. There are two reasonable choices for the characteristic length scale L_c: (i) the usual approach is to scale by the height of the flow domain as in [e.g., *Riaz et al.*, 2006; *Wooding et al.*, 1997a] and (ii) the more elegant approach by *Slim and Ramakrishnan* [2010] (SR2010) is to scale by the convection-diffusion length. Both are discussed in the following, whereas the generic denotion L_c is used for now. The characteristic velocity that is encountered in the temporal development of the instability is given by the buoyancy velocity $U_\mathrm{c} = \Delta\rho_w g k/\mu_w$, which is defined as the sinking velocity of a water parcel with density $\rho_w(C_\mathrm{max})$ in a surrounding with $\rho_w(C_0)$. For the entirety of the characteristic quantities this results in

$$L_\mathrm{c}, \quad U_\mathrm{c} = \frac{\Delta\rho_w g k}{\mu_w}, \quad T_\mathrm{c} = \frac{L_\mathrm{c}}{U_\mathrm{c}}, \quad P_\mathrm{c} = \frac{\mu_w U_\mathrm{c} L_\mathrm{c}}{k}, \quad \text{and} \quad C_\mathrm{c} = C_\mathrm{max} - C_0 \tag{2.39}$$

with the additional characteristic pressure P_c and concentration C_c. This leads to

$$\mathbf{u} = U_\mathrm{c}\tilde{\mathbf{u}}, \quad \mathbf{x} = L_\mathrm{c}\tilde{\mathbf{x}}, \quad t = \phi T_\mathrm{c}\tilde{t}, \quad p - \rho_0 g z \mathbf{e}_z = P_\mathrm{c}\tilde{p}, \quad \text{and} \quad C_w - C_0 = C_\mathrm{c}\tilde{C}_w, \tag{2.40}$$

where $\tilde{\cdot}$ denotes the now dimensionless quantities and $\tilde{C}_w$ is again the reduced concentration as defined above. Choosing L_c to either be the domain height or the convection-diffusion length leads to different dimensionless formulations.

Scaling by the Domain Height With Eq. (2.40) and $L_c = H$ the scaling of Eqs. (2.31) to (2.33) results in

$$\tilde{\nabla} \cdot \tilde{\mathbf{u}} = 0, \tag{2.41}$$

$$\tilde{\mathbf{u}} = -\tilde{\nabla}\tilde{p} + \tilde{C}_w \mathbf{e}_z, \tag{2.42}$$

$$\frac{\partial \tilde{C}_w}{\partial \tilde{t}} = -\tilde{\mathbf{u}} \cdot \tilde{\nabla}\tilde{C}_w + \frac{1}{\mathrm{Ra}} \tilde{\nabla}^2 \tilde{C}_w. \tag{2.43}$$

While the continuity equation and the flux law render independent of any parameter, the Rayleigh number emerges in the convection dispersion equation:

$$\mathrm{Ra} = \frac{\Delta\rho_w g k H}{\mu_w \phi D_{\mathrm{eff}}}. \tag{2.44}$$

Note that dispersion is assumed to be isotropic here and $\mathsf{D}_{\mathrm{eff}}$ therefore reduces to D_{eff}. In this formulation Ra is interpreted as the scaling parameter between the convective and the diffusive term and determines the instability of the system by quantifying the aspects as introduced in the discussion of the instability onset above. For small Ra, diffusion is dominant and smooths out perturbations before they can lead to convection. Above a certain threshold value $\mathrm{Ra}_{\mathrm{crit}}$ that depends on the initial and boundary condition [*Nield and Bejan*, 2006] convection sets in and leads to the formation of the density fingers. Values of $\mathrm{Ra}_{\mathrm{crit}}$ have been reported to be in the range between $\mathcal{O}(40)$ and $\mathcal{O}(500)$ [*Lapwood*, 1948; *Lindeberg and Wessel-Berg*, 1997; *Graf et al.*, 2002; *Kneafsey and Pruess*, 2010]. Comparing the definition of the Rayleigh number with Eq. (2.38) yields the relation $\lambda_{\mathrm{crit}} \propto H(\phi \mathrm{Ra})^{-1}$ and therefore for the scaled critical wavelength

$$\tilde{\lambda}_{\mathrm{crit}} = \lambda_{\mathrm{crit}}/H \propto (\phi \mathrm{Ra})^{-1} \tag{2.45}$$

in this formulation.

Scaling by the Convection-Diffusion Length SR2010 propose the convection-diffusion length as characteristic length scale $L_c = \phi D_{\mathrm{eff}}/U_c$, which defines the length over which diffusion and convection do balance. Scaling with this length yields

$$\tilde{\nabla} \cdot \tilde{\mathbf{u}} = 0, \tag{2.46}$$

$$\tilde{\mathbf{u}} = -\tilde{\nabla}\tilde{p} + \tilde{C}_w \mathbf{e}_z, \tag{2.47}$$

$$\frac{\partial \tilde{C}_w}{\partial \tilde{t}} = -\tilde{\mathbf{u}} \cdot \tilde{\nabla}\tilde{C} + \tilde{\nabla}^2 \tilde{C}_w. \tag{2.48}$$

In contrast to the scaling by H this formulation is independent of any dimensionless parameter, revealing the universal development of the system. This means that the transport in any saturated porous medium with an unstable layering of a more dense fluid layer above a less dense one (cf. Fig. 2.3) will undergo the exact same temporal development independent of the initial values. However, the system is bounded in space

by the lower boundary. Therefore, the temporal development halts, when the fastest plumes reach the bottom, the domain begins to saturate, and the transport starts to shut down. In fact, scaling the position of the lower boundary by the convection-diffusion length results in

$$\tilde{H} = \frac{H}{L_c} = \frac{\Delta\rho_w g k H}{\mu_w \phi D_{\text{eff}}} = \text{Ra}, \tag{2.49}$$

which is identified to be the Rayleigh number in this formulation. Also quantifying the instability of the system, Ra is interpreted as the dimensionless height of the domain that bounds the temporal development. For a shallow dimensionless formulated system Ra is small and only diffusive transport is encountered until the bottom is reached. Above a critical value the system is deep enough for convection to develop. Further increased Ra leads to the development of more and more temporal regimes before transport shuts down.

Comparison Although the two dimensionless formulations, as introduced above, seem to be quite simple, their implications may not be intuitive. Deeper understanding can be achieved by observing the response of the system and its dimensionless formulations to variation of the quantities defining the initial condition, i.e. domain height H, porosity ϕ, dispersion coefficient D_{eff}, density difference $\Delta\rho_w$, and permeability k. For brevity the scaling by the domain height is referred to as classical formulation, whereas the scaling by the convection-diffusion length is referred to as formulation of SR2010 in the following.

Consider a dimensional system 1 of height H with an arbitrary but fixed width. At a fixed time t, let system 1 exhibit two initial density fingers right after convection onset. Now, change the quantities individually to represent a system 2 such that the Rayleigh number doubles in value: $\text{Ra}_2 = 2\text{Ra}_1$. The resulting changes of system 2 together with the changes in the dimensionless scales reveal the differences between the classical formulation and the formulation of SR2010. This proceeding is illustrated in Fig. 2.4. In the upper left of the figure, the considered dimensional system 1 is depicted along with the general formulation of the characteristic scales. Note that for the classical formulation and the formulation of SR2010, Ra and U_c have the same form, while the definitions of L_c and T_c differ significantly. The scales are correspondingly superscripted with H for the domain height and CD for the convection-diffusion length. In the following the individual initial quantities are varied and the implications are discussed:

- **Case 1:** $H_2 = 2H_1$ (upper right in Fig. 2.4). Despite the change in depth, system 2 does not change, still exhibiting two initial fingers at time t. However, $\text{Ra}_2 = 2\text{Ra}_1$ indicates a more unstable situation. For the formulation of SR2010 this does make sense, as larger Ra relates to a deeper domain and consequently leads to the development of more temporal regimes, which we expect for the dimensional system in this case. Accordingly, L_c^{CD} and T_c^{CD} remain unchanged.

For the classical formulation, on the other hand, the increase of Ra also implies a more unstable situation. According to Eq. (2.45), this leads to an increased number of initial fingers per unit length. In the dimensionless formulation this is true, since $L^H_{c,2} = 2L^H_{c,1}$, but there is no direct implication for the dimensional system in this case. The fact that $T^H_{c,2} = 2T^H_{c,1}$ represents that time t corresponds to an earlier point in time in this formulation, which similarly to the formulation of SR2010 implies that the system develops further in time. Therefore, the implication of both dimensionless formulations are similar for the dimensional system, however, the interpretation of the classical formulation is more obscure in the case of the variated domain height H.

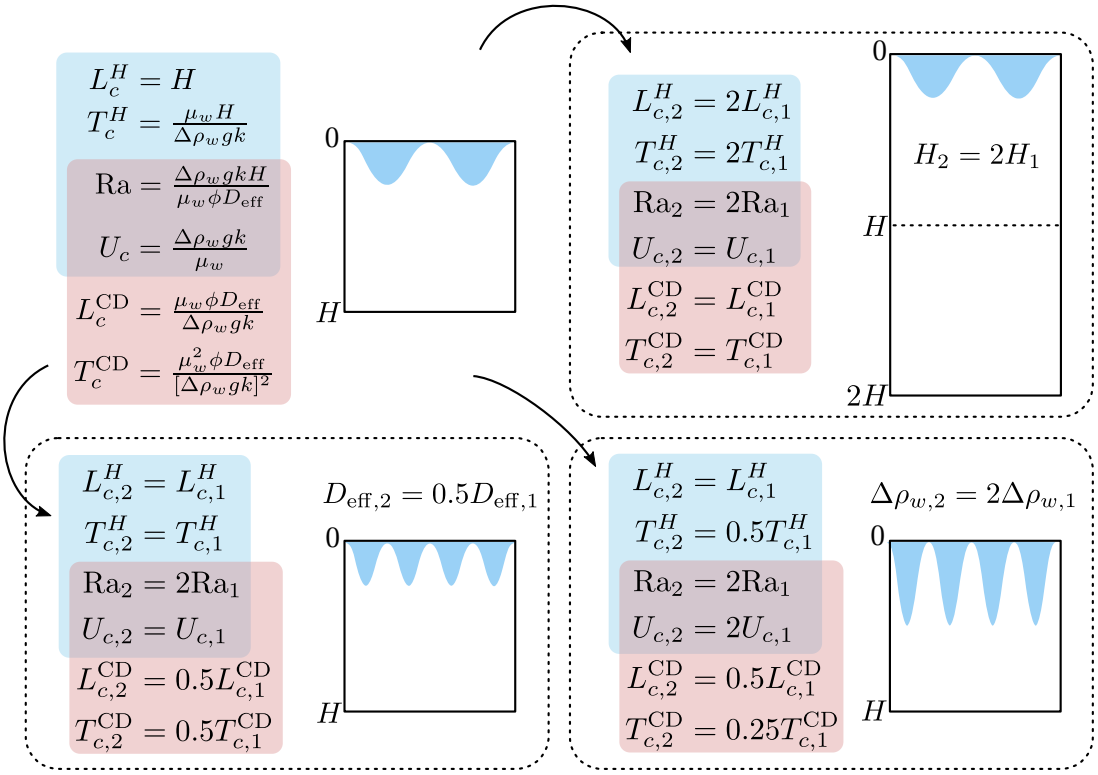

Figure 2.4: Comparison of the characteristic scales as obtained from scaling by the domain height (light blue) and the convection-diffusion length (light red): A dimensional system with domain height H exhibits two initial density fingers at time t (upper left). Snapshots at t of the otherwise same dimensional system are illustrated for an altered domain height (upper right), effective dispersion coefficient (lower left), and density difference (lower right). The corresponding changes of the characteristic scales reveal the differences of the formulations.

- **Case 2:** $D_{\mathrm{eff},2} = 0.5 D_{\mathrm{eff},1}$ (lower left in Fig. 2.4). From Eq. (2.38) we know that the number of initial fingers doubles for this case. Here, this is coherent with the classical formulation, where the more unstable situation implies an increase in the number of initial fingers per unit length. Accordingly, L_c^{H} and T_c^{H} now remain unchanged. In the formulation of SR2010 the characteristic length scale reduces: $L_{c,2}^{\mathrm{CD}} = 0.5 L_{c,1}^{\mathrm{CD}}$. Therefore, in relation to $L_{c,1}^{\mathrm{CD}}$, the lower boundary moves further away and the more unstable situation, again, implies that the system encounters more temporal regimes. $L_{c,2}^{\mathrm{CD}} = 0.5 L_{c,1}^{\mathrm{CD}}$ also reveals that in this scaling the number of fingers per unit length is conserved. Hence, the interpretation of the formulation of SR2010 is more obscure with respect to the spatial characteristics at the onset. (Analog to this case is the variation of porosity ϕ.)

- **Case 3:** $\Delta\rho_{w,2} = 2\Delta\rho_{w,1}$ (lower right in Fig. 2.4). This case is similar to case 2, as the dimensional system exhibits four initial fingers. However, the penetration depth of the fingers at time t is larger, which is due to the increased buoyancy velocity U_c. The fact that the fingers are already closer to the lower boundary is reflected in both dimensionless formulations as $T_{c,2}^{H} = 0.5 T_{c,1}^{H}$ and $T_{c,2}^{\mathrm{CD}} = 0.25 T_{c,1}^{\mathrm{CD}}$. Again, the number of fingers per unit length is preserved for the formulation of SR2010, while prediction of an increase of the total number of initial fingers is coherent with the classical formulation. (Analog to this case is the variation of permeability k.)

Summarizing the considerations above, both dimensionless formulations aid the understanding of the system. While both are consistent within their framework, the classical formulation emphasizes the spatial aspects of the initial density fingers, whereas the formulation of SR2010 gives direct insight into the transient development of the relaxation process.

3 | Numerical Experiments

The numerical experiments form the data basis of the information transfer from the simulation to the laboratory experiment. The assumption is that the physical process is completely and faithfully represented in the generated synthetic data that then can be learned by the CNNs. The trained CNNs are then able to transfer the information to the laboratory experiment, making the inaccessible flow fields accessible. This chapter summarizes the settings used to generate the datasets for training the CNNs and presents the temporal development in the numerical experiments.

For the numerical simulation of the dynamics of active solute transport I used a numerical solver implemented by P. Bastian in Dune [*Blatt and Bastian*, 2007; *Bastian et al.*, 2008a, b; *Blatt and Bastian*, 2008; *Blatt et al.*, 2016], which solves the dimensionless formulated problem according to the scaling by the domain height (cf. Eqs. (2.41) to (2.43)). The solver uses a cell-centered finite volume method to solve the flow problem, a symmetric interior penalty discontinuous Galerkin method to solve the transport problem, and offers several explicit and implicit time stepping schemes. For a broad range of Rayleigh numbers I ran the simulations on a 2-dimensional rectangular structured grid using the Alexander time stepping scheme of order two [*Alexander*, 1977]. The spatial extent of the flow domain was set to $\tilde{x} = 1.0$ and $\tilde{z} = 0.5$ at a corresponding resolution of 768×384 and the temporal resolution for the data output was set to $\mathrm{d}\tilde{t} = 0.02$. To cut down the computation time for the individual simulation runs, each run was computed in parallel on four CPU cores. With this, I conducted three distinct sets of numerical experiments: the first set of experiments (NE1) implements a constant upper concentration boundary condition of $\tilde{C}_w = 1$ (Section 3.1), the second set of experiments (NE2) introduces spatial and temporal variation of the upper concentration boundary condition (Section 3.2), and the third set of experiment (NE3) includes superscale convection by adding driving of the background flow at the lateral boundaries, while resembling the first dataset in the upper concentration boundary condition (Section 3.3). In their entirety, these sets of numerical experiments represent the processes that are expected to occur in the laboratory experiment.

3.1 Constant Concentration Boundary Condition (NE1)

For NE1, the concentration was set to a constant value of $\tilde{C}_w = 1$ at the upper boundary, while all boundaries were set to be impermeable to the fluid flow. Initially, the flow domain exhibited the density $\tilde{C}_w = 0$ and each forward simulation was run for 8,000 time steps. For the individual runs this roughly corresponded to the time, when the fastest density finger reached the bottom boundary. I ran 39 individual simulations in the range of $\mathrm{Ra}_{\mathrm{sim}} \in [2{,}000,\ 27{,}000]$, where the Rayleigh numbers were chosen to be almost equally distributed.

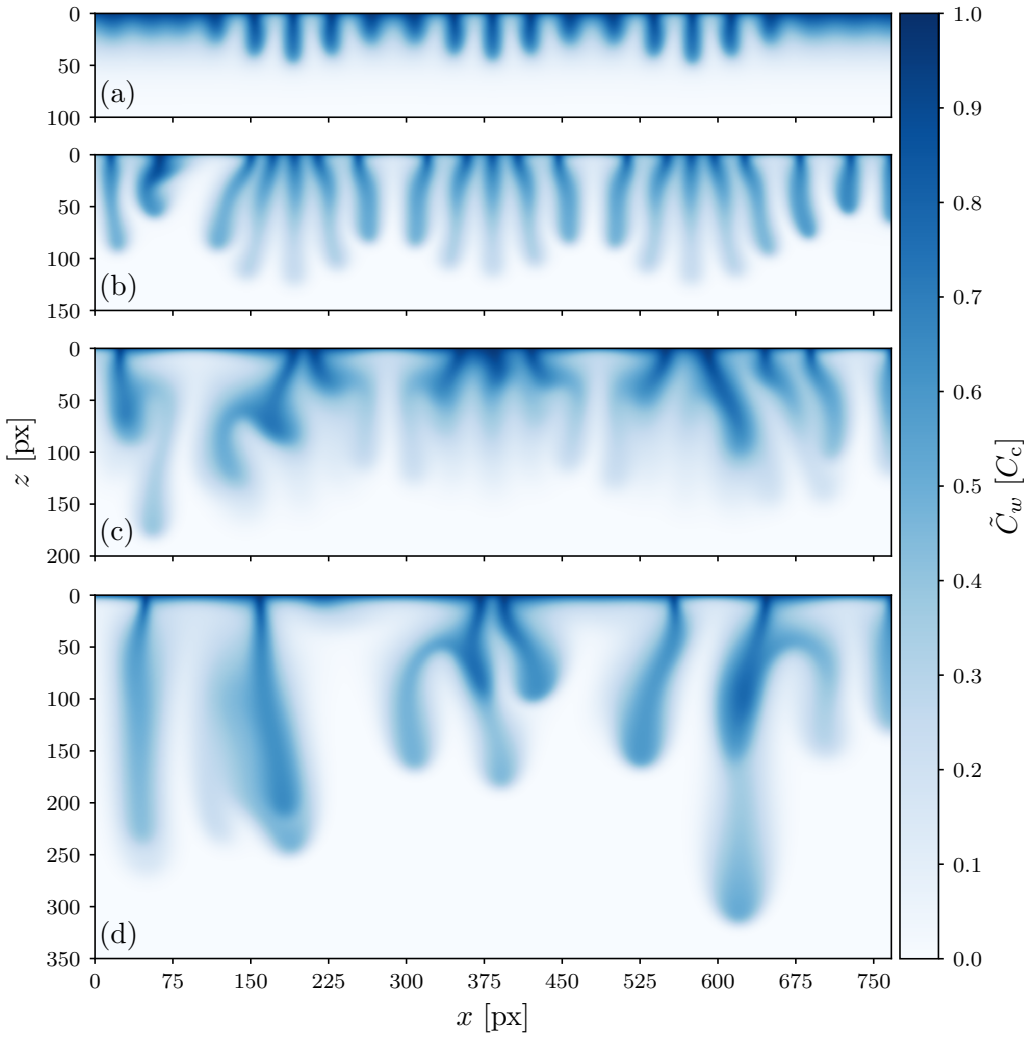

Figure 3.1: Concentration fields of NE1 for $\mathrm{Ra}_{\mathrm{sim}} = 4{,}000$ at $\tilde{t} = 3.1$ (a), $\tilde{t} = 3.8$ (b), $\tilde{t} = 4.8$ (c), and $\tilde{t} = 5.9$ (d). The complete width of the domain is presented, while the shown portion of the depth is adapted to the density fingers.

Examples of the temporal development of the concentration fields are shown in Fig. 3.1 for $\mathrm{Ra}_{\mathrm{sim}} = 4{,}000$ and additionally in Figs. A.1 and A.2 for $\mathrm{Ra}_{\mathrm{sim}} = 12{,}000$ and 26,000. At first, the initial fingers develop, whereby more fingers are observed for higher Rayleigh numbers. Note that the nonuniform initiation of these fingers is due to the parallelized computation on four CPU cores. The domain is divided into smaller subdomains that are computed on the individual cores. This introduces boundary effects at the interfaces of the subdomains that cause earlier initiation. For later times the initial fingers start to interact and merge, leading to dominant plumes that are

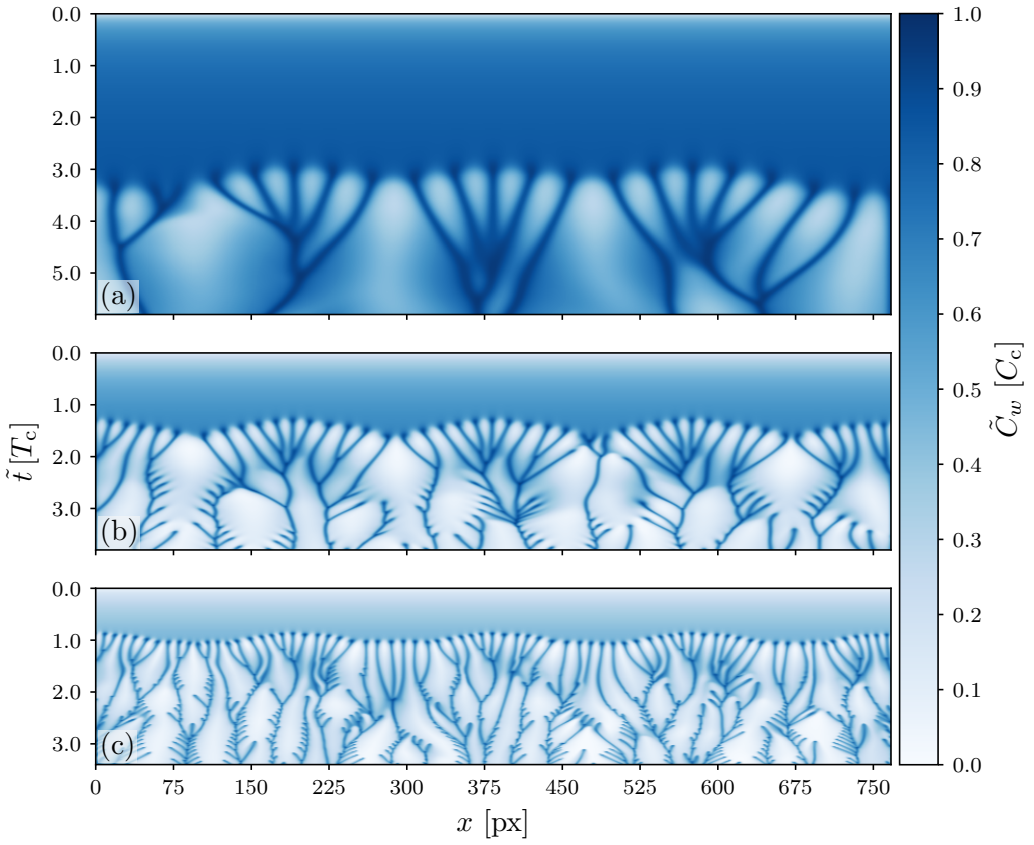

Figure 3.2: Space-time maps of NE1: temporal development of concentration profiles 5 px below the upper boundary for $\mathrm{Ra}_{\mathrm{sim}} = 4{,}000$ (a), $\mathrm{Ra}_{\mathrm{sim}} = 12{,}000$ (b), and $\mathrm{Ra}_{\mathrm{sim}} = 26{,}000$ (c).

now more coarsely spaced. Whereas for $\mathrm{Ra}_{\mathrm{sim}} = 4{,}000$ no secondary density fingers are initiated in between the dominant plumes, they get more and more abundant for higher Rayleigh numbers.

Space-time maps for NE1 for $\mathrm{Ra}_{\mathrm{sim}} = 4{,}000$, 12,000, and 26,000 are shown in Fig. 3.2, where the concentration profile at the fixed position of 5 px below the upper boundary is plotted over the course of the numerical experiments. In the beginning, the concentration is zero, before the solute gets uniformly transported to $z = 5$ px by pure diffusion. For lower Rayleigh numbers the diffusive process is more pronounced and has a longer duration when compared to higher Rayleigh numbers. At the instability onset the initial fingers develop and accumulate the solute from their surrounding. This initiation appears earlier at distinct, regularly spaced positions, which is again due to the parallelized computation on four CPU cores. In this representation, the subsequent lateral movement of the seeding points during the merging of the fingers that leads to a spatial coarsening of the resulting plumes becomes observable. In

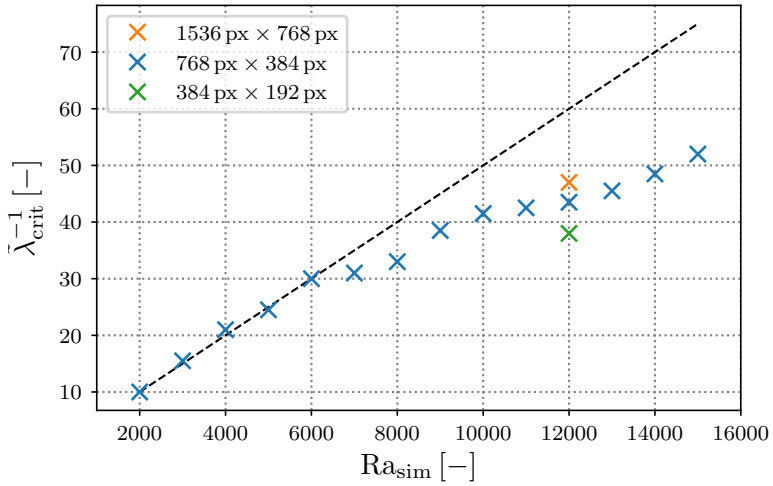

Figure 3.3: Observed relation of the reciprocal critical wavelength and the Rayleigh number of NE1: The black dashed line gives the theoretical proportional relation. At $\mathrm{Ra}_{\mathrm{sim}} = 12{,}000$ simulation runs with higher (orange cross) and lower resolution (green cross) are given to show the influence of numerical dispersion.

the spaces in between, the concentration again increases due to diffusion, before the secondary fingers reinitiate. Subsequently, the secondary fingers are swept into the dominant plumes, providing them with more solute. In addition to the lateral movement of the finger seeding points, the space-time maps reveal the self similar temporal behavior of the transient relaxation process. Going from low to high Rayleigh numbers, the same spatio-temporal structures appear on decreasing spatial and shortening temporal scales.

In Fig. 3.3 the observed relation of the initial finger wavelength $\tilde{\lambda}_{\mathrm{crit}}$ to the Rayleigh number $\mathrm{Ra}_{\mathrm{sim}}$ is compared to the expected proportionality (cf. Eq. (2.45)). The presentation of values for $\mathrm{Ra}_{\mathrm{sim}} > 15{,}000$ is omitted, however, as it does not provide further insight into the relation for the numerical experiment. For low Rayleigh numbers the numerical experiments agree with the expected proportional relation, whereas the values start to divert for $\mathrm{Ra}_{\mathrm{sim}} > 6{,}000$.

Discussion I attribute the observed disagreement to the effect of numerical dispersion that arises as a numerical error when the spatial discretization of the problem is too coarse. As a result, the solute is subjected to stronger spreading than due to the pure physical dispersion. Since the phenomenology of the numerical and physical dispersion is the same, this results in an apparent dispersivity in the simulations, which according to Eq. (2.44), effectively reduces the Rayleigh number. Accordingly, the value of $\tilde{\lambda}_{\mathrm{crit}}^{-1}$ is reduced in this case. To support this, I ran two additional simulations at $\mathrm{Ra}_{\mathrm{sim}} = 12{,}000$ with a decreased resolution of 384×192 and an increased resolution of 1536×768.

As shown in Fig. 3.3, the simulation with a finer discretization alleviates the issue, while the coarser discretization exacerbates it. With this assessment I argue that the physical process is still represented correctly, but in accordance to an adjusted apparent Rayleigh number that is smaller than the chosen simulation parameter: $\mathrm{Ra}_{\mathrm{app}} < \mathrm{Ra}_{\mathrm{sim}}$. This justifies the choice of the resolution of 768×384, which made the generation of the large datasets computationally feasible.

3.2 Modified Concentration Boundary Condition (NE2)

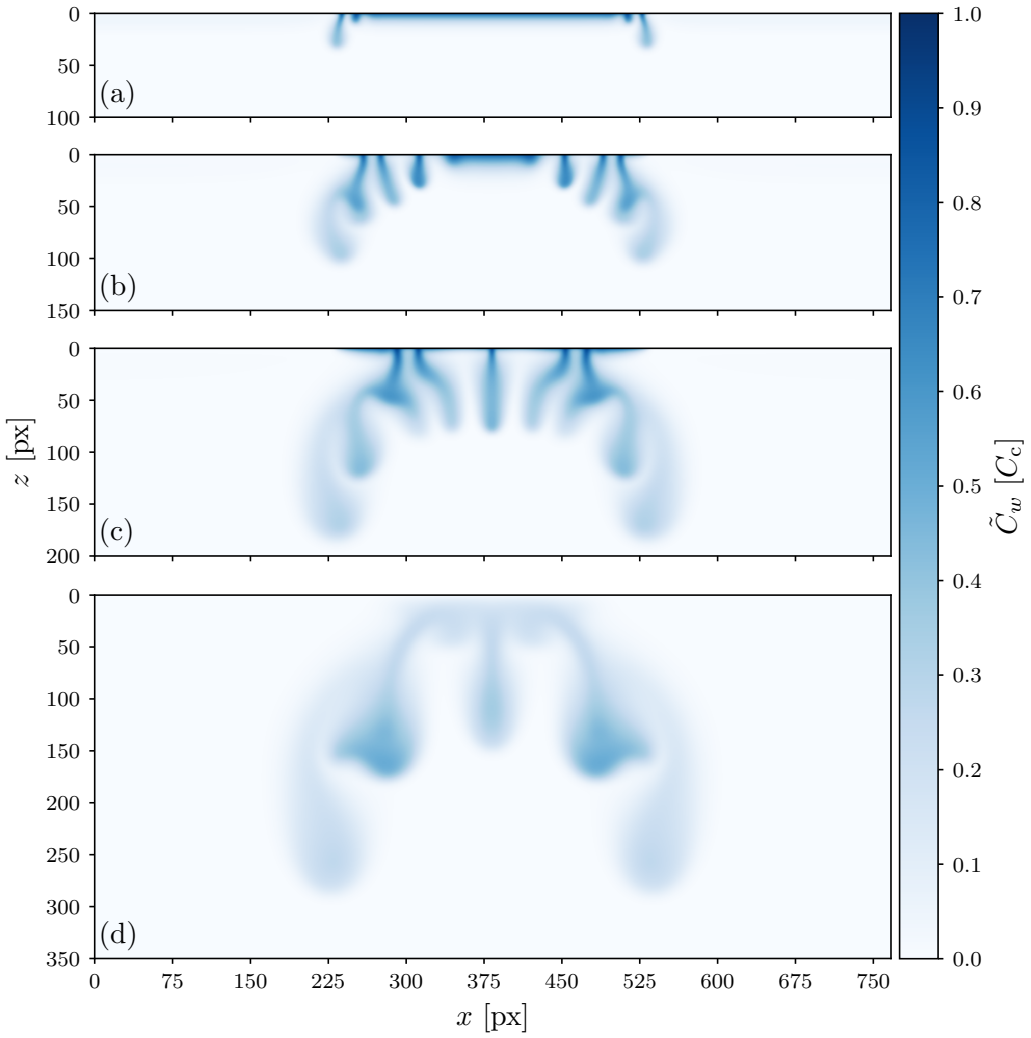

Figure 3.4: Concentration fields of NE2 for $\mathrm{Ra}_{\mathrm{sim}} = 5{,}750$ at $\tilde{t} = 0.3$ (a), $\tilde{t} = 1.0$ (b), $\tilde{t} = 1.8$ (c), and $\tilde{t} = 2.9$ (d). Same representation as in Fig. 3.1.

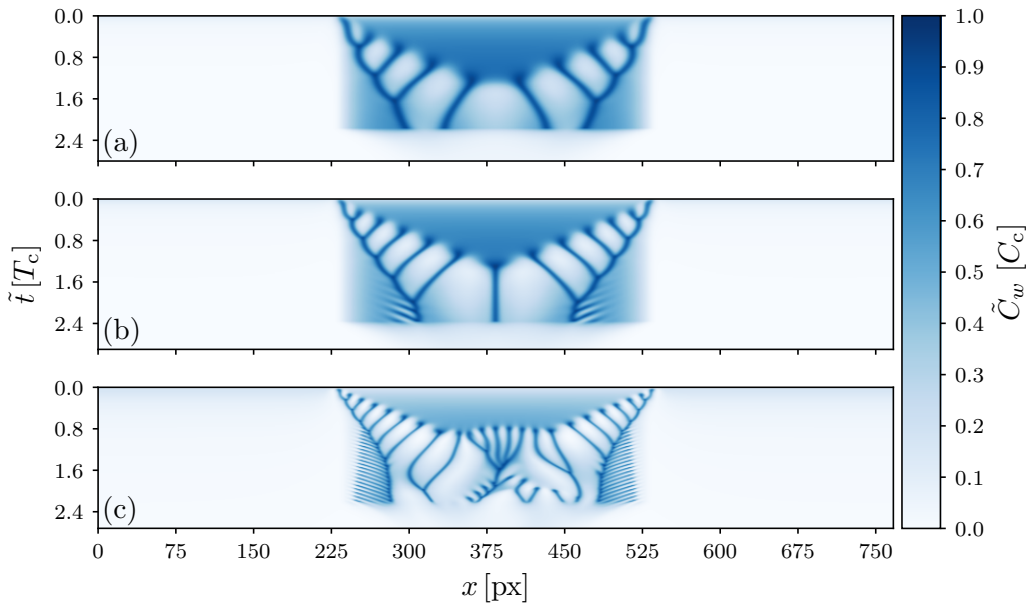

Figure 3.5: Space-time maps of NE2: temporal development of concentration profiles 5 px below the upper boundary for $\mathrm{Ra}_{\mathrm{sim}} = 3{,}750$ (a), $\mathrm{Ra}_{\mathrm{sim}} = 5{,}750$ (b), and $\mathrm{Ra}_{\mathrm{sim}} = 13{,}750$ (c).

In the second set of numerical experiments, NE2, the upper concentration boundary condition was chosen to be spatially nonuniform. The concentration was set to $\tilde{C}_w = 1$ for $230\,\mathrm{px} < x < 538\,\mathrm{px}$ and $\tilde{C}_w = 0$ for the remainder. Additionally, the concentration was set to $\tilde{C}_w = 0$ after $\tilde{t} = \frac{5}{6}T_c$ at the entire upper boundary, introducing a temporal cutoff. The boundary conditions with respect to the flow were set equivalent to NE1. With these settings, I ran 6 individual simulations evenly spaced in the parameter range of $\mathrm{Ra}_{\mathrm{sim}} \in [3{,}750,\ 13{,}750]$, again for 8,000 time steps each.

The temporal development of the concentration fields are shown for the example of $\mathrm{Ra}_{\mathrm{sim}} = 5{,}750$ in Fig. 3.4 and for the examples of $\mathrm{Ra}_{\mathrm{sim}} = 3{,}750$ and 13,750 in Figs. A.3 and A.4, respectively. Again, finer initial fingers are found for larger Rayleigh numbers, the detailed development of the instability is different to NE1, however. Due to the spatially nonuniform boundary condition, fine instabilities are initiated at the positions of the concentration jumps. These first fingers propagate downward as their seeding points laterally move toward the center of the domain. During this lateral movement the first fingers incorporate smaller fingers that were initiated at later times. Only very late independent fingers get initiated at the center of the domain, before the temporal cutoff of the concentration at the top slows down the development of the system.

The same behavior is observed in the space-time maps for NE2. In Fig. 3.5 examples are depicted for $\mathrm{Ra}_{\mathrm{sim}} = 3{,}750$, 5,750, and 13,750. The first fingers initiate right away, due to the introduced instabilities at the concentration jumps, and their seeding points

laterally move to the center. For low $\mathrm{Ra}_{\mathrm{sim}}$, it appears that almost all initial fingers are influenced by the first instabilities due to the nonuniform boundary condition. For larger values, more and more independent fingers form in the center and initiation patterns similar to the observations for NE1 emerge. After the temporal cutoff of $\tilde{C}_w$ at the top, the remaining solute is transported downward and the concentration drops to zero at the observation position of $z = 5\,\mathrm{px}$.

3.3 Representation of Superscale Convection (NE3)

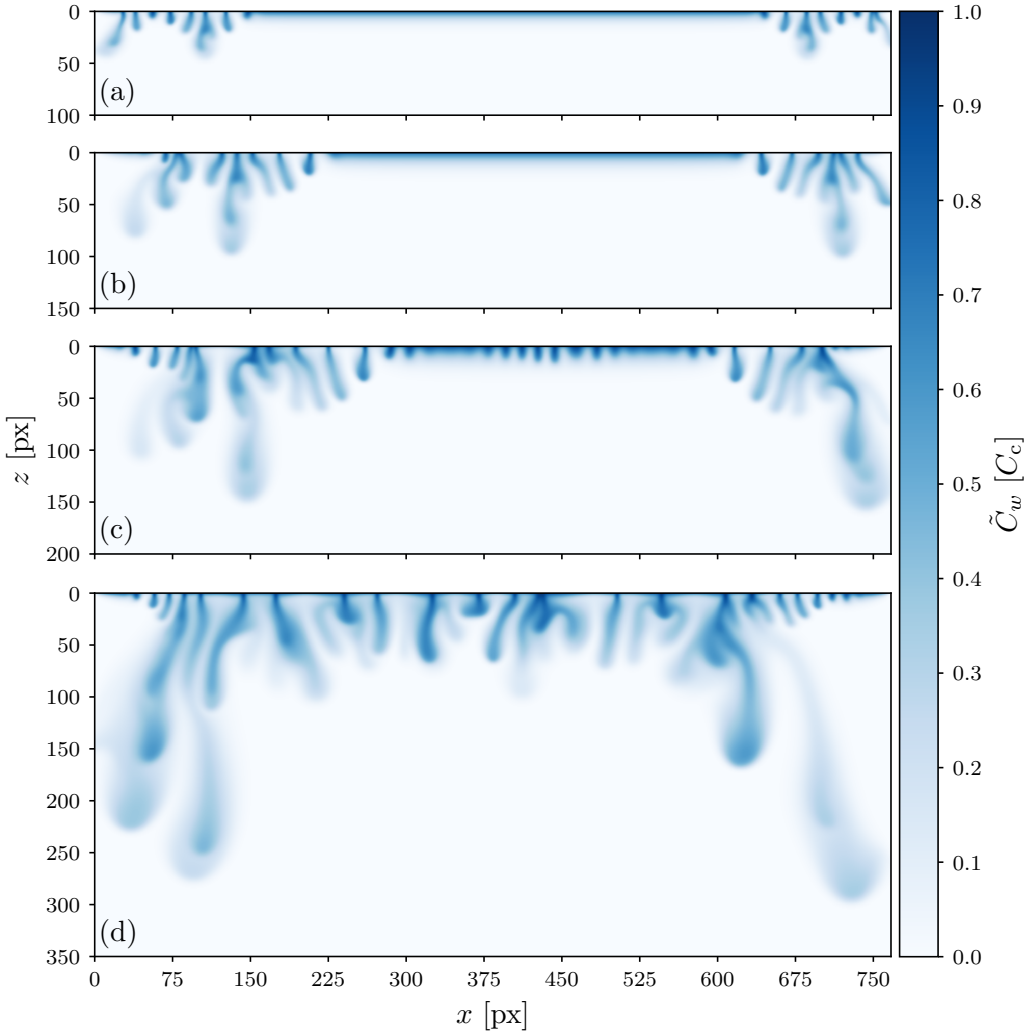

Figure 3.6: Concentration fields of NE3 for $\mathrm{Ra}_{\mathrm{sim}} = 9{,}000$ at $\tilde{t} = 0.4$ (a), $\tilde{t} = 0.8$ (b), $\tilde{t} = 1.2$ (c), and $\tilde{t} = 2.2$ (d). Same representation as in Fig. 3.1.

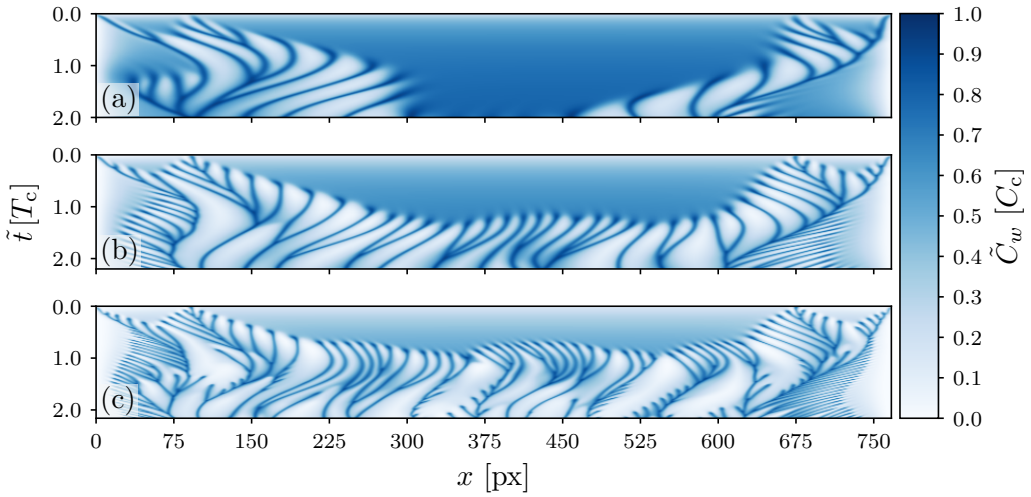

Figure 3.7: Space-time maps of NE3: temporal development of concentration profiles 5 px below the upper boundary for $\mathrm{Ra}_{\mathrm{sim}} = 4{,}500$ (a), $\mathrm{Ra}_{\mathrm{sim}} = 9{,}000$ (b), and $\mathrm{Ra}_{\mathrm{sim}} = 16{,}000$ (c).

NE1 and NE2 exclusively accounted for the transport processes of the density-driven instability itself. For the third set of numerical experiments, NE3, I additionally introduced superscale convection represented as lateral background flow that was superposed to the development of the instability. The upper and lower boundaries were still set to be impermeable. At the lateral positions I implemented Dirichlet boundary conditions, where the chosen potential led to a lateral gradient, driving the background convection in the domain. The right boundary was set to $\tilde{p} = 0$ throughout the complete simulation runs. On the left boundary the pressure was variated with time:

$$\tilde{p}(\tilde{t}, \tilde{x} = 0) = 0.15 \sin\left(2\pi \frac{\tilde{t}}{\tilde{t}_{\mathrm{max}}}\right), \tag{3.1}$$

where $\tilde{t}_{\mathrm{max}}$ is the dimensionless time that corresponds to the maximum time step of the simulation run. Accordingly, the pressure gradient slowly increased, driving the background fluid flow in the positive x-direction, before it decreased and inverted to drive the flow in the negative x-direction. The concentration boundary condition was set equivalent to NE1, with $\tilde{C}_w = 1$ at $\tilde{z} = 0$. With these settings, I ran 16 simulations over the course of 8,000 time steps in the range of $\mathrm{Ra}_{\mathrm{sim}} \in [1{,}500,\ 16{,}000]$, again almost evenly distributed over the parameter range.

For the example of $\mathrm{Ra}_{\mathrm{sim}} = 9{,}000$ the temporal development is shown in Fig. 3.6 and additional examples are given in Figs. A.5 and A.6 for $\mathrm{Ra}_{\mathrm{sim}} = 4{,}500$ and 16,000. The onset of the instability in NE3 combines effects of NE1 and NE2. Similar to NE2 the instabilities are already initiated at lateral positions, while in the center of the domain the transport is still governed by diffusion. As the early initiation regions are

further apart than in NE2, more regularly spaced initial fingers can develop in the center, which is similar to the observations in NE1. Apart from these similarities to NE1 and NE2, a qualitative difference of the transport is observed for NE3. Due to the background flow, complete fingers and large plumes are convected in lateral direction. For NE1 and NE2 lateral movement is only observed for smaller fingers and the finger seeding points, while larger fingers and plumes stay at their lateral position.

In the space-time maps for NE3 the influence of the background flow on the positions of the seeding points becomes clearly visible. Examples for $\mathrm{Ra}_{\mathrm{sim}} = 4{,}500$, 9,000, and 16,000 are given in Fig. 3.7. According to the sinusoidal boundary condition on the pressure, the seeding points shift to the right for the first half of the simulation runs, then shift to the left for the second half. This lateral movement is superposed to the lateral movement due the density-driven instability as already observed for NE1 and NE2.

4 | Laboratory Experiment

In parts based on *Kreyenberg et al.* [2019].

The real world data that represent the target of the information transfer are provided by a laboratory experiment to observe active solute transport in a Hele-Shaw cell. These kind of experiments allow for high resolution measurements of the concentration fields and their temporal development. However, it is impossible to experimentally infer the corresponding flow fields. This is obvious for regions where the solute distribution is homogeneous and, therefore, no information about the flow is given. In regions where the temporal development of the solute concentration provides information about the movement, it is still impossible to disentangle the solute transport due to dispersion from convective transport without modeling the underlying processes numerically, which implicates the estimation of the system parameters and is a challenge of its own. Since the process is slow and sensitive to small density differences, typical methods to measure fluid flow, e.g. particle image velocimetry, are generally inapplicable.

The data used for the analysis in this work were originally recorded in the course of the experimental study of *Kreyenberg* [2015], where a detailed description of the measurement technique is given. This chapter summarizes the laboratory experiment to support the understanding of the data. First, the experimental setup is presented, before the temporal development is discussed to allow for a comparison to the numerical experiments.

4.1 Experimental Setup

The experimental setup (Figure 4.1) is composed of a Hele-Shaw cell, an LED light source, an optical band-pass filter, and a CCD camera to observe the solute concentration distribution by employing light transmission measurements.

The Hele-Shaw cell is made from two 8 mm thick glass plates with the dimensions of 500×300 mm. Spacers of 0.4 mm thickness and sealing material along the left, right, and lower edges hold the glass plates apart to create a narrow gap d that serves as the quasi 2-dimensional flow domain. In this narrow gap, the fluid flow is described by the same governing equations as for porous media (cf. Section 2.1.3) with porosity $\phi = 1$. It was ensured that the gap width satisfies $d < 10(\phi D/U_c)$, as deduced by *Slim et al.* [2013] from the results of *Fernandez et al.* [2002], in order to reduce the influence of possible 3-dimensional effects.

Three inlet ports at the lower edge were connected to an external fluid reservoir to keep the fluid level stable during the course of the experiment. The fluid used in the experiment is deionized water with *Brilliant Blue FCF* (BB) as a tracer and solute

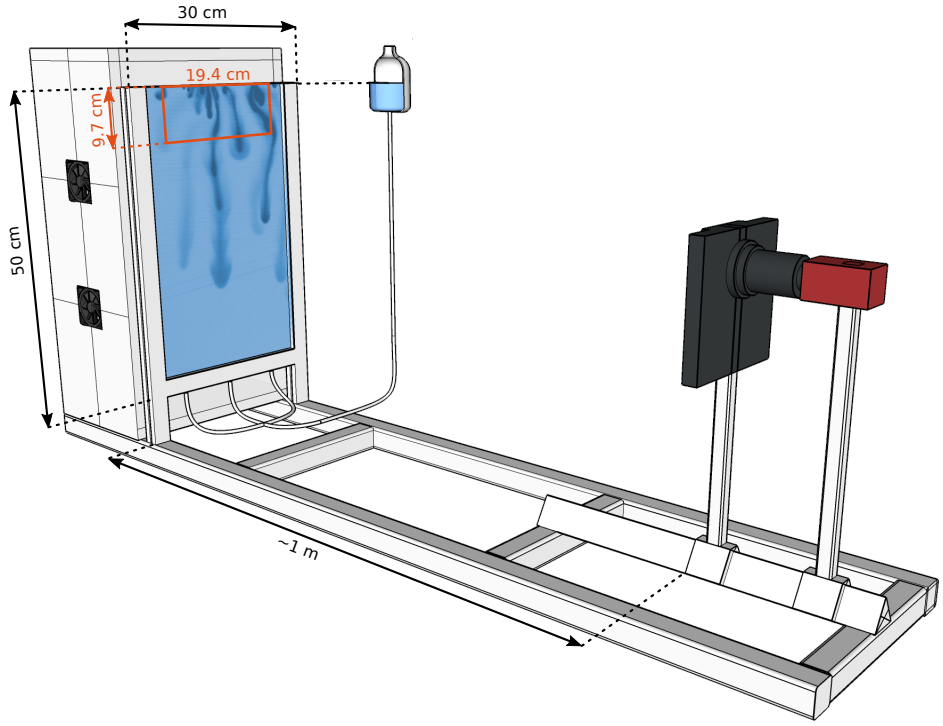

Figure 4.1: Illustration of the experimental setup: The Hele-Shaw cell (50×30 cm) is placed in front of the LED light source. During the evaporation induced density-driven instability experiments, the external fluid reservoir keeps the fluid level inside the Hele-Shaw cell stable. The CCD camera captures the transmitted light. The red rectangle highlights the chosen observation area used for the flow field estimation.

simultaneously that allows to observe the dynamics as well as to introduce density contrasts depending on the BB concentration. Based on the Beer-Lambert law the absorption coefficient of BB was determined using the measured light extinction in a spectrometer.

The light source consists of an LED-array and a translucent acrylic glass plate as optical diffuser up front to create a homogeneous illumination of the Hele-Shaw cell. The LEDs match the maximum light absorption wavelength of BB (630 nm) in their maximum emission wavelength (625 nm). The optical band-pass filter in front of the CCD camera (AVT Pike F505B, 2452×2054 px) was chosen accordingly with a transmission band of 632 ± 11 nm.

To determine the spatially resolved gap width of the Hele-Shaw cell, calibration measurements were performed using two uniform BB concentration solutions. By measuring the light intensity with the CCD camera the width can be inferred by the

Table 4.1: Characterization of the laboratory experiment.

Parameter	Value	Unit
Height of the Hele-Shaw cell H	490 ± 5	mm
Width of the Hele-Shaw cell W	270 ± 5	mm
Height of observation area H_{obs}	97.2 ± 1.1	mm
Width of observation area W_{obs}	194.4 ± 1.1	mm
Temporal resolution Δt	600	s
Mean gap width at the upper edge[a] $\bar{d}_{\mathrm{top}}$	0.61 ± 0.04	mm
Mean gap width in observation area $\bar{d}_{\mathrm{obs}}$	0.72 ± 0.05	mm
Permeability[a,b] k	$[3.1 \pm 0.4] \times 10^{-8}$	m^2
Base concentration C_0	86.5 ± 0.5	$\mathrm{kg/m^3}$
Maximum concentration C_{max}	154.4 ± 11.9	$\mathrm{kg/m^3}$
Solutal expansion coefficient of BB β_{s}	$[1.90 \pm 0.15] \times 10^{-4}$	$\mathrm{kg\,m^3/(m^3\,kg)}$
Density difference[c] $\Delta\rho$	$[12.9 \pm 2.5] \times 10^{-3}$	$\mathrm{kg/m^3}$
Dynamic viscosity of water[d] μ_w	1.002×10^{-3}	$\mathrm{Pa\,s}$
Porosity ϕ	1	$\mathrm{m^3/m^3}$
Diffusion coefficient of BB D_m	$[2.9 \pm 1.0] \times 10^{-10}$	$\mathrm{m^2/s}$
Rayleigh number[a] $\mathrm{Ra}_{\mathrm{lab}}$	$6{,}600 \pm 2{,}700$	–

[a] Relevant value for the instability onset.
[b] Results from $k = \bar{d}_{\mathrm{top}}^2/12$.
[c] Results from $\Delta\rho = \beta_{\mathrm{s}}(C_{\mathrm{max}} - C_0)$.
[d] From *Lide* [2004] for 20°C.

Beer-Lambert law. This calibration for the gap width allowed for the spatially resolved determination of the BB concentrations during the experiment, where an image was recorded every 10 minutes.

For the density-driven instability experiment, the flow domain was initially completely filled with uniform BB concentration solution ($C_0 = 86.5\,\mathrm{kg/m^3}$). Through water evaporation at the upper unsealed edge of the Hele-Shaw cell, BB accumulated locally up to $C_{\mathrm{max}} = 154.4\,\mathrm{kg/m^3}$. This set the upper boundary condition for the experiment. Note that this temporal build up qualitatively differs from the numerical experiments, where the upper boundary is set to the maximum concentration at the initial time. Therefore, the initial times $t_{\mathrm{lab}} = 0\,\mathrm{h}$ and $\tilde{t}_{\mathrm{sim}} = 0$ have different meaning with respect to the temporal development. The fluid loss due to evaporation was compensated by inflow of BB solution with C_0 through the three inlet ports at the bottom, which increased the total BB amount in the cell over time. The increased BB concentration at the top altered the density ($\Delta\rho = [12.9 \pm 2.8] \times 10^{-3}\,\mathrm{kg/m^3}$) of the solution there, leading to an unstable layering within the fluid.

The above and additional parameters that characterize the laboratory experiment are summarized in Table 4.1. The given values are determined experimentally with

their according uncertainty from propagating the errors, except for the diffusion coefficient of BB, where only an estimation based on the size of the dye molecules and their assumed movement in water is available. Accordingly, the estimated Rayleigh number of $\mathrm{Ra}_{\mathrm{lab}} = 6{,}600 \pm 2{,}700$ incorporates this uncertainty, where the values relevant to the onset of the instability have been taken into account.

4.2 Temporal Development

After the evaporation at the upper boundary of the Hele-Shaw cell induced the unstable fluid layering due to the increased BB concentration there, the density-driven instability developed in its full beauty (video of the observation available at `https://doi.org/10.11588/data/7NEEKF`). The temporal development of the concentration fields is depicted in Fig. 4.2 for the observation area in the upper center of the cell as indicated in Fig. 4.1. The observation area is 19.4 cm wide and is chosen to exclude regions at the left and right side of the cell, where the lateral boundaries are expected to affect the flow. The depth of the observation area is set to 9.7 cm, since I focus on the early and intermediate temporal regimes in accordance to the numerical experiments. In general, the observed concentration fields qualitatively agree with the concentration fields of the numerical experiments, while substantial noise in the concentration values is observed due to the measurement technique. At early times the initial fingers form, which are almost regularly spaced. Their individual initiation time is less uniform, however, with fingers that already have developed a length of about 10 mm being right next to fingers that are barely initiated. I attribute this to irregularities in the experimental setup. Due to the capillarity and imperfections on the surface of the glass plates the water interface at the upper edge was not perfectly flat. Also, the gap in the center of the cell was 0.2 mm wider than the gap at the sides. Both effects can cause nonuniform evaporation, which influences the initiation and the regularity of the initial fingers. For later times the initial fingers start to merge, forming the large dominant plumes. In the laboratory experiment these plumes are subjected to lateral movement, which is most prominent for the outermost ones that show the fastest development and get deflected toward the center of the cell. Qualitatively, these effects are similar to the observations of NE2 and NE3 with the nonuniform concentration boundary condition and the introduced background flow, but are much stronger in the laboratory experiment. This comparison supports that there are influences of a nonuniform concentration boundary condition and that the flow is affected by the background flow in the Hele-Shaw cell that balances the evaporation at the upper edge.

The space-time map of the laboratory experiment as depicted in Fig. 4.3 confirms these observations. The seeding points of the initial fingers move closer together as they merge to form larger plumes. The outermost seeding points move toward the center, incorporating the other fingers on the way, which is similar to the observations in NE2. Superposed to that is a general drift to the right at early times, similar to the observation in NE3, that vanishes for later times. Until $t = 19.83$ h, when the

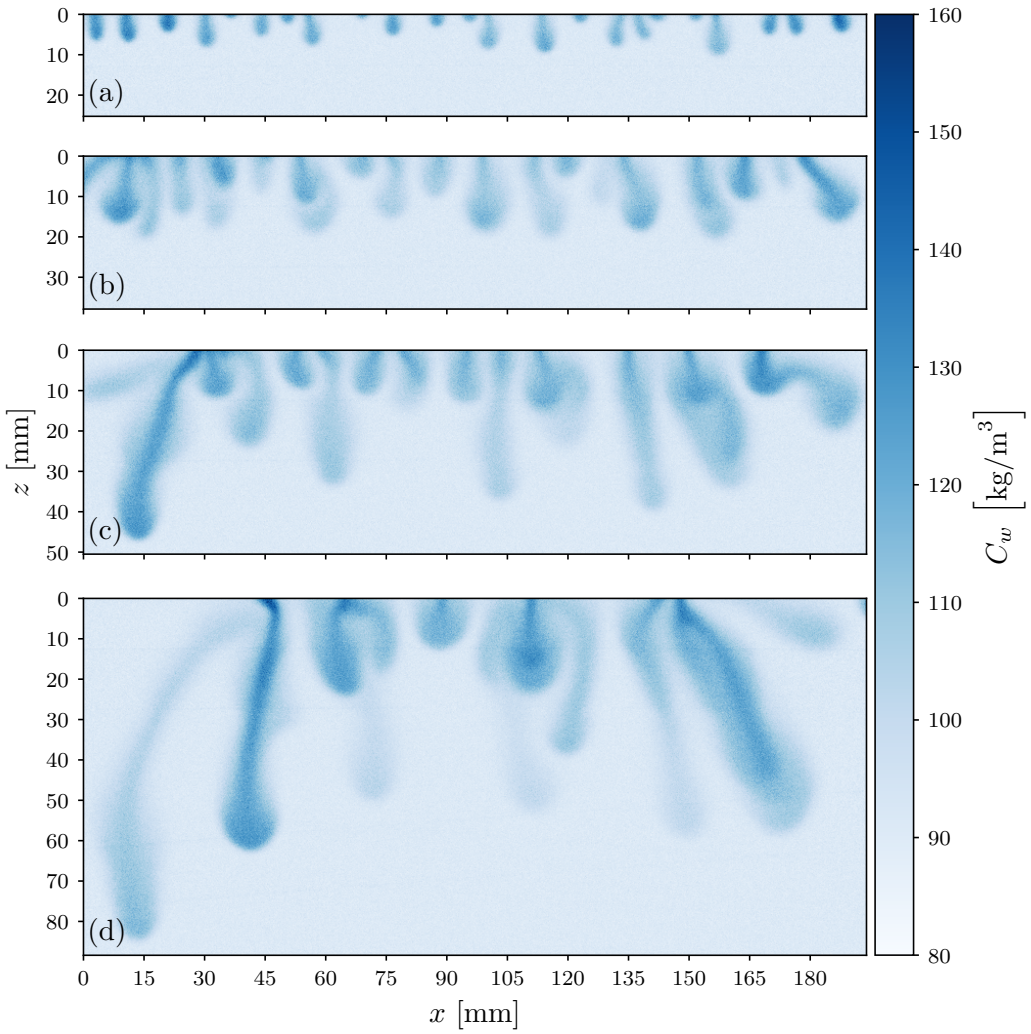

Figure 4.2: Concentration fields of the laboratory experiment for $\mathrm{Ra}_{\mathrm{lab}} \sim 6{,}600$ at $t = 2.33\,\mathrm{h}$ (a), $t = 4.17\,\mathrm{h}$ (b), $t = 6.50\,\mathrm{h}$ (c), and $t = 9.83\,\mathrm{h}$ (d).

fastest plume reaches the bottom of the observation area, the regimes encountered in the temporal development are similar to the ones encountered for $\mathrm{Ra} = 4{,}000$ in NE1, since no initiation of secondary fingers is apparent yet. For later times the initiation patterns on the left are congruent with the ones observed at the outermost plumes for NE2 and NE3 at large Rayleigh numbers. On the right the pattern is quite different, however, as there fewer finger seeding point move toward the dominant plume.

Besides the qualitative agreement between the laboratory experiment and the numerical experiments after the onset of the instability, a major difference becomes evident in Figs. 4.2 and 4.3. In the laboratory experiment, the temporal regime, in which the transport is purely diffusive, is not observable. This is due to the construction

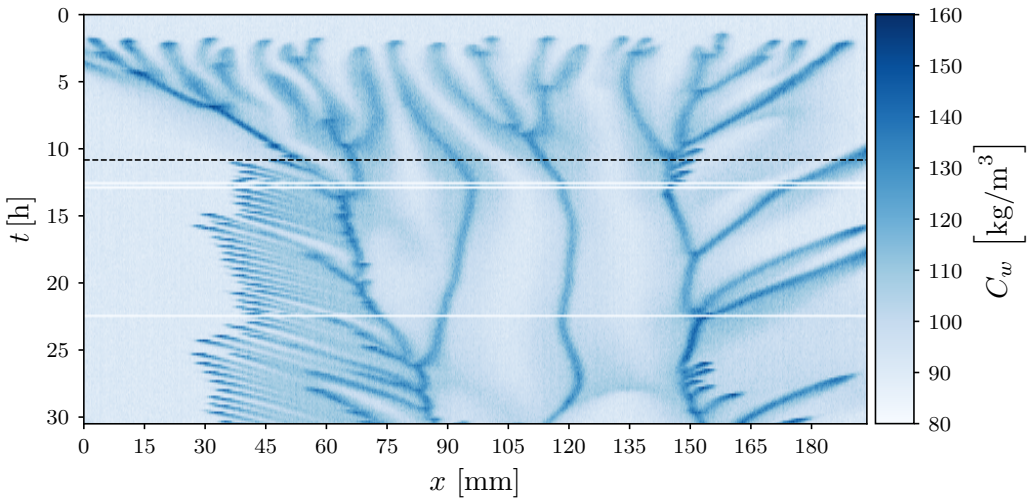

Figure 4.3: Space-time map of the laboratory experiment: Temporal development of the concentration profile 2.6 mm below the upper boundary for $\mathrm{Ra}_{\mathrm{lab}} \sim 6{,}600$. Blank data are due to missing images in the observation at $t = 12.67$, 13.00, and 22.50 h. The dashed black line indicates the time $t = 19.83$ h, when the fastest plume reaches the bottom of the observation area as marked by the red rectangle in Fig. 4.1.

of the Hele-Shaw cell. The two glass plates have chamfered edges, which introduced refraction of the light in the small region at the top, where pure diffusion is expected to be apparent. Therefore, this small region was optically inaccessible.

The critical wavelength λ_{crit} for the laboratory experiment can be determined from the spatial distribution of the initial fingers. The concentration profile 1.3 mm below the upper boundary for $t = 2.0$ h is depicted in Fig. 4.4 (black dashed line) with its subsequent development (blue shades). To cope with the measurement noise and allow for a clean signal a standard median image filter with a kernel size of 5 px, which corresponds to 1.26 mm, was applied to the concentration field before taking the profile. Again, the temporally nonuniform initiation due to the expected irregularities in the experiment becomes apparent. While 24 peaks can be distinguished over the width of the observation area, 4 positions with a very small increase in the concentration can be identified in between. In the subsequent development these small increases in concentrations vanish as they are swept into the adjacent peaks. Therefore, it is ambiguous whether initial fingers might have developed at these positions in case of a more uniform temporal initiation. Representing this uncertainty, the critical wavelength for the laboratory experiment is determined to be $\lambda_{\mathrm{crit}} = 8.1 \pm 1.7$ mm.

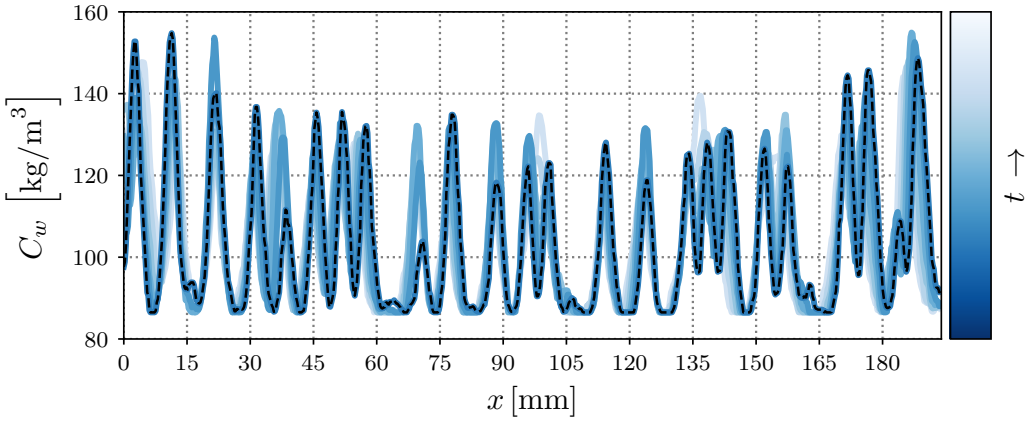

Figure 4.4: Spatial distribution of the initial fingers of the laboratory experiment at $\mathrm{Ra}_{\mathrm{lab}} \sim 6{,}600$: The dashed black line represents the concentration profile 1.3 mm below the upper boundary at $t = 2.0$ h. The subsequent development is indicated in blue shades.

Scaling the critical wavelength with the domain height $H = 490 \pm 5$ mm results in the dimensionless value of $\tilde{\lambda}_{\mathrm{crit}} = [1.65 \pm 0.36] \times 10^{-2}$. Invoking the linear relation between the Rayleigh number and the reciprocal dimensionless critical wavelength (cf. Eq. (2.45)) and using the proportionality as reported by *Riaz et al.* [2006] (cf. Eq. (2.38)), the observed wavelength corresponds to a Rayleigh number of $\mathrm{Ra} = 5{,}400 \pm 1{,}200$. Within the error bounds this agrees with the estimated value from the experimental parameters $\mathrm{Ra}_{\mathrm{lab}} = 6{,}600 \pm 2{,}700$, however, the uncertainties for both values are large. Unfortunately, these estimates reveal a discrepancy toward the numerical experiments. Using the same approach by invoking the linear relation between Ra and $\tilde{\lambda}_{\mathrm{crit}}^{-1}$ and using the linearity as predicted in NE1 (cf. Fig. 3.3) results in $\mathrm{Ra}_{\mathrm{sim}} = 12{,}100 \pm 2{,}600$ for the measured value of $\tilde{\lambda}_{\mathrm{crit}} = [1.65 \pm 0.36] \times 10^{-2}$.

Discussion Although the phenomenology of the density-driven instability observed in the numerical experiments is consistent with the observations of the laboratory experiment, a discrepancy is encountered in the quantification of the Rayleigh number. The values obtained from the experimental parameters and the critical wavelength according to the proportionality as reported by *Riaz et al.* [2006] do not agree with the value predicted from the numerical simulations. Since there are large uncertainties in the estimation of the diffusion coefficient for the laboratory experiment, there is no robust insight at this point to resolve this discrepancy between the experimental value and the value motivated from linear stability analysis, on the one side, and the value from the numerical prediction, on the other side. However, the disagreement does not affect the applicability of the flow field estimation based on the numerical experiments. As long as the correct phenomenology is contained in the data, the correct relation

between the concentration fields and the corresponding flow fields is represented. This is due to the self-similarity of the density-driven instability, where the same spatio-temporal patterns emerge for different Rayleigh numbers, but on smaller spatial and shorter time scales for increasing values of Ra.

5 Convolutional Neural Networks

Deep learning in general and convolutional neural networks (CNNs) in particular are powerful methods to extract complicated features contained in data. CNNs are designed to extract the features of multi dimensional data with a given spatial structure, such as image data, and can learn to associate these features to a desired output. In this chapter the deep learning methods employed in this work are introduced. First, an overview of the concepts important to CNNs is given in Section 5.1. Sections 5.2 and 5.3 provide deeper insight into the background of the concepts and are based on *Goodfellow et al.* [2016], where a comprehensive description of deep learning is given. The specific network architectures used for the flow field estimation in this work are described in detail in Section 5.4.

5.1 Conceptual Outline

The majority of deep learning methods can be described by combining the same basic concepts. A deep learning model (Section 5.2) is used to perform a desired task on the basis of provided input data. Instances of such tasks are image classification, where labels describing objects in an image are predicted, or more general regression tasks, such as optical flow estimation that predicts the apparent movement of objects in subsequent images. Instead of specifically designing the model to perform the task, which can be intractable, a multitude of adaptable parameters that are linked using simple linear and nonlinear mathematical operations are incorporated in the model and arranged in a network structure. The aim is to achieve generic approximation capability for very complicated mappings between the input data and the desired output that is specified by the task. Subsequently, the model is trained on input-output example data to learn the task by adapting the internal parameters (Section 5.3).

Besides restricted Boltzmann machines, autoencoders, and sparse coding that are not further discussed in this work, methods based on CNNs are among the most popular deep learning models [*Guo*, 2017]. Typically, they are utilized to extract features from multidimensional data with a grid like topology, such as image data, but find their application also in analyzing sequential data, like audio data. Their architecture is organized in stacked layers that are connected by convolution filters (cf. Section 5.2.1). Since convolution itself is linear, the chaining of several convolutions is still linear. Therefore, activation functions (cf. Section 5.2.2) following the convolution layers introduce nonlinearities to the model to generate an universal approximator. Indeed, it has been shown that CNNs can represent any continuous function, given

that they are deep enough [*Petersen and Voigtlaender*, 2018; *Yarotsky*, 2018; *Zhou*, 2018]. Here, the term depth refers to the number of layers.

Given the hierarchically layered structure, CNNs are able to efficiently extract hierarchical features in the input data [*Zeiler and Fergus*, 2014]. Low-level features like edges combine to wrinkles, as an example, and more and more abstract concepts in subsequent layers. Reducing the resolution of the layers with increasing depth of the network by introducing pooling or strided convolutions (cf. Section 5.2.3) improves the computational efficiency and condenses the information into an abstract latent representation [*Goodfellow et al.*, 2016]. Often it is useful to reconstruct the output at a higher spatial resolution as the latent representation [*Ronneberger et al.*, 2015]. This can be achieved by appropriate upsampling, which can even be incorporated into the deep learning algorithm by the introduction of transposed convolution layers (cf. Section 5.2.4). This divides the architecture in a contracting part that encodes the information and an expanding part that reconstructs or decodes the information. Hence, these specific architectures are referred to as encoder-decoder CNNs. For an example of such an architecture see Fig. 5.6.

During training, the internal model parameters are adapted based on a training dataset. For supervised learning, the training dataset contains a large number of examples representing the task by associated input-output pairs, such as labeled images for image classification. The model processes the input data examples to predict the output in a forward pass. A loss function (Section 5.3.1) quantifies the discrepancy between the model prediction and the corresponding desired output. This information can then be utilized to adapt the model parameters in a backward pass using backpropagation (Section 5.3.2) in order to decrease the loss in the next forward pass. This procedure is repeated iteratively using an according optimization method (Section 5.3.3).

Despite the expressive power of CNNs, there is no assurance that they actually learn to represent the desired function during the training process. This is because nearly all deep learning problems are ill-posed with the degree of freedom in the model parameters surpassing the available training data examples [*Hinton et al.*, 2012]. Therefore, CNNs are prone to overfitting, which means that they may learn to represent the training data very well, but fail to generalize to any data beyond that. This is a major concern when using deep learning methods [*Goodfellow et al.*, 2016] that needs to be addressed by thoroughly selecting the optimization method and incorporating regularization of the model parameters (Section 5.3.1).

Besides the internal model parameters that are adapted during training, the network is defined by multiple hyperparameters. These hyperparameters include several design choices, such as number of layers, filter number and size, and choice of activation function, additional to the hyperparameters involved in the optimization process, such as choice of the regularization and the learning rate. In some cases there is a theoretical or empirical basis proposing standard choices for these hyperparameters, whereas in other cases only rough guidance exists [*Bengio*, 2012]. Therefore, it often is tedious to

find a set of good hyperparameters, enabling the model to perform well [*Goodfellow et al.*, 2016].

Under this premise, the training process needs to be monitored and the generalization of the trained model needs to be evaluated. Additional to the training dataset, this requires a distinct validation dataset and a test dataset. The validation dataset is a fixed set of data examples that typically is randomly drawn from the original training dataset. These data examples are then excluded from the dataset which is used to adapt the model parameters during training. The loss, which indicates the agreement between the model prediction and the truth, calculated on the validation dataset represents the validation error and is a measure of the performance of the model on the desired task. This information may be used to improve the model by tuning the hyperparameters. The test dataset is used to test the generalization of the fully trained network. To test for a broad generalization, the test dataset may include structural differences to the training and validation datasets. The measure to quantify the generalization is the loss calculated on the test dataset representing the test error. To allow for an unbiased evaluation of the generalization, it is crucial that the test dataset has not been utilized in the training process to adapt the model parameters, nor to adapt the model hyperparameters.

Empirically, it has been demonstrated that CNNs perform remarkably well for various computer vision tasks, while greater depth seems to improve the generalization [e.g., *Bengio et al.*, 2007; *Bengio*, 2009; *Cireşan et al.*, 2012; *Farabet et al.*, 2012; *Krizhevsky et al.*, 2012; *Sermanet et al.*, 2013; *Simonyan and Zisserman*, 2014a; *Dosovitskiy et al.*, 2015; *Szegedy et al.*, 2015; *Ilg et al.*, 2017]. However, there is a lack in the understanding of how they internally work rendering the methods into 'black boxes', which is unsatisfactory from a scientific point of view. Therefore, research is concerned with supporting the empirical findings with a theoretical understanding. Providing a basis that CNNs actually can work, only recently it has been shown that the discrepancy between the validation and the test error is bounded for CNN architectures [*Zhou and Feng*, 2018]. Additionally, techniques to visualize intermediate feature maps provide insight into how CNNs extract the hierarchical features [*Simonyan et al.*, 2013; *Zeiler and Fergus*, 2014; *Nguyen et al.*, 2017; *Olah et al.*, 2018], but are primarily explored with a focus on image classification only.

To employ the methods, there are several deep learning software frameworks, e.g., Caffe [*Jia et al.*, 2014], PyTorch [*Paszke et al.*, 2017], and TensorFlow [*Abadi et al.*, 2016]. They provide the implementation of the building blocks needed to design and train a deep CNN model and therefore make the usage of deep learning methods readily applicable. Typically, the implementation is highly optimized to cut the computational cost and to be able to train and deploy the models on suited hardware, like graphics processing units (GPUs) or tensor processing units (TPUs). Because of these optimizations, the implementation often deviates from the formulation of the concepts as introduced in the following, the operations they represent is equivalent however.

5.2 Model Components

CNNs can be thought of as modular models composed of a variety of components that are arranged in layers, where the output of each layer represents the input of the subsequent layer. This section introduced the most common components used in modern CNNs. With the CNNs being applied to gray scale image data in this work, I predominantly motivate the concepts with this data structure in mind. Nevertheless, the same concepts can be extended toward differently structured data.

5.2.1 Convolution Layers

CNNs make use of a specific mathematical operation in their computational graph: the convolution operation. Mathematically the convolution s of two real-valued functions x and ω is given by

$$s(t) = \int_{-\infty}^{\infty} x(\tau)\omega(t-\tau)d\tau = [x * \omega](t), \tag{5.1}$$

where the convolution operation is denoted by the asterisk. Although suggested by the symbol t, convolution is not restricted to the temporal domain. Especially in deep learning the convolution is often calculated in the spatial domain. There we refer to x as the input, to ω as the kernel, and to s as the feature map. With the assumption that x and ω are only defined on discretized t, the convolution can be defined as

$$s(t) = \sum_{\tau=-\infty}^{\infty} x(\tau)\omega(t-\tau) = [x * \omega](t). \tag{5.2}$$

To make this useful for the framework of deep learning the discretized convolution needs to be expanded to be applicable to the inputs that are typically multidimensional arrays of data:

$$S(i,j) = [I * K](i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n), \tag{5.3}$$

where the convolution is given for the example of 2-dimensional data, as for instance image data, with the input I, kernel K, and feature map S. Here, the kernel K contains the parameters or weights that are adapted during training of the CNN to extract a feature, given the respective input. In practice, the Kernel K is chosen to have a small spatial extent in comparison to the input I, meaning that the weights are nonzero in only a small region. An illustration of Eq. (5.3) is shown in Fig. 5.1.

Depending on the size of the kernel, the output is reduced in size compared to the input, if the convolution is restricted to positions where the filter fits completely into the input (cf. Fig. 5.1). This can be resolved by zero padding the input: adding rows and columns of zeros to the boundaries such that there is a valid position of the kernel for every pixel in the input. With this, the size from input to output is conserved.

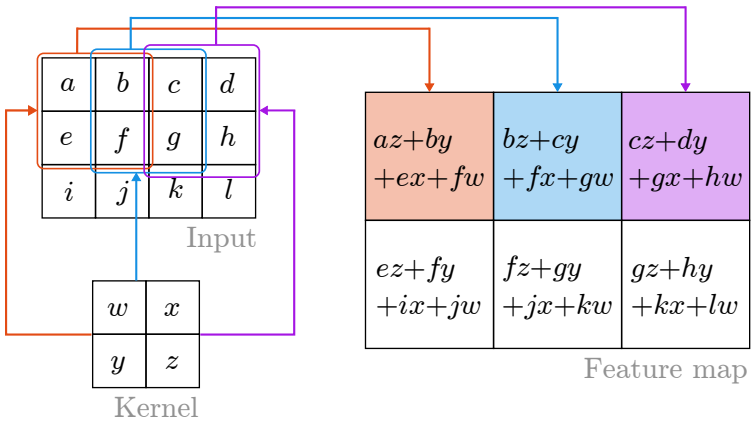

Figure 5.1: Illustration of the discrete convolution on 2-dimensional input: Here, the output in the feature map is restricted to positions where the kernel fits completely into the input. The red, blue, and purple arrows and boxes indicate the convolution in the three possible top positions of the kernel. Figure adapted from *Goodfellow et al.* [2016].

In deep learning terminology the convolution operation as introduced above is referred to as convolution filter. To build a deep learning model these convolution filters are organized in layers. Each layer contains several convolution filters with the aim to extract different features from the input. The number of channels (in analogy to the color channels of RGB-images) or the depth of the resulting feature map is determined by the number of filters used in the convolution layer. An illustration of a convolution layer with two filters is shown in Fig. 5.3. Several convolution layers are then chained together with the feature map resulting from one layer as the input to the next layer:

$$S_k(i,j) = [[[I * K_1] * K_2] * \cdots * K_k](i,j), \tag{5.4}$$

where for simplicity of the formulation only one convolution filter K_k is contained in the individual k layers here. Also note that typically an element-wise bias term is added after each convolution filter to introduce additional parameters. These are omitted here as well. In this layered structure the input of the network, the very first layer, and the output, the very last layer, are called visible layers, whereas the internal layers are called hidden layers. Consequently, the number of hidden layers defines the depth of the network. Fig. 5.2 depicts the connectivity pattern of a CNN in one dimension with two hidden convolution layers, feature map (s_1, s_2, s_3), input layer $(i_1, \ldots, i_5)$, and output z_1. By stacking the convolution layers, the regions in the network input that influence entries in the individual feature maps increase with depth of the network, which is also denoted as the perceptual field. In the example of Fig. 5.2 the perceptual field of z_1 has the size of 5.

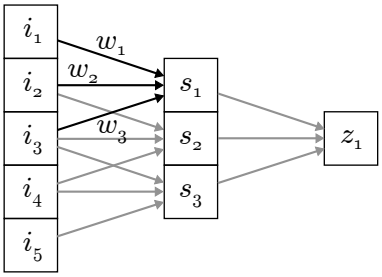

Figure 5.2: Illustration of the connectivity in the CNN: Example of a 1-dimensional representation of two convolutional layers with a kernel size of 3. The elements in the feature maps (boxes) are connected by the filter weights (arrows). In each layer the filter weights are redundant at the different positions.

5.2.2 Activation Functions

Since convolution itself is linear the chaining of convolutions as in Eq. (5.4) is still linear. However, the aim of using deep learning models is to be able to approximate very complicated functions. To achieve this, nonlinearities are introduced to the network structure.

The introduced nonlinearities are referred to as activation functions $g(S_k(i,j))$ and are applied to the feature maps, typically in an element-wise fashion. In deep learning several of these activation functions exist with their specific advantages and disadvantages. Here, only three possible choices are presented that are illustrated in Fig. 5.4.

One of the most classical activation functions is the sigmoid activation:

$$g_{\text{sigmoid}}(x) = \frac{1}{1+e^{-x}}. \tag{5.5}$$

Before the introduction of rectified linear units (ReLU), the sigmoid activation was commonly used in neural networks. the major drawback of the sigmoid activation is that it saturates over the larger portion of its domain, meaning it returns only values close to 1 for $x > 5$ and values close to 0 for $x < -5$. This becomes a problem when we calculate the gradients of the activation as it is needed for the optimization method used to train the network (5.3.3). This makes it hard to train neural networks that employ sigmoid activation functions in practice. The introduction of rectified linear units

$$g_{\text{relu}}(x) = \max(0, x) \tag{5.6}$$

alleviates this for positive input values. However, for negative input values this drawback still exists and may lead to many dead activations in the network, meaning that the activation kills the signal as it always returns 0 for any negative input. To circumvent this it may help to initialize the parameters of the convolution filters with

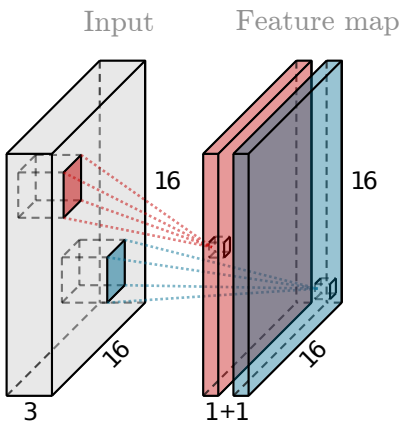

Figure 5.3: Illustration of a single convolution layer: In this example the input has a depth of three, as for instance for the red, green, and blue channels of a color image. The depicted layer contains two convolution filters, shown as smaller boxes in the input with red and blue faces, that extract two different features. The results for every filter position is stored in the feature map and its depth is given by the number of filters in the layer. Two exemplified entries are depicted as small boxes in the feature map connected with the filters by dashed lines of the respective color.

positive values. Another approach is to use leaky rectified linear units (leaky ReLU) [*Maas et al.*, 2013]:

$$g_{\text{lrelu}}(x) = \max(\alpha x, x), \tag{5.7}$$

where the slope of the function in the negative domain is chosen to be $\alpha < 1$, effectively preventing dead activations there. Since ReLU and leaky ReLU are very close to a linear function they make it comparably easy to train the neural network and represent a sensible standard choice of activation functions.

Note that ReLU and leaky ReLU are not differentiable at $x = 0$ and therefore seem to be unsuited for neural networks. Nonetheless, as iterative methods are used for optimization of the parameters it can be argued that the value $x = 0$ is quite rarely encountered in the process of training. Also, the issue can be avoided by setting the gradient returned by the activation function to 1 for $x = 0$.

5.2.3 Pooling and Strided Convolutions

As empirically found, the depth of the network is important for CNNs to perform well [e.g., *Krizhevsky et al.*, 2012; *Simonyan and Zisserman*, 2014a; *Szegedy et al.*, 2015]. To be able to design deep CNNs there is a need to constrain the number of parameters

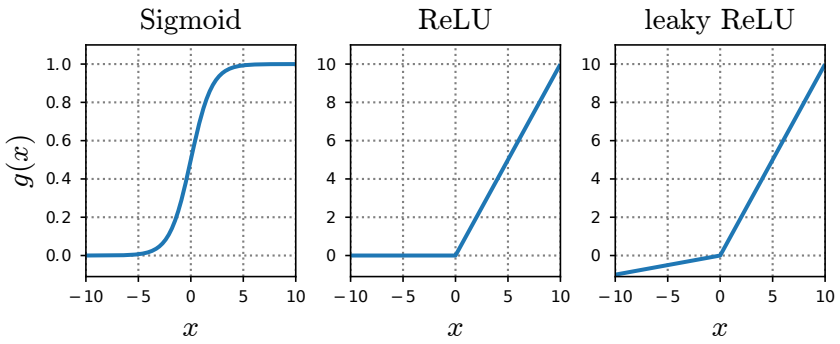

Figure 5.4: Illustration of activation functions: example activation functions that are applied to the feature maps to introduce nonlinearities and therefore the expressive power of the neural network.

in the model to preserve the computational feasibility. Therefore, methods to downsample the feature maps, i.e. to decrease their resolution, are introduced. There are two common approaches in doing so: (i) additional pooling layers at several locations of the network and (ii) convolution filters with a discrete stride larger than 1 are used at some instances of the convolution layers.

Pooling is typically applied to the feature maps following the activation functions. The individual channels of the feature maps are divided in small rectangular regions or neighborhoods of a few pixels. From these neighborhoods a metric is calculated determining a single output. For the case of max pooling, this output is the maximum value in the neighborhood [*Zhou and Chellappa*, 1988]. There are also several other pooling options by taking some weighted average over the pixel region. Depending on the size of the neighborhood, the resolution of the feature map is decreased, while the number of channels is conserved. Additionally, pooling introduces some degree of invariance to translations of the input data, as for small spatial shifts in the pixel values, the output of the pooling layer remains similar. Note that this might be an issue, if estimation of movement in the data is the task the CNN is ought to learn.

Strided convolution, on the other hand, avoids the introduction of additional layers to the network. At some of the existing convolution layers, the convolution filters are only applied to every r-th position of its input. This can be expressed as

$$S(i,j) = [I * K](i,j) = \sum_m \sum_n I(r \cdot i - m, r \cdot j - n) K(m,n), \tag{5.8}$$

where r is the stride of the filter. Choosing $r = 1$, this again results in Eq. (5.3). Depending on the choice of r this also reduces the resolution of the returned feature map and simultaneously reduces the number of computations.

With either of these two approaches, the information contained in the abstract features is condensed with increasing depth. Also, there is evidence that for the application to image recognition there is no performance difference between the two choices

[*Springenberg et al.*, 2014]. Given this, when designing a convolution one might choose the more simple approach by omitting pooling layers.

5.2.4 Transposed Convolutions

Depending on the task the neural network is ought to learn, a mapping from the condensed abstract feature representation is needed to reconstruct the output. For instance for image classification this is usually accomplished by adding few fully connected layers that map to the given class labels. When the aim is to reconstruct multidimensional output data the condensed representation has to be extrapolated. This up-sampling can also be incorporated into the learning process and is typically done by introducing transposed convolutions [*Zeiler et al.*, 2010; *Zeiler et al.*, 2011] to the network. Note that in the deep learning literature transposed convolution layers are often referred to as deconvolution layers, albeit this term is technically incorrect as the mathematical definition of deconvolution describes a different operation.

To understand transposed convolutions, the discrete convolution introduced in Eq. (5.3) has to be revisited. Resuming to the example of 2-dimensional data, the input and output can be rearranged as vectors $\mathbf{i} = (i_1, i_2, \ldots, i_i)^\top$ and $\mathbf{s} = (s_1, s_2, \ldots, s_j)^\top$, by assigning the entries from left to right and top to bottom. For the example as illustrated in Fig. 5.1 this results in

$$\mathbf{i} = \begin{pmatrix} a \\ b \\ c \\ \vdots \\ k \\ l \end{pmatrix} \text{ and } \mathbf{s} = \begin{pmatrix} az + by + ex + fw \\ bz + cy + fx + gw \\ cz + dy + gx + hw \\ ez + fy + ix + jw \\ fz + gy + jx + kw \\ gz + hy + kx + lw \end{pmatrix} \tag{5.9}$$

as input and feature map vectors, respectively. With this, the convolution with the kernel K can be rewritten as matrix multiplication:

$$\mathbf{K} \cdot \mathbf{i} = \mathbf{s}, \tag{5.10}$$

where $\mathbf{K}$ is a sparse Toeplitz matrix constructed from the weights in the convolution kernel K. Each row shares the weights of the previous row while being shifted. For the given 2-dimensional example $\mathbf{K}$ is a doubly block circulant matrix, a special case of a Toeplitz matrix, and takes the form

$$\mathbf{K} = \begin{pmatrix} z & y & 0 & 0 & x & w & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & z & y & 0 & 0 & x & w & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & z & y & 0 & 0 & x & w & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & z & y & 0 & 0 & x & w & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & z & y & 0 & 0 & x & w & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & z & y & 0 & 0 & x & w \end{pmatrix}, \tag{5.11}$$

which samples down from $\mathbf{i}$ with length 12 to $\mathbf{s}$ with length 6 in the example. Constructing a transformation that is capable of up-sampling while maintaining the connectivity structure from convolution filters as depicted in Figs. 5.1 to 5.3 can be achieved with the multiplication by the transposed matrix $\mathbf{K}^\top$ that is therefore also defined by the kernel K:

$$\mathbf{K}^\top \cdot \mathbf{i}_\mathrm{t} = \mathbf{s}_\mathrm{t}, \tag{5.12}$$

where $\mathbf{i}_\mathrm{t}$ and $\mathbf{s}_\mathrm{t}$ are now the input and output vectors respectively. This up-sampling is called transposed convolution, where the dimension of $\mathbf{s}_\mathrm{t}$ is larger than the dimension of $\mathbf{i}_\mathrm{t}$.

Equivalent to the convolution filters, the transposed convolutions are arranged in layers containing several of these transformations, while the adaptability of the weights of the transposed convolutions makes the extrapolation learnable by the deep learning method. By also applying activation functions to the output of each transposed convolution layer, the extrapolation can be nonlinear as well.

5.3 Training Process

Designing CNNs, using the concepts introduced above, aims at creating deep learning models with great expressive power. To enable the CNN to actually approximate the complicated mappings to perform the desired task, the parameters of the network need to be adapted. In the process of training, the adaption of the parameters is guided by learning the task on a large set of example data, the training dataset. First an objective function on the output generated by the CNN needs to be defined. In deep learning this objective function is composed by the so-called loss function on the data and added regularization on the parameters to counter overfitting. The loss function provides a global signal on how to adapt the parameters that is distributed to the individual parameters with backpropagation. According to this signal, typically the gradients, the parameters are adapted iteratively during optimization.

5.3.1 Loss Functions and Regularization

The loss function quantifies the performance of the CNN on the task based on the output of the network. In a supervised learning problem this is done by introducing a measure of comparison between the network prediction and the ground truth contained in the training dataset.

Defining the loss function L_i for the individual training data examples i, the global loss over the training dataset is the averaged sum $L = \frac{1}{N}\sum_i L_i$, where N is the size of the training dataset. For regression tasks, aiming to predict real valued quantities that in general can be multidimensional, often the squared L^2 vector norm of the difference

between the estimated prediction $\mathbf{s}^{\text{est}}$ and the ground truth $\mathbf{s}^{\text{true}}$ is employed as loss:

$$L_i(\mathbf{s}^{\text{est}}) = \|\mathbf{s}^{\text{est}} - \mathbf{s}^{\text{true}}\|_2^2 = \sum_{j=1}^{N_s} (s_j^{\text{est}} - s_j^{\text{true}})^2, \tag{5.13}$$

where $\mathbf{s}^{\text{est}}$ and $\mathbf{s}^{\text{true}}$ are enrolled into vectors in the same way as in Eqs. (5.9) and (5.10) and N_s is their respective dimension. Another common choice for the loss is the L^1 vector norm of the difference:

$$L_i(\mathbf{s}^{\text{est}}) = \|\mathbf{s}^{\text{est}} - \mathbf{s}^{\text{true}}\|_1 = \sum_{j=1}^{N_s} |s_j^{\text{est}} - s_j^{\text{true}}|, \tag{5.14}$$

depending on the desired influence of outliers. Several other choices can be made that might depend on the task specifically and can significantly facilitate the process of training.

Since a forward pass of the model is needed to calculate the loss for each data example, it is computationally infeasible to calculate the global loss for large datasets. Sampling a minibatch of m examples that are randomly drawn from the training dataset and calculating the loss over the minibatch

$$L_m = \frac{1}{m} \sum_{i=1}^{m} L_i \tag{5.15}$$

as an approximation of the global loss overcomes this issue. Subsequently, the minibatch loss is calculated iteratively to update the model parameters, whereby a training epoch is defined by the number of iterations needed to process every of the N/m minibatches and therefore every training example once. In practice, the minibatch sizes range from 1 to a few hundreds, depending on how many examples can be loaded into the memory simultaneously.

Because of the ability of CNNs to approximate very complicated mappings, it is crucial to introduce regularization during the training process. This is done to prevent the model from overfitting the training data and thereby enabling it to generalize to data that it has not encountered during training.

A common approach is to include norm penalties $\Omega(\mathbf{w})$ on the network parameters $\mathbf{w}$ into the objective function J:

$$J(\mathbf{w}) = L(\mathbf{w}) + \alpha\Omega(\mathbf{w}), \tag{5.16}$$

where the additional hyperparameter $\alpha > 0$ is introduced to scale the norm penalty. Regularization based on the L^2 norm of the network parameters

$$\Omega_{\mathrm{L}^2}(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2 = \frac{1}{2}\mathbf{w}^\top\mathbf{w} \tag{5.17}$$

is also known as weight decay. Over the course of the training process this regularization term incrementally decreases the network parameters that are not decisive to

decreasing the loss function $L(\mathbf{w})$. Ideally, these parameters that the loss function is not sensitive to are decreased to zero, effectively reducing the degrees of freedom of the model. Along this line, also pooling and strided convolutions (Section 5.2.3) can be interpreted as a measure of regularization. Hereby, the number of parameters of the network is strongly reduced, when compared to an architecture of the same depth that doesn't implement pooling or strided convolutions. Consequently, the degrees of freedom are reduced and the possibility to overfit is contained.

The obvious way to increase the generalization of a model is to train it on a larger dataset. However, the amount of available data is limited, but can be synthetically increased by data augmentation. The idea is to transform the input data in such a way that it still corresponds to the same ground truth. For some tasks this is straight forward, e.g. in image classification the location and orientation of the object in the image should not affect the given label, hence additional generic data can be created by performing these operations on the existing data. For tasks, where the position and orientation of the features in the input are crucial for the correct prediction, ways to properly augment the data are not obvious. In these cases data augmentation can still be used to introduce random noise to the pixels of the input images to make the model predictions more robust. Otherwise, neural networks seem to not cope well with noise in the input data [*Tang and Eliasmith*, 2010].

Applying random noise to the parameters of the hidden layers while training also increases the robustness of the model and improves generalization. The most radical approach is known as dropout [*Srivastava et al.*, 2014], where a random fraction of the weights is set to zero during each model forward pass. This randomly eliminates some connections in the network, enforcing the model to learn a redundant representation of the task.

Beyond the comparatively generic measures to regularize the model as introduced above it can be beneficial to introduce restrictions that represent prior knowledge about the output data. For the prediction of synthetically generated sea surface temperature data using an encoder-decoder CNN *de Bezenac et al.* [2017] successfully added the divergence, the magnitude, and the smoothness of the flow field to the objective function.

5.3.2 Backpropagation

Executing a forward pass of the model results in the calculation of the loss during training. Calculating the gradient of the loss with respect to the model weights $\nabla_{\mathbf{w}} L_i(\mathbf{w})$ represents the signal on how to adapt the parameters to minimize the loss. Backpropagation [*Rumelhart et al.*, 1986] distributes the gradients to the individual model parameters during the backward pass.

To understand the principle of backpropagation we need to investigate the signal path backwards through the network starting from the loss. The network can be interpreted as interlinked chained mathematical operations. The signal propagation along one of these chains can be understood with the chain rule of calculus. For real

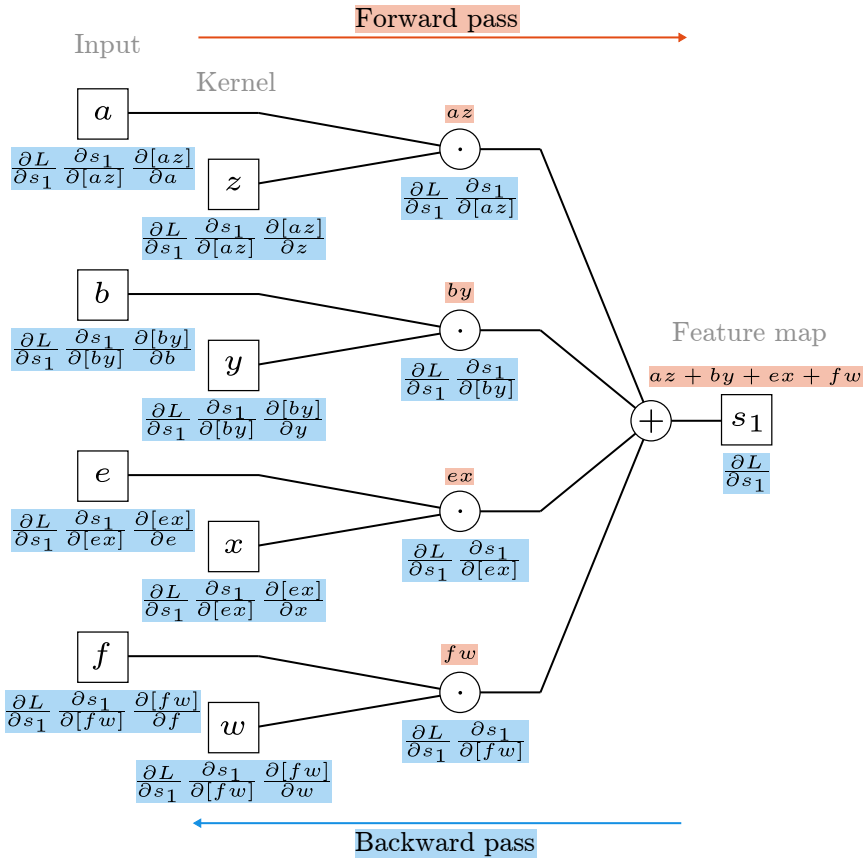

Figure 5.5: Computational graph of a discrete convolution as illustrated in Fig. 5.1 for the first output element s_1. In the forward pass (red) the input values and the kernel weights (boxes) are mapped to the output with simple operations (circles). In the backward pass (blue) the partial derivatives of the loss L are calculated using the chain rule.

valued functions, the chain rule is used to calculate the derivatives by decomposition into functions with known derivatives. With $x \in \mathbb{R}$, $p : \mathbb{R} \to \mathbb{R}$, and $q : \mathbb{R} \to \mathbb{R}$ let $y = q(x)$ and $z = p(y) = p(q(x))$. For the derivative the chain rule states

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x}. \tag{5.18}$$

Backpropagation makes use of the chain rule to propagate the derivatives through the network in an efficient way.

Further, the concept of a computational graph of the network explains how the chain rule is calculated along the individual network paths. However, illustrating the complete computational graph even of a simple CNN would already be unfeasible. To

get an understanding, Fig. 5.5 depicts the computational graph for one element in the feature map s_i of the convolution as formulated in Eq. (5.10) and illustrated in Fig. 5.1. In the given example, the illustration shows how the value of the element s_1 is calculated in the forward pass, while the derivatives in the backward pass for each connection are calculated using the chain rule. Suppose, the derivative of the loss $\frac{\partial L}{\partial s_1}$ is known at s_1 either from already backpropagating the derivative to this point of the network, or the output s_1 is at the last layer of the network. For instance, the derivative of the loss at input element f then results in

$$\frac{\partial L}{\partial f} = \frac{\partial L}{\partial s_1}\frac{\partial s_1}{\partial [fw]}\frac{\partial [fw]}{\partial f} = \frac{\partial L}{\partial s_1}\frac{\partial [az+by+ex+fw]}{\partial [fw]}\frac{\partial [fw]}{\partial f} = \frac{\partial L}{\partial s_1}w, \tag{5.19}$$

when only considering the contribution as depicted in Fig. 5.5. In the forward pass the input at element f additionally contributes to the output at elements s_2, s_4, and s_5 (cf. Eqs. (5.9) and (5.10)). Using backpropagation, the sum over all the contributions is calculated. For the input element f this results in

$$\frac{\partial L}{\partial f} = \frac{\partial L}{\partial s_1}w + \frac{\partial L}{\partial s_2}x + \frac{\partial L}{\partial s_4}y + \frac{\partial L}{\partial s_5}z. \tag{5.20}$$

Enrolling the derivatives at every input and output in the vectors $\nabla_{\mathbf{i}} L = \left(\frac{\partial L}{\partial a}, \frac{\partial L}{\partial b}, \ldots, \frac{\partial L}{\partial l}\right)^\top$ and $\nabla_{\mathbf{s}} L = \left(\frac{\partial L}{\partial s_1}, \frac{\partial L}{\partial s_2}, \ldots, \frac{\partial L}{\partial s_m}\right)^\top$, where the indices $\mathbf{i}$ and $\mathbf{s}$ denote the partial derivatives with respect to the inputs and outputs, the backpropagation of the convolution can be written as

$$\mathbf{K}_{\mathbf{w},\text{backprop}} \cdot \nabla_{\mathbf{s}} L = \nabla_{\mathbf{i}} L, \tag{5.21}$$

where

$$\mathbf{K}_{\mathbf{w},\text{backprop}} = \begin{pmatrix} z & 0 & 0 & 0 & 0 & 0 \\ y & z & 0 & 0 & 0 & 0 \\ 0 & y & z & 0 & 0 & 0 \\ 0 & 0 & y & 0 & 0 & 0 \\ x & 0 & 0 & z & 0 & 0 \\ w & x & 0 & y & z & 0 \\ 0 & w & x & 0 & y & z \\ 0 & 0 & w & 0 & 0 & y \\ 0 & 0 & 0 & x & 0 & 0 \\ 0 & 0 & 0 & w & x & 0 \\ 0 & 0 & 0 & 0 & w & x \\ 0 & 0 & 0 & 0 & 0 & w \end{pmatrix}. \tag{5.22}$$

When comparing Eqs. (5.10) and (5.11) with Eqs. (5.21) and (5.22) it turns out that the convolution with $\mathbf{K}$ in the forward pass results in a transposed convolution with $\mathbf{K}_{\mathbf{w},\text{backprop}} = \mathbf{K}^\top$ in the backward pass and vice versa. In this sense, convolutions and transposed convolutions are symmetric with respect to the direction of the signal propagation through the network. Hence, the backpropagation of transposed convolutions is convolution and already known.

Analogously, backpropagating the derivatives of the loss with respect to the model weights $\nabla_{\mathbf{w}} L = \left(\frac{\partial L}{\partial w}, \frac{\partial L}{\partial x}, \ldots, \frac{\partial L}{\partial z}\right)^\top$ can be computed with

$$\mathbf{K}_{\mathbf{i},\text{backprop}} \cdot \nabla_{\mathbf{s}} L = \nabla_{\mathbf{w}} L \tag{5.23}$$

that represents a convolution with the input values. For the given example $\mathbf{K}_{\mathbf{i},\text{backprop}}$ takes the values

$$\mathbf{K}_{\mathbf{i},\text{backprop}} = \begin{pmatrix} f & g & h & j & k & l \\ e & f & g & i & j & k \\ b & c & d & f & g & h \\ a & b & c & e & f & g \end{pmatrix}, \tag{5.24}$$

which is not sparse since the weights contribute to every output value in the forward pass.

When backpropagating the loss through the network, also the activation functions and the pooling layers, depending on their usage, need to be considered. Since the activation functions are applied element-wise their backpropagation is done by simply including the partial derivatives according to the chain rule. For pooling layers the backpropagation is equally straight forward. For max pooling the gradient is passed to the input with the maximum value during the forward pass, whereas for weighted pooling the gradient is distributed to all inputs according to the weights.

Accumulation of all derivatives with respect to all model weights $\mathbf{w}$ and the derivative of the regularization term results in the gradient $\nabla_{\mathbf{w}} J(\mathbf{w}) = \nabla_{\mathbf{w}} L(\mathbf{w}) + \alpha \nabla_{\mathbf{w}} \Omega(\mathbf{w})$, which is subsequently used in the stochastic gradient descent to optimize the model parameters.

5.3.3 Stochastic Gradient Descent

In deep learning, the optimization of the model parameters is done using an update procedure that is based on gradient descent [*Cauchy*, 1847]. The aim is to significantly reduce the objective function $J(\mathbf{w})$ that is composed of the loss $L(\mathbf{w})$ and regularization terms $\Omega(\mathbf{w})$ (cf. Eq. (5.16)). The gradient of the objective function, with respect to the weights $\nabla_{\mathbf{w}} J(\mathbf{w})$, resulting from backpropagation, represents the sensitivity of the model performance on the parameters $\mathbf{w}$ and is used for the iterative updates:

$$\mathbf{w}(t) = \mathbf{w}(t-1) - \epsilon \nabla_{\mathbf{w}} J(\mathbf{w}(t-1)), \tag{5.25}$$

with iteration t and the learning rate ϵ as hyperparameter to scale the update. Graphically speaking, this corresponds to the step-wise descent along the slope of the optimization landscape.

Since it is unfeasible to calculate the loss for large datasets the update is calculated using the loss over minibatches of size m with the according gradients $\nabla_{\mathbf{w}} L_m$. Depending on the size m the loss L_m is an approximation of the global loss L that fluctuates according to the radomly sampled data examples in the minibatch. Using

the minibatch approach, the update procedure is therefore called stochastic gradient descent (SGD) with the update rule for each iteration:

$$\mathbf{w}(t) = \mathbf{w}(t-1) - \epsilon \left[\nabla_{\mathbf{w}} L_m(\mathbf{w}(t-1)) + \alpha \nabla_{\mathbf{w}} \Omega(\mathbf{w}(t-1))\right]. \tag{5.26}$$

There are several improvements to SGD to prevent it from getting stuck at saddle points of the optimization landscape or being inefficient in regions, where the gradient is steep in one direction, but shallow in another. Also the noise in the gradients due to the minibatch approach can impede the optimization process substantially.

Momentum One popular approach involves taking an exponentially decaying moving average over the past gradients. This introduces a formal velocity $\mathbf{v}$ to the update:

$$\mathbf{w}(t) = \mathbf{w}(t-1) + \mathbf{v}(t-1) \tag{5.27}$$

with

$$\mathbf{v}(t-1) = \gamma \mathbf{v}(t-2) - \epsilon \nabla_{\mathbf{w}} L_m(\mathbf{w}(t-1)) \text{ and } \mathbf{v}(0) = \mathbf{0}, \tag{5.28}$$

where the hyperparameter $\gamma \in [0, 1)$ determines the decay of the previous gradient contributions. In analogy to a physical particle in a potential the method gains momentum along past gradients, keeps moving in this direction and, therefore, is able to alleviate issues with noisy gradients and in shallow regions. However, this improvement comes with the cost of an additional hyperparameter γ that needs to be set.

Adaptive Learning Rates Setting the learning rate ϵ adequately is crucial to a successful optimization process. If the value is too small, the update increments are also too small to allow for efficient optimization. Eventually the process will be able to explore a minimum of the objective function, but the required time might be unfeasible. If the learning rate and therefore the update steps are set too large, the optimization method is likely to overshoot the minima and is unable to converge. Choosing this hyperparameter by hand and testing for the resulting performance is therefore a tedious task and approaches to automatically adapt the learning rate are introduced. Among others, Adam [*Kingma and Ba*, 2014] is an algorithm that employs adaptive learning rates.

Adam, from 'adaptive moment estimation', estimates the first and second moments of the gradient to individually adapt the learning rate of each model weight. Analogously to the momentum method, the estimators $\mathbf{m}$ and $\mathbf{v}$ of the moments of the gradient $\mathbf{g}(t) = \nabla_{\mathbf{w}} L_m(\mathbf{w}(t))$ are calculated using moving averages:

$$\mathbf{m}(t) = \beta_1 \mathbf{m}(t-1) + [1-\beta_1]\mathbf{g}(t) = [1-\beta_1] \sum_{i=1}^{t} \beta_1^{t-i} \mathbf{g}(i), \tag{5.29}$$

$$\mathbf{v}(t+1) = \beta_2 \mathbf{v}(t) + [1-\beta_2]\mathbf{g}(t) \circ \mathbf{g}(t) = [1-\beta_2] \sum_{i=1}^{t} \beta_2^{t-i} \mathbf{g}(t) \circ \mathbf{g}(t), \tag{5.30}$$

with the exponential decay rates $\beta_1, \beta_2 \in [0, 1)$ and $\circ$ denoting the element-wise product of the vectors. Initializing the estimators to zero: $\mathbf{m}(0), \mathbf{v}(0) = \mathbf{0}$ results in biased estimation of the moments in the beginning of training. Considering the expected value of the estimator $\mathbf{m}(t)$ yields

$$\langle \mathbf{m}(t) \rangle = \left\langle [1 - \beta_1] \sum_{i=1}^{t} \beta_1^{t-i} \mathbf{g}(i) \right\rangle \tag{5.31}$$

$$= \langle \mathbf{g}(t) \rangle [1 - \beta_1] \sum_{i=1}^{t} \beta_1^{t-i} + \zeta \tag{5.32}$$

$$= \langle \mathbf{g}(t) \rangle [1 - \beta_1^t] + \zeta, \tag{5.33}$$

where the formula for the finite geometric series has been used in the last line. The estimation error ζ of the true gradient $\langle \mathbf{g}(t) \rangle$ emerges due to the moving average, but can be neglected, if only small weights are assigned to gradients far in the past. This leaves the term $[1 - \beta_1^t]$ due to the initialization with zero that needs to be corrected. Analogously, this can be derived for the estimation of the second moment $\langle \mathbf{v}(t) \rangle$ and a bias correction is introduced to the estimators:

$$\hat{\mathbf{m}}(t) = \frac{\mathbf{m}(t)}{1 - \beta_1^t} \text{ and } \hat{\mathbf{n}}(t) = \frac{\mathbf{n}(t)}{1 - \beta_2^t}. \tag{5.34}$$

With this, the update rule for Adam is defined as

$$\mathbf{w}(t) = \mathbf{w}(t-1) - \epsilon \frac{\hat{\mathbf{m}}(t)}{\sqrt{\hat{\mathbf{n}}(t)} + \delta}, \tag{5.35}$$

where $\sqrt{\hat{\mathbf{n}}(t)}$ denotes the element-wise square root of $\hat{\mathbf{n}}(t)$, a small constant δ is introduced to stabilize the division for small gradients, and ϵ sets an upper bound for the learning rate.

With the adaptive learning rate and the introduction of momentum with the moving averages, Adam is seen to be a comparably reliable optimization method for deep learning problems and is therefore widely used. Still, the method introduces additional hyperparameters, but *Kingma and Ba* [2014] provide very robust default values of $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-3}$, and $\delta = 10^{-8}$ often eliminating the need to hand-pick them.

5.3.4 Weight Initialization

The iterative optimization methods used for training deep learning models require an starting point to be specified. This starting point is set with the initialization of the model weights, representing one point in the parameter space. Since training in deep learning is sufficiently difficult, the most methods are highly sensitive to the initialization scheme. Consequently, the choice may influence how quickly the optimization converges and can impede convergence altogether. Unfortunately, the current insight

into optimizing deep networks is limited and therefore specifically designed initialization strategies are lacking. However, there are some strategies to achieve nice properties for the optimization.

When initializing the weights, the aim is to set them as close as possible to the optimal values. Since already very rough estimates of the optimal values are inaccessible a reasonable first guess is that the trained weights will be symmetrically distributed with an expected value of zero. Additionally, it is important that the weights are initialized with varying values to avoid filter redundancies in the individual layers of the network. This often leads to a initialization strategy, where the weights are randomly sampled from small numbers distributed around zero. Following this simple approach, issues due to too small or too large values can arise. From backpropagation (Section 5.3.2) we know that the gradients in the shallow layers (close to the network input) depend on the weights in the deeper layers as they appear as coefficients in the chain rule (cf. Eqs. (5.20) to (5.22)). Suppose the weights are initialized with large values. In the backward pass the multiplication of the large values leads to phenomenon called exploding gradients in the shallower layers resulting in large parameter updates during optimization that are likely to impede convergence. Contrarily, very small initialization values lead to vanishing gradients that are too small to propagate the signal through the network during the backward pass, resulting in insufficiently small updates of the weights. The desired appropriate choice prevents these effects and allows optimal signal propagation through the network.

Glorot and Bengio [2010] introduced a strategy that conserves the variance of the activations $g_l(\mathbf{s}_l)$ across layers l, meaning that

$$\mathrm{var}(g_l(\mathbf{s}_l)) = \mathrm{var}(g_{l-1}(\mathbf{s}_{l-1})), \tag{5.36}$$

effectively eliminating issues with exploding and vanishing gradients. For hyperbolic tangent $\tanh(\mathbf{s}_l)$ as activation function, they deduce the initialization of the weights per layer $\mathbf{w}_l$ sampled from a normal distribution

$$\mathbf{w}_l \sim \mathcal{N}\left(\mu = 0, \sigma^2 = \frac{1}{n_{l-1}}\right) \tag{5.37}$$

to fulfill the requirement of Eq. (5.36). Here, the variance σ^2 depends on the number of connections n_{l-1} to the previous layer $l-1$. *He et al.* [2015] extended this approach specifically for the usage of ReLU activation functions and derive a weight initialization scheme with an adapted variance σ^2:

$$\mathbf{w}_l \sim \mathcal{N}\left(\mu = 0, \sigma^2 = \frac{1}{2n_{l-1}}\right). \tag{5.38}$$

Commonly this strategy is seen as a default choice, when ReLU activation functions and their derivatives are employed in the network architecture.

5.4 Network Architectures for Flow Field Estimation

In parts based on *Kreyenberg et al.* [2019].

In this work I utilized two CNNs to estimate flow fields for active solute transport. The networks were trained on purely synthetic data that was generated by numerical simulation of the physical process over a broad parameter range with the aim to incorporate the physical representation of the process. The used models are detailed in Section 5.4.1 and their training is described in Section 5.4.2. The datasets used to train, validate, and test the models are introduced in Section 5.4.3.

5.4.1 Models

The architectures I used for the CNNs are encoder-decoders very similar to two FlowNet2 variants as proposed by *Ilg et al.* [2017]. The first architecture I will refer to as Architecture 1 from here on, which is derived from FlowNet2-s, and the second architecture I will refer to as Architecture 2, which is derived from FlowNet2-SD. The architectures are illustrated in Figs. 5.6 and 5.7, while the details are given in Table 5.1, Table 5.2, Fig. A.7, and Fig. A.8. The CNNs were implemented using the Caffe deep learning software framework [*Jia et al.*, 2014] and trained on a single Nvidia GTX 1080 Ti GPU. As input the CNNs take two subsequent concentration fields, concatenated to produce the input image, and output a flow field prediction.

Both CNNs consist of the same base architecture. An encoder of either 10 or 13 convolution layers followed by a decoder of 4 transposed convolution layers, with leaky ReLU activation functions with negative slope of 0.1 following the individual layers. Additional convolution layers were used to predict the estimated flow fields. At the input of the networks, I added Gaussian noise with standard deviation uniformly sampled from [0, 0.04] as data augmentation to the concentration fields. Generally, every second convolution layer is strided and reduces the resolution of the feature maps by a factor of 2 to allow for a deep architecture that is still reasonably fine at the bottleneck. Exceptions are at the first few layers, where the resolution is reduced more often. Including more layers and therefore increasing the nonlinearity with the according leaky ReLU layers is beneficial as it makes the approximator more discriminative [*Simonyan and Zisserman*, 2014a]. In the decoder each of the four transposed convolution layers increases the resolution by a factor of 2. Following the decoder a convolution layer without leaky ReLU layer predicts the flow estimation and the flow is split in its $x-$ and $z-$components. I introduced another convolution layer (without leaky ReLU layer) with a single 1×1 convolution for each flow velocity component. This enables the CNNs to adapt a scaling of the flow velocity vector components.

The differences of Architecture 1 and 2 are the following: (i) Architecture 2 swaps the first few layers from Architecture 1 for more layers with 3×3 convolution filters instead of the 7×7 and 5×5 convolution filters. (ii) Architecture 2 makes use of skip connections (or residual connections) that are omitted in Architecture 1. These

Table 5.1: Network and training hyperparameters for Architecture 1.

Network architecture:		
Layer	N_{filter}, kernel size, stride, pad	Resolution[a]
	Encoder	
Input	-	768×384
conv1[b]	24, 7×7, 2, 3	384×192
conv2[b]	48, 5×5, 2, 2	192×96
conv3[b]	96, 5×5, 2, 2	96×48
conv3_1[b]	96, 3×3, 1, 1	96×48
conv4[b]	192, 3×3, 2, 1	48×24
conv4_1[b]	192, 3×3, 1, 1	48×24
conv5[b]	192, 3×3, 2, 1	24×12
conv5_1[b]	192, 3×3, 1, 1	24×12
conv6[b]	384, 3×3, 2, 1	12×6
conv6_1[b]	384, 3×3, 1, 1	12×6
	Decoder	
convT5[b]	192, 4×4, 2, 1	24×12
convT4[b]	96, 4×4, 2, 1	48×24
convT3[b]	48, 4×4, 2, 1	96×48
convT2[b]	24, 4×4, 2, 1	192×96
	Prediction	
Predict_conv	2, 3×3, 1, 1	192×96
Slice	-	192×96; 192×96
Scale_X_conv || Scale_Z_conv	1, 1×1, 1, 0 || 1, 1×1, 1, 0	192×96; 192×96
Training hyperparameters:		
Optimization algorithm	Adam[c]	$\beta_1 = 0.9$, $\beta_2 = 0.999$
Learning rate	10^{-5}	
Batch size	24	
Training epochs	540	
Weight decay	$4 \cdot 10^{-4}$	

[a]Refers to the resolution of the feature map after the respective network layer.
[b]Layer is followed by a leaky ReLU activation function with negative slope of 0.1.
[c]*Kingma and Ba* [2014].
|| denotes parallel layers at same depth of the network.

Table 5.2: Network hyperparameters for Architecture 2.

Layer	N_{filter}, kernel size, stride, pad	Resolution[a]
	Encoder	
Input	-	768×384
conv0[b]	64, 3×3, 1, 1	768×384
conv1[b]	64, 3×3, 2, 1	384×192
conv1_1[b]	128, 3×3, 1, 1	384×192
conv2[b]	128, 3×3, 2, 1	192×96
conv2_1[b]	128, 3×3, 1, 1	192×96
$\leftarrow$ skip2	-	-
conv3[b]	256, 3×3, 2, 1	96×48
conv3_1[b]	256, 3×3, 1, 1	96×48
$\leftarrow$ skip3	-	-
conv4[b]	512, 3×3, 2, 1	48×24
conv4_1[b]	512, 3×3, 1, 1	48×24
$\leftarrow$ skip4	-	-
conv5[b]	512, 3×3, 2, 1	24×12
conv5_1[b]	512, 3×3, 1, 1	24×12
$\leftarrow$ skip5	-	-
conv6[b]	1024, 3×3, 2, 1	12×6
conv6_1[b]	1024, 3×3, 1, 1	12×6
	Decoder	
convT5[b] || upscaling[c]	512, 4×4, 2, 1 || -	24×12
$\rightarrow$ skip5	-	-
convT4[b] || upscaling[c]	256, 4×4, 2, 1 || -	48×24
$\rightarrow$ skip4	-	-
convT3[b] || upscaling[c]	128, 4×4, 2, 1 || -	96×48
$\rightarrow$ skip3	-	-
convT2[b] || upscaling[c]	64, 4×4, 2, 1 || -	192×96
$\rightarrow$ skip2	-	-
	Prediction	
Predict_conv	2, 3×3, 1, 1	192×96
Slice	-	192×96; 192×96
Scale_X_conv || Scale_Z_conv	1, 1×1, 1, 0 || 1, 1×1, 1, 0	192×96; 192×96

[a] Refers to the resolution of the feature map after the respective network layer.
[b] Layer is followed by a leaky ReLU activation function with negative slope of 0.1.
[c] For details of the upscaling connection see Fig. A.10.
|| denotes parallel layers at same depth of the network.
$\leftarrow$ and $\rightarrow$ indicate the connection pattern of the skip connections.

Table 5.3: Training hyperparameters for Architecture 2.

Optimization algorithm	Adam[a]	$\beta_1 = 0.9$, $\beta_2 = 0.999$
Learning rate	10^{-5}	
Batch size	12	
Training epochs	540	
Weight decay	$4 \cdot 10^{-4}$	

[a] *Kingma and Ba* [2014].

connections bypass the downsampling and upsamling of feature maps (cf. Fig. A.9) at certain layers to allow signals at a finer resolution to pass through the network [*He et al.*, 2016]. (iii) Architecture 2 incorporates upscaling connections [*Ilg et al.*, 2017], which can be seen as a special type of skip connections. Parallel to the transposed convolution layers, the flow is predicted at a coarse resolution with a convolution layer of two filters and then upsampled using a transposed convolution layer (cf A.10), while no activation function is employed. (iv) In general, Architecture 2 has substantially more filters in each layer with a maximum of 1024 filters per layer compared to a maximum of 384 filters per layer for Architecture 1.

For both Architectures, the coarsening in the encoder effectively reduces the resolution by a factor of 64, while the decoder refines the resolution by a factor of 16, resulting in a total resolution decrease by a factor of 4. I used bilinear interpolation as a postprocessing step on the estimated flow fields to recover the original resolution of the input concentration fields during deployment of the CNNs. Further upsampling using additional transposed convolution layers in the decoder does not necessarily produce significantly better results *Dosovitskiy et al.* [2015].

5.4.2 Training Scheme

During training I did sample down the true flow field to match the resolution of the estimated flow field before the loss was calculated. As training loss I calculated the L^2 regularized sum of squares error of the flow field components:

$$J(\mathbf{w}) = \sum_{j=1}^{N^{\text{est}}} \left[\left[\tilde{u}_{x,j}^{\text{est}} - \tilde{u}_{x,j}^{\text{true}} \right]^2 + \left[\tilde{u}_{z,j}^{\text{est}} - \tilde{u}_{z,j}^{\text{true}} \right]^2 \right] + \frac{\alpha}{2} \mathbf{w}^\top \mathbf{w} \tag{5.39}$$

with network weights $\mathbf{w}$, weight decay $\alpha = 4 \cdot 10^{-4}$, and the number of pixels N^{est} of the estimated, coarse flow fields. For the training scheme I used a batch size of 24 for Architecture 1 and 12 for Architecture 2 and a learning rate of 10^{-5} over 540 training epochs. Following *Dosovitskiy et al.* [2015], I used the Adam optimization algorithm [*Kingma and Ba*, 2014] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for the optimization during the training, while the weights were initialized according to *He et al.* [2015]. Additionally

to the loss calculation at the output of the network, Architecture 2 incorporates calculation of the loss at the coarser resolution levels. After each transposed convolution layer, the coarse flow is predicted and the loss is calculated as formulated in Eq. (5.39) and illustrated in Fig. A.11. The loss contributions are then combined by a weighted sum, where the weights are chosen such that the individual contributions are similar. The respective training hyperparameters are summarized in Tables 5.1 and 5.3.

With this, the essential modifications toward the original FLowNet2 architectures are: (i) I only introduced noise to augment the data, whereas no other transformations where applied at the input. (ii) I introduced an additional convolution layer with 1×1 convolution filters to scale the individual flow components of the prediction. (iii) I introduced the component-wise loss as given in Eq. (5.39) compared to the standard L^1 loss (cf. Eq. (5.14)). Additionally, Architecture 1 omits the skip connections of FlowNet2-s.

5.4.3 Datasets

The datasets I used to train, validate and test the CNNs were obtained from the numerical experiments as introduced in Chapter 3. The concentration fields of two subsequent time steps were grouped into image pairs, together with the flow field that conveys the solute between the two. I constructed five distinct datasets that I named according to their purpose and to indicate their origin from the individual set of numerical experiments. The datasets are described below and a summary of the details is given in Table 5.4.

TrainNE1 & ValidateNE1 The first training and validation sets were obtained from NE1. Accordingly, these datasets represent the process of plain active solute transport. Out of the total of 5,616 image pairs I equidistantly excluded 269 image pairs as the first validation dataset ValidateNE1 resulting in 5,347 image pairs for the Training dataset TrainNE1.

TrainNE1+3 & ValidateNE1+3 The data in the second training and validation sets originate from the combined data of NE1 and NE3. Accordingly, active solute transport is represented as in TrainNE1 and ValidateNE1, whereby in addition the superscale convection is incorporated due to the background flow introduced in NE3. Since the background convection in NE3 is consistently initiated in the positive x-direction, I additionally included the mirrored concentration and flow fields of NE3 to introduce a symmetric distribution of the flow direction. Again, I equidistantly excluded image pairs to create the validation set ValidateNE1+3 of 591 image pairs, which resulted in 11,705 image pairs for TrainNE1+3.

TestNE2 The test dataset is obtained from NE2, which is qualitatively different from NE1 and NE3, since the upper concentration boundary condition is chosen to be

Table 5.4: Summary of the synthetic training, validation, and test datasets.

TrainNE1	
Number of image pairs	5,347
Range of Ra	[2,000, 27,000]
Resolution	768 px × 384 px
Upper boundary condition	$\tilde{C} = 1$
Superscale convection	not represented
ValidateNE1	
Number of image pairs	269
Range of Ra	[2,000, 27,000]
Resolution	768 px × 384 px
Upper boundary condition	$\tilde{C} = 1$
Superscale convection	not represented
TrainNE1+3	
Number of image pairs	11,705
Range of Ra	[2,000, 27,000] (background flow only in [3,750, 13,750])
Resolution	768 px × 384 px
Upper boundary condition	$\tilde{C} = 1$
Superscale convection	represented as described in Section 3.3 for 6,358 image pairs
ValidateNE1+3	
Number of image pairs	591
Range of Ra	[2,000, 27,000] (background flow only in [3,750, 13,750])
Resolution	768 px × 384 px
Upper boundary condition	$\tilde{C} = 1$
Superscale convection	represented as described in Section 3.3 for 322 image pairs
TestNE2	
Number of image pairs	829
Range of Ra	[3,750, 13,750]
Resolution	768 px × 384 px
Upper boundary condition	$\tilde{C} = 1$ for 230px $< x <$ 538px and $\tilde{t} < \frac{5}{6}\tilde{t}_{\max}$; $\tilde{C} = 0$ else
Superscale convection	not represented

nonuniform and a temporal cutoff of the supplied solute is introduced, while no superscale convection is represented (cf. Chapter 3). TestNE2 contains 829 image pairs in total.

I chose to train Architecture 1 on TrainNE1 to learn the physical representation of pure active solute transport. Architecture 2, the model with the larger capacity (more model weights), was trained on TrainNE1+3 to learn the physical representation of active solute transport along with the introduced superscale convection. The training processes of both CNNs were monitored using ValidateNE1 and ValidateNE1+3, respectively, and also to manually adapt the hyperparameters. TestNE2 was neither used to optimize the model parameters nor the hyperparameters. Therefore, this data represents the synthetic test case, where the true flow fields are available, to test the generalization of the CNNs toward a qualitatively different concentration boundary condition, besides the real world test case of the laboratory experiment, where the quantitative knowledge of the concentration boundary condition is missing and flow is experimentally inaccessible.

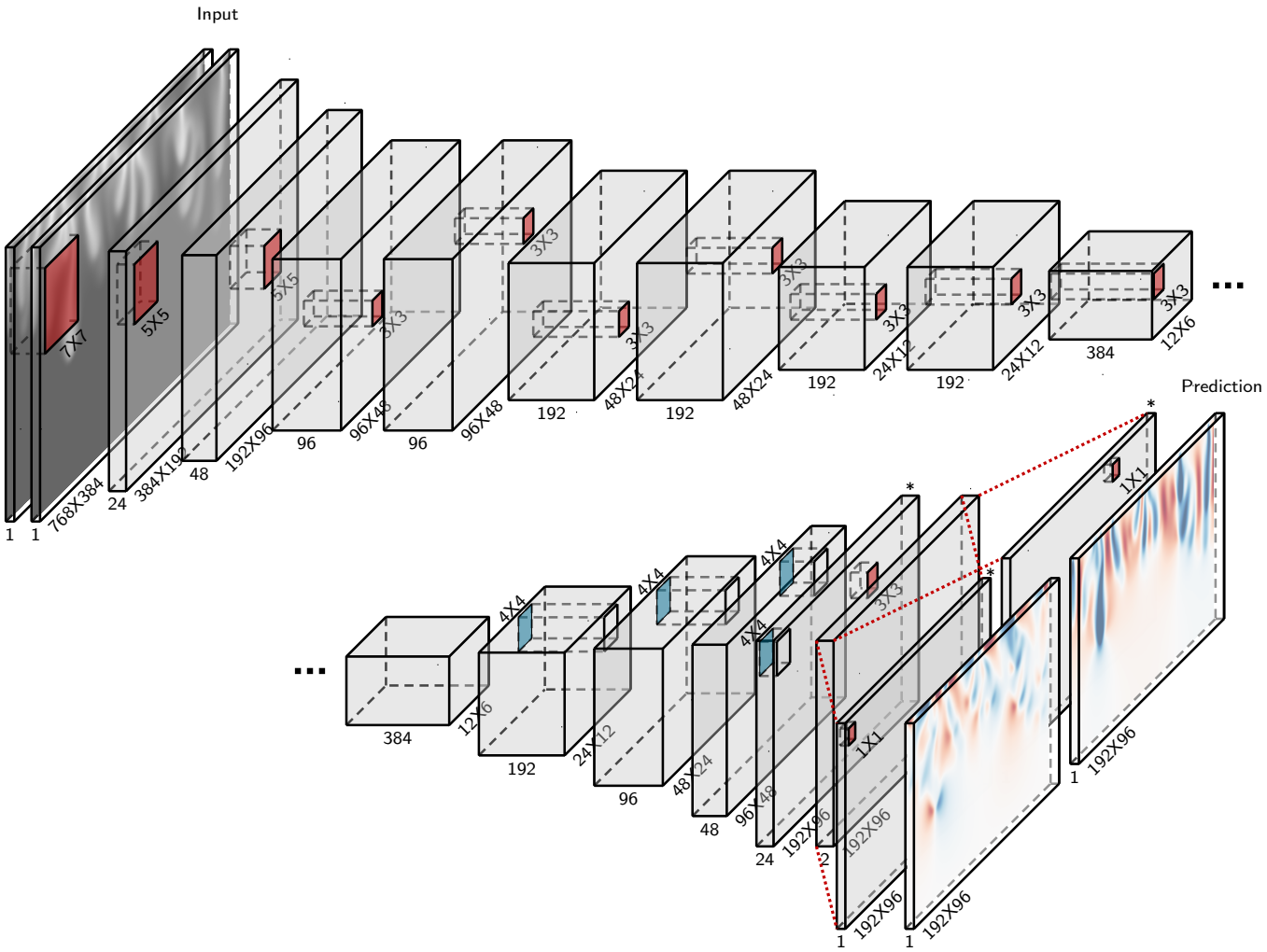

Figure 5.6: Architecture 1 (details in Table 5.1): Two subsequent concentration fields are fed into the network. Feature maps are depicted as light gray boxes. Convolution layers are depicted as boxes with red faces indicating their mapping from the respective feature map to the next. Transposed convolution layers are depicted as boxes with blue faces indicating their mapping from the previous to the respective feature map. Asterisks indicate convolution layers without leaky ReLU activation function. The dotted red lines indicate slicing of the feature map into x- and z-components of the predicted flow field. Bilinear interpolation as postprocessing of the predicted flow field is used to recover the full input resolution (not depicted).

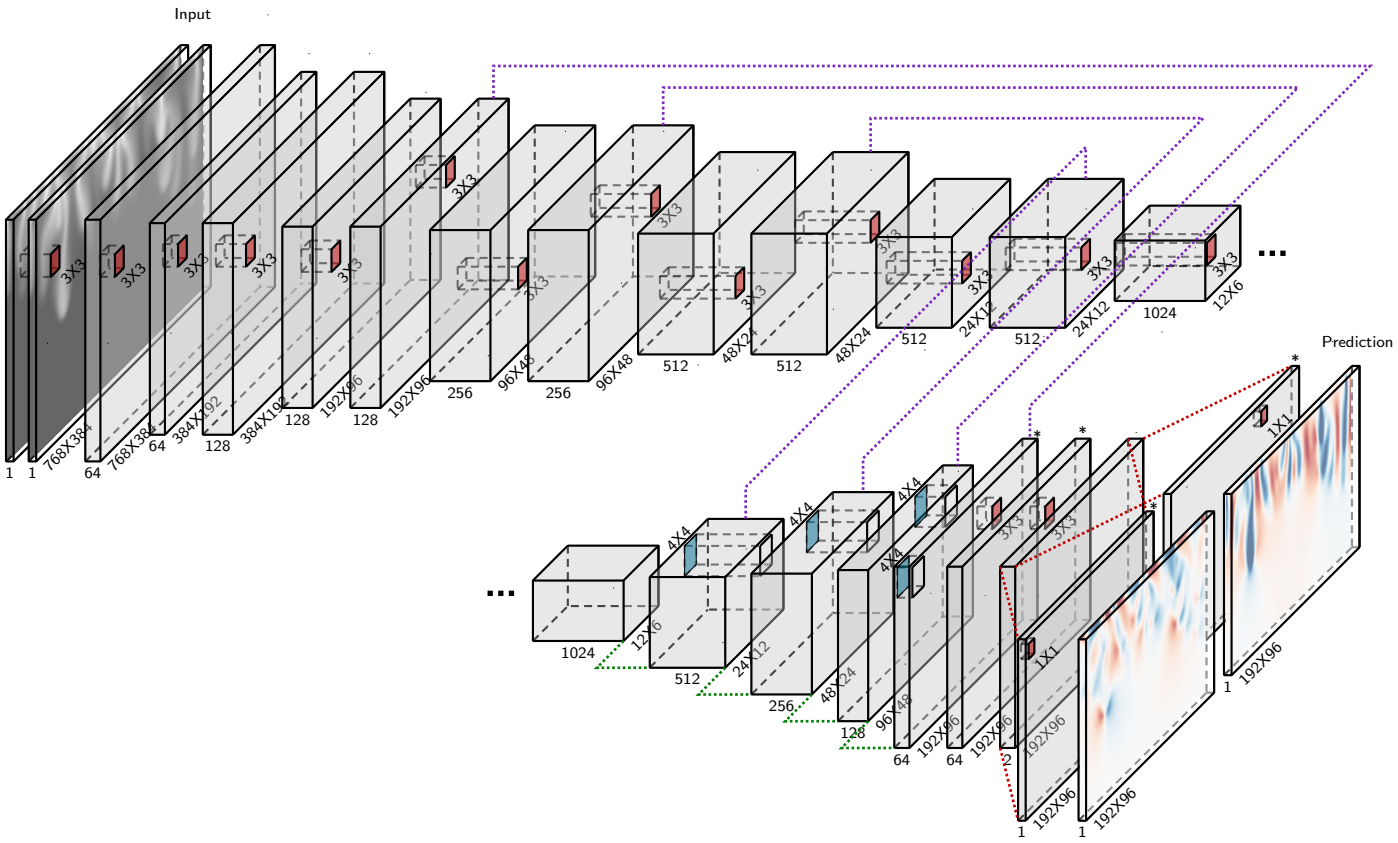

Figure 5.7: Architecture 2 (details in Table 5.2): Two subsequent concentration fields are fed into the network. Feature maps are depicted as light gray boxes. Convolution layers are depicted as boxes with red faces indicating their mapping from the respective feature map to the next. Transposed convolution layers are depicted as boxes with blue faces indicating their mapping from the previous to the respective feature map. Asterisks indicate convolution layers without leaky ReLU activation function. Skip and upscaling connections (details in Figs. A.9 and A.10) are indicated by the dotted purple and dotted green lines, respectively. The dotted red lines indicate slicing of the feature map into x- and z-components of the predicted flow field. Bilinear interpolation as postprocessing of the predicted flow field is used to recover the full input resolution (not depicted).

6 | Application

In parts based on *Kreyenberg et al.* [2019].

The conceptual objective of this work is to transfer information that is contained in synthetically generated data, which completely and faithfully represent the process understanding, to real world data, where system quantities are inaccessible. I realized this on the example of active solute transport, where I generated large datasets of the process by numerical simulation distributed over a wide parameter range. Two variants of a CNN, Architecture 1 and Architecture 2, were trained on that data to incorporate the physical process representation with respect to the prediction of the flow fields. The application of the trained CNNs to the laboratory experiment enabled the estimation of the flow fields in a real world case that were otherwise experimentally inaccessible.

This chapter presents the results of this approach and is structured as follows: The preprocessing of the laboratory experiment data used as a preparation step for the flow field estimation is described in Section 6.1. The true flow fields are inaccessible in the laboratory experiments. Therefore, the quality and accuracy of the flow field estimation is not directly measurable in this case. As introduced in Section 6.2, I used the estimated flow fields to propagate the measured concentration fields forward in time to make the performance of the flow field estimation assessable. The results of the flow field estimation using Architecture 1 trained on pure active solute transport (TrainNE1) are given for the synthetic test case and the real world test case in Sections 6.3 and 6.4, respectively. The representation of superscale convection, which is expected to be apparent in the laboratory experiment, is incorporated by training Architecture 2 on TrainNE1+3. The respective results are given in Section 6.5. Section 6.6 summarizes and discusses the results presented in this chapter.

6.1 Data Preprocessing

To prepare the data of the laboratory experiment for the flow field estimation I used the following preprocessing steps. To reduce the image noise, I used a standard median filter with a kernel size of 5×5 px. The use of a median image filter, in comparison to a Gaussian image filter, keeps the blurring minimal and therefore reduces the introduced error with respect to the dispersive transport process. Additionally, I used a threshold filter slightly above the base concentration of $C_0 = 86.5\,\mathrm{kg/m^3}$, setting all concentration values beneath $C = 90.0\,\mathrm{kg/m^3}$ to zero to extract the density fingers only.

To enable the direct comparison to the numerical experiment, I scaled the measured concentrations according to Eq. (2.35). Also, I chose to restrict my observations to

the upper 9.7 cm of the Hele-Shaw cell with a observation window width of 19.4 cm to achieve geometric similarity to the domain of the numerical experiments. With this, I received the resolution of 768×384 px for the concentration fields in accordance to the simulations.

6.2 Concentration Field Propagation

When estimating the flow fields with the trained CNNs, ground truth is only available for the numerical experiments, but not for the laboratory measurements. To be able to evaluate the performance of the estimation, I propagated the concentration fields forward in time using the estimated flow fields. The resulting estimated concentration fields were then compared to the concentration fields of the corresponding time step dt as a basis for the performance evaluation of the flow field estimation. The propagation was performed through two subsequent processing steps: (i) I used warping [*Ilg et al.*, 2017], the bilinear interpolation of the propagated concentration values, to account for the convective transport process and (ii) I used a standard Gaussian image filter with $\sigma = \sqrt{2D\mathrm{d}t}$ to approximate the dispersive transport process that is a solution to Eq. (2.19). Note that the phenomenology of the Gaussian image filter is similar, but not identical, to the physical dispersive process, since the smoothing is isotropic. Therefore, it is expected that an additional error is introduced that is not due to the performance of the flow field estimation.

As the training of the CNN was performed based on dimensionless simulations, the estimations contain dimensionless flow velocities $\tilde{\mathbf{u}}$. Therefore, I needed to scale these flow velocities with the dimensionless time step $\mathrm{d}\tilde{t}$ to get the displacement for the warping method. For the synthetic data, this was given by the chosen output time step of the numerical simulation $\mathrm{d}\tilde{t} = 0.02$. With the known dispersion coefficient of $D = 1/\mathrm{Ra}$ (cf. Eq. (2.43)) in the numerical simulations the standard deviation for the Gaussian image filter is given by $\tilde{\sigma} = \sqrt{2\,\mathrm{d}\tilde{t}/\mathrm{Ra}}$.

For the laboratory experiment, I lack the accurate knowledge of the dimensionless time step $\mathrm{d}\tilde{t}$ and the standard deviation for the Gaussian blurring $\tilde{\sigma}$, because of uncertainties in the experimental determination of $T_\mathrm{c} = 12\mu_w H/[\Delta\rho g \bar{d}^2_\mathrm{obs}]$ and D_m. When scaling the measurements contained in the observation area to be dimensionless, the relevant length scale is the height of the observation area $H_\mathrm{obs} = 9.7\,\mathrm{cm}$. For the time step this results in $\mathrm{d}\tilde{t} = \mathrm{d}t/T_\mathrm{c} = 0.034 \pm 0.07$. Because of the large uncertainties and since the values are expected to introduce biases that cannot be assessed, I chose to optimize the time step and standard deviation for the warping and Gaussian filtering of the measured data by a simple parameter scan, such that the root mean squared error between the propagated and the corresponding measured concentration field is minimized. This implicitly scales the flow velocities, resulting in a better agreement of the propagated and measured concentration fields. However, I argue that the incorporation of this information is valid, since the objective is to get the best estimate of

the inaccessible flow fields, which is different from the correct forward propagation of the concentration fields.

6.3 Results on the Numerical Experiments

For the first application case of the flow field estimation, I trained Architecture 1 on TrainNE1 to incorporate the process representation of pure active solute transport. This section presents the results on the synthetic test case TestNE2 along with a comparison to the validation data ValidateNE1. To present the qualitative results, I chose two representative image pair examples from ValidateNE1 and TestNE2 such that they exhibit the typical characteristics of the respective dataset. The estimated flow field is shown in Fig. 6.1(b) for the ValidateNE1 example and in Fig. 6.3(b) for the TestNE2 example. For comparison, the true flow fields are shown in Figs. 6.1(a) and 6.3(a), respectively.

The flow field estimation captures the characteristic of the true flow field reasonably well. In general, regions where the flow points downward and where the flow points upward are predicted reliably. As regions with upward flow in between of the density fingers are predicted without explicit information about the flow velocity there, since the solute concentration in these regions is zero, this shows that the CNN is able to learn the characteristics of the physical flow phenomenon of the density-driven instability. Also, the flow structures at the upper boundary are accurately reproduced with the lateral flow pointing toward the density finger seeding points. At the tips of the density fingers the flow separates toward the positive and negative $x-$direction, as it is found in the true flow field. Inspecting areas around the finger tips shows that the prediction gradually fails when moving further away from the tip. The method is unable to reliably predict the flow field there without information about the flow given as the transport of the concentration.

In addition to the good reproduction of the large flow patterns I find that some finer patterns can be predicted as well. For instance in Fig. 6.1(b) in the region around (420, 90) the complicated flow structure of interacting density fingers are reproduced. The flow pattern around (140, 210) is reproduced in the qualitative shape but not quite as accurately. In contrast, in the region around (40, 225) the prediction deviates apparently due to insufficient information given by the low concentration values there.

The predictive capabilities of the CNN for the validation data ValidateNE1 remain intact also for the test dataset TestNE2. This is quite remarkable, since there I chose the upper boundary condition for the concentration to be different from the boundary condition of the training (TrainNE1) and validation dataset (ValidateNE1) in spatial extent as well as in temporal duration (cf. Chapter 3). Especially the larger areas of the upward flow left and right to the middle region containing the density fingers are captured astonishingly well (cf. Figs. 6.3(a), (b)), when considering that the training data do not contain image pairs showing such large upwelling zones.

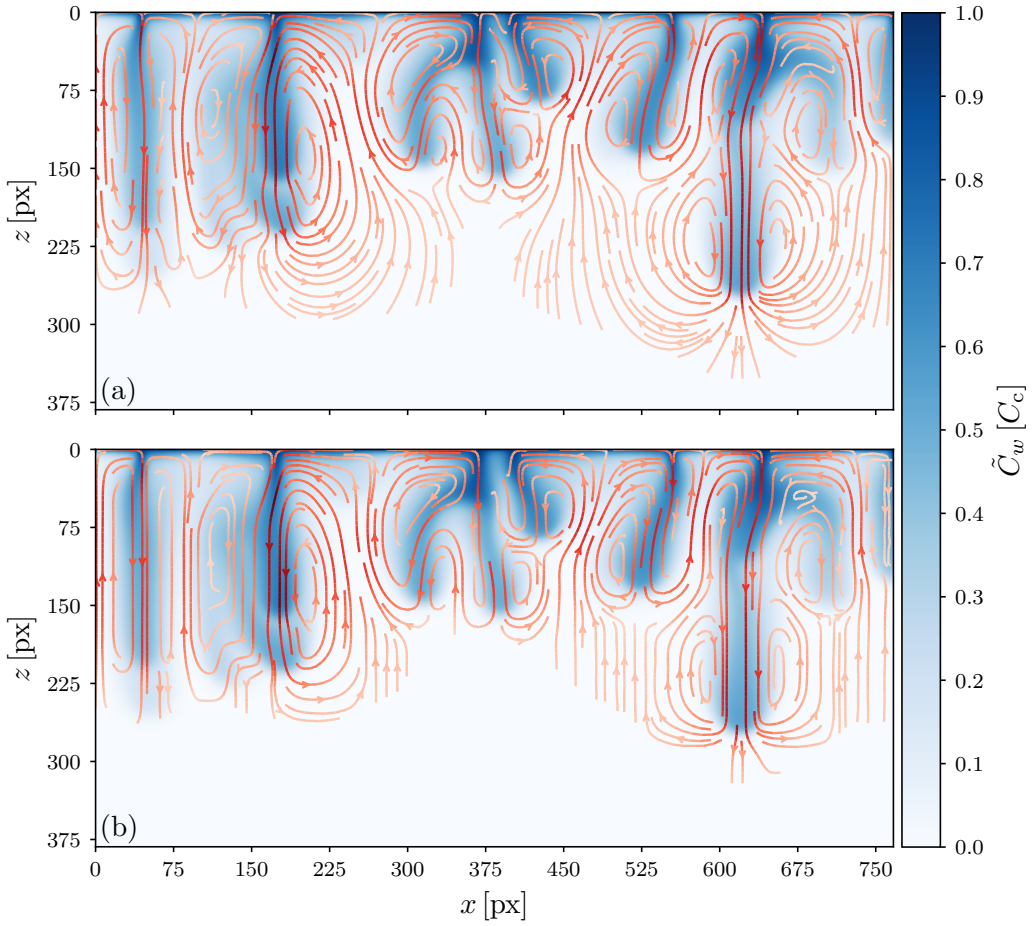

Figure 6.1: Estimated flow field for the ValidateNE1 example: truth (a) and estimation (b) of the flow field shown as streamlines (red color intensity indicates absolute flow velocity) on top of the color coded prior concentration field.

Figures 6.2 and 6.4 present the corresponding flow field divergence values for the true and the estimated flow field for the ValidateNE1 and the TestNE2 examples. For the true flow fields from numerical simulation, I observe divergences in the range $[-9.3, 8.6] \cdot 10^{-3}$ and divergences in the range $[-30.0, 9.0] \cdot 10^{-3}$ for the estimated flow fields. The larger range of divergences for the estimated flow fields arise from the imperfect estimation of the CNN and are related to the velocity errors occurring per characteristic time period T_c. Still, the encountered values are at least two orders of magnitude smaller than the encountered typical flow velocities of $|\tilde{\mathbf{u}}| \approx 3.5$.

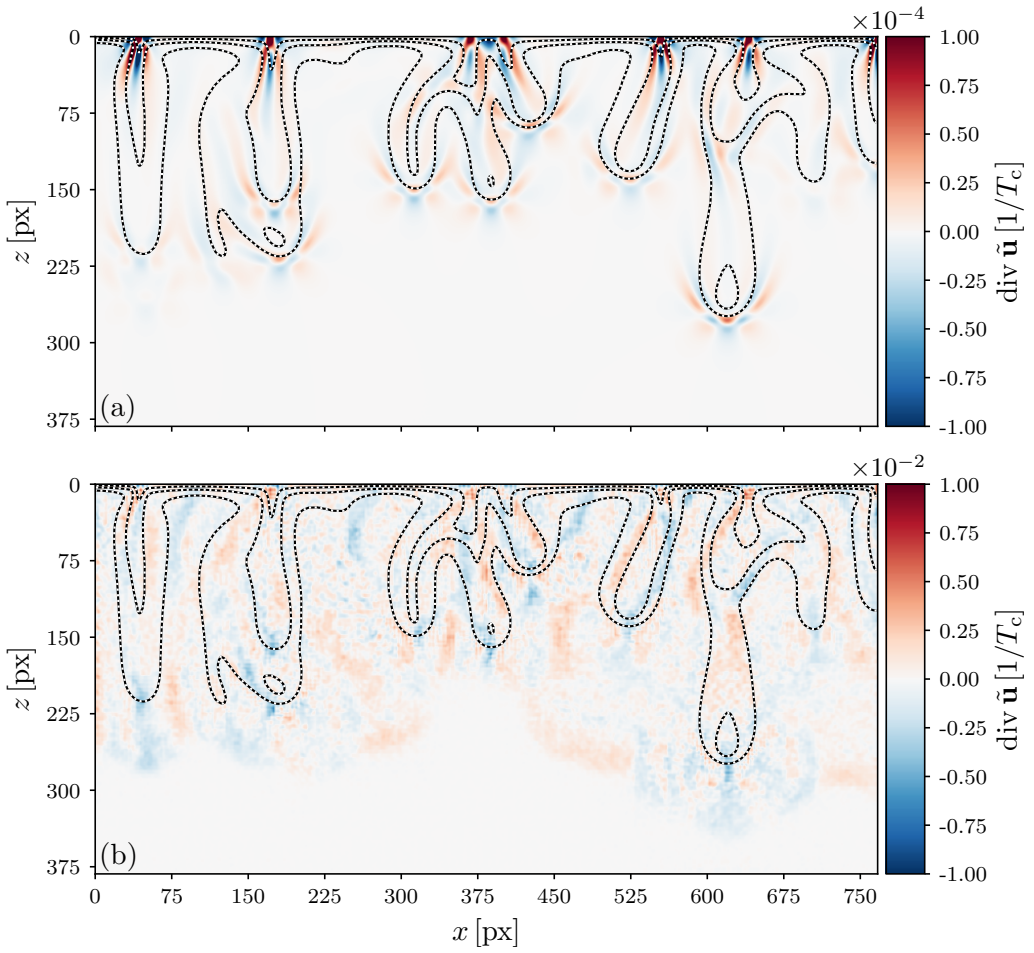

Figure 6.2: Flow field divergence for the ValidateNE1 example for the true (a) and the estimated (b) flow field. Concentration isolines are given at levels $\tilde{C}_w = (0.25, 0.5, 0.75)$. Note that for better visibility of low values the scales for the divergence differ by two orders of magnitude, while the actual deviation between maxima and minima is lower.

As a quantitative error measure, I took the mean endpoint error (MEPE = $\sum_{j=1}^{N} \sqrt{[\tilde{u}_j^{\text{est}} - \tilde{u}_j^{\text{true}}]^2 + [\tilde{w}_j^{\text{est}} - \tilde{w}_j^{\text{true}}]^2}/N$ with N being the number of pixels in the images) between the estimated and the true flow field. The distributions of this error measure over ValidateNE1 and the TestNE2 are shown in Figs. 6.5(a) and (b), respectively. The MEPE is given in the units of the characteristic velocity U_c and is a measure of the error in flow velocity averaged over all pixels of the respective image pair. Typical absolute velocity values encountered after the onset of the instability are in the range of $|\tilde{\mathbf{u}}| \approx 0.5\,U_c \ldots 0.8\,U_c$. Note that the MEPE is expected to result

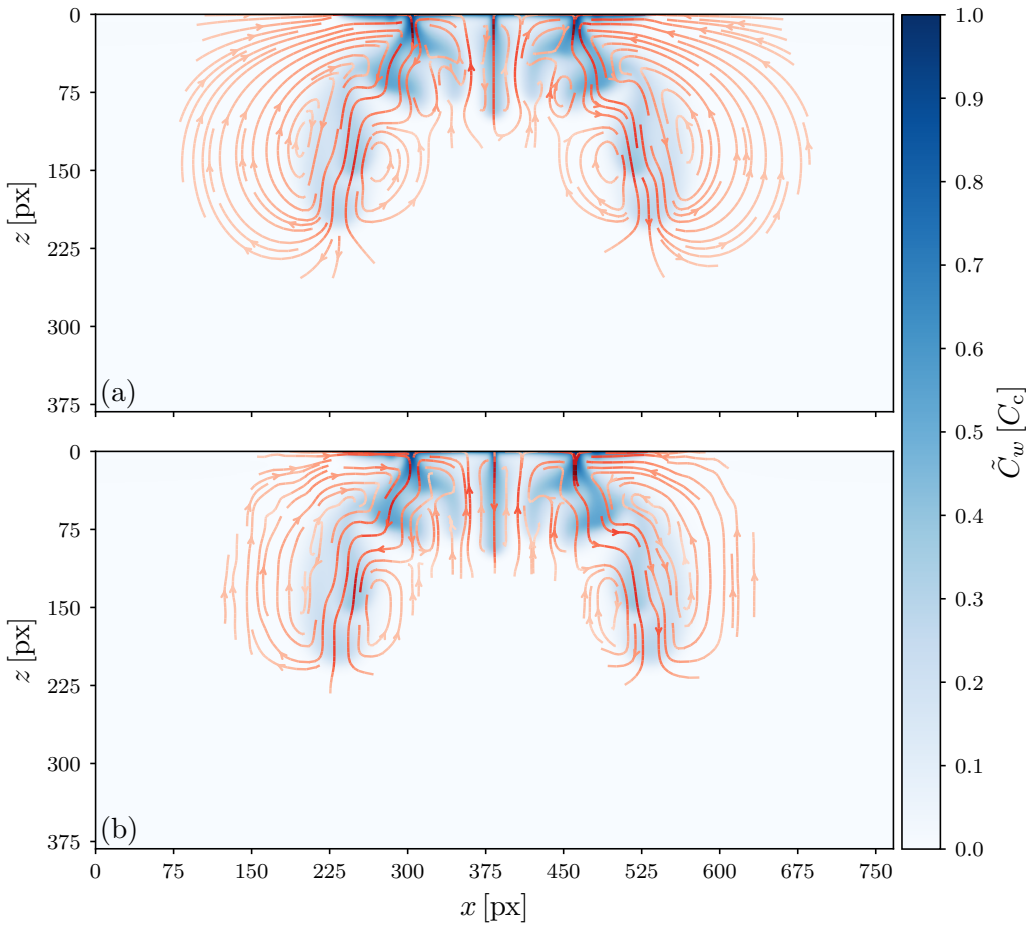

Figure 6.3: Estimated flow field for the TestNE2 example. Same representation as in Fig. 6.1.

in low values for image pairs with very little to no dynamics, hence for the very early development of the instability.

For ValidateNE1 the MEPE remains below 0.022 for all the image pairs, suggesting good agreement between the estimation and the truth. Image pairs with very little dynamics result in low values beneath 0.001. In fact the majority of the data count in this range originates from image pairs during the purely diffusive regime. For the image pairs after the instability onset, the majority is contained within the MEPE range from 0.007 to 0.022.

The distribution of the MEPE over TestNE2 (Fig. 6.5(b)) is similar to the distribution over the validation dataset with the maximum MEPE being 0.021. Note that for the test dataset the flow is being contained in smaller regions, given the nonuniform

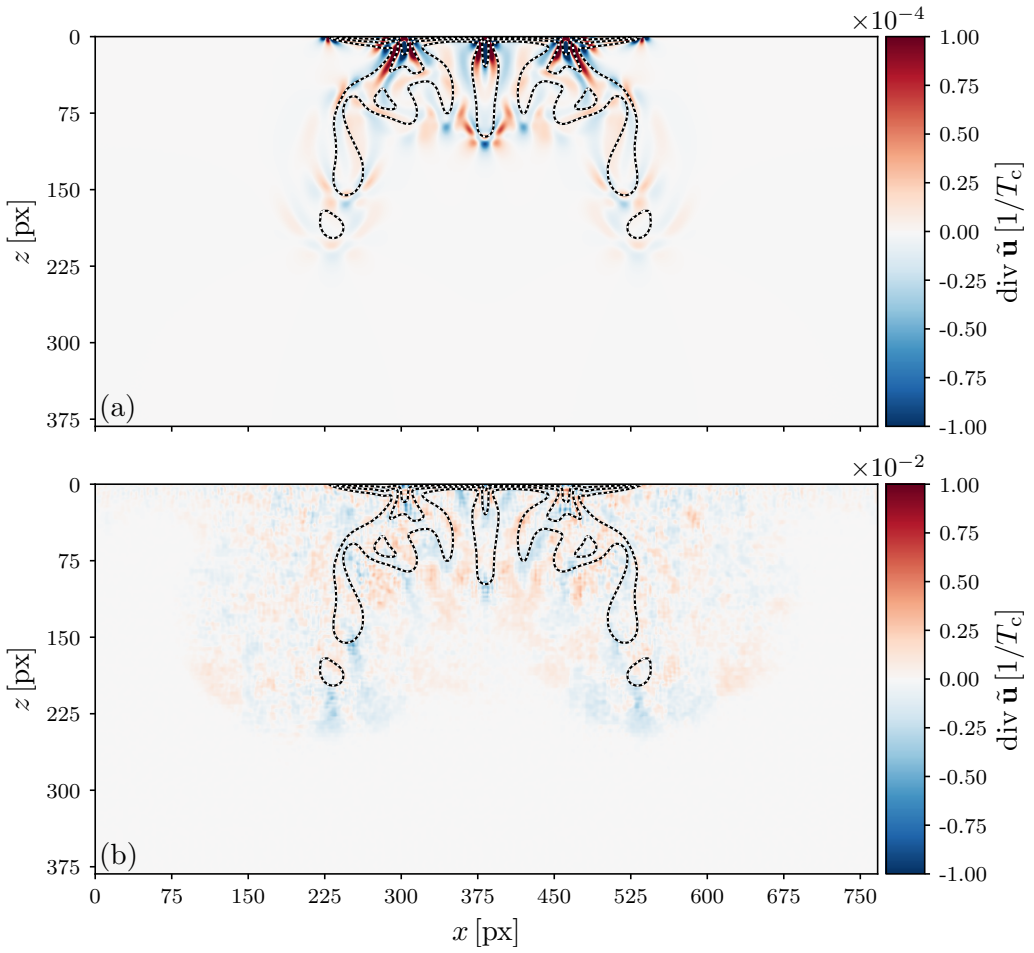

Figure 6.4: Flow field divergence for the TestNE2 example. Same representation as in Fig. 6.1.

boundary condition. In contrast to the MEPE distribution over ValidateNE1, the portion of image pairs with mean endpoint error below 0.001 is lower. This is due to the composition of the test dataset. With the nonuniform boundary condition, I introduce an earlier onset of the instability there. I observe a broad distribution of the MEPE in the range between 0.001 and 0.021. Overall the distribution exhibits that the estimation performs well on the test dataset, showing the robustness of the method toward the introduced spatial and temporal variability in the upper concentration boundary condition.

To get an assessment of the spatial distribution of the errors, I additionally calculated the angular error and the normalized velocity error as local error measures. The results for the ValidateNE1 example and the TestNE2 example are shown in Figs. 6.6 and 6.7, respectively. The respective MEPE values of the examples are indicated in

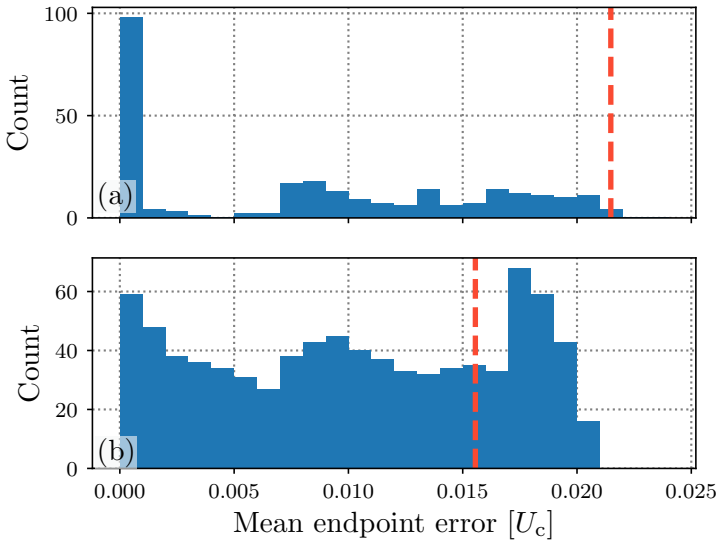

Figure 6.5: Error distributions of the mean endpoint error for ValidateNE1 (a) and TestNE2 (b). Mean endpoint error values of the representative dataset examples, shown in Figs. 6.1 and 6.3, respectively, are indicated by the red dashed lines.

Figs. 6.5(a) and (b) with the dashed red lines, supporting their choice as representative examples, as the majority of the concentration pairs exhibit a lower MEPE and hence a better agreement between truth and estimation. The angular error shows the angular difference between the estimated flow direction and the true flow direction for each pixel and is given in degrees. The normalized velocity error is the difference between the absolute estimated flow field and the absolute true flow field normalized by the maximum absolute value of the true flow field: $[|\tilde{\mathbf{u}}^{\text{est}}| - |\tilde{\mathbf{u}}^{\text{true}}|] / \max(|\tilde{\mathbf{u}}^{\text{true}}|)$ and yields structurally similar results to the L^2 error often used in the field of deep learning.

Figure 6.6(a) and Fig. 6.7(a) show that the direction of the flow is generally estimated accurately where information is given by the concentration. The CNN is even capable of estimating the flow direction accurately in larger regions surrounding the density fingers. Areas where the estimation of the flow direction fails are located directly at the flanks of density fingers. There the flow is typically small and deviates from being parallel to the orientation of the density fingers. Additionally, the concentration values are low. Combined, this results in low dynamics impeding the flow field estimation there. Also boundary effects become apparent in the ValidateNE1 example (Fig. 6.6(a)) at the rightmost density finger. There, the flow points downward directly at the right boundary causing the shape of the finger to be structurally different as it appears to be cut in half. In the regions further away from the density fingers, the flow velocity is close to zero, the angle correspondingly ill-defined.

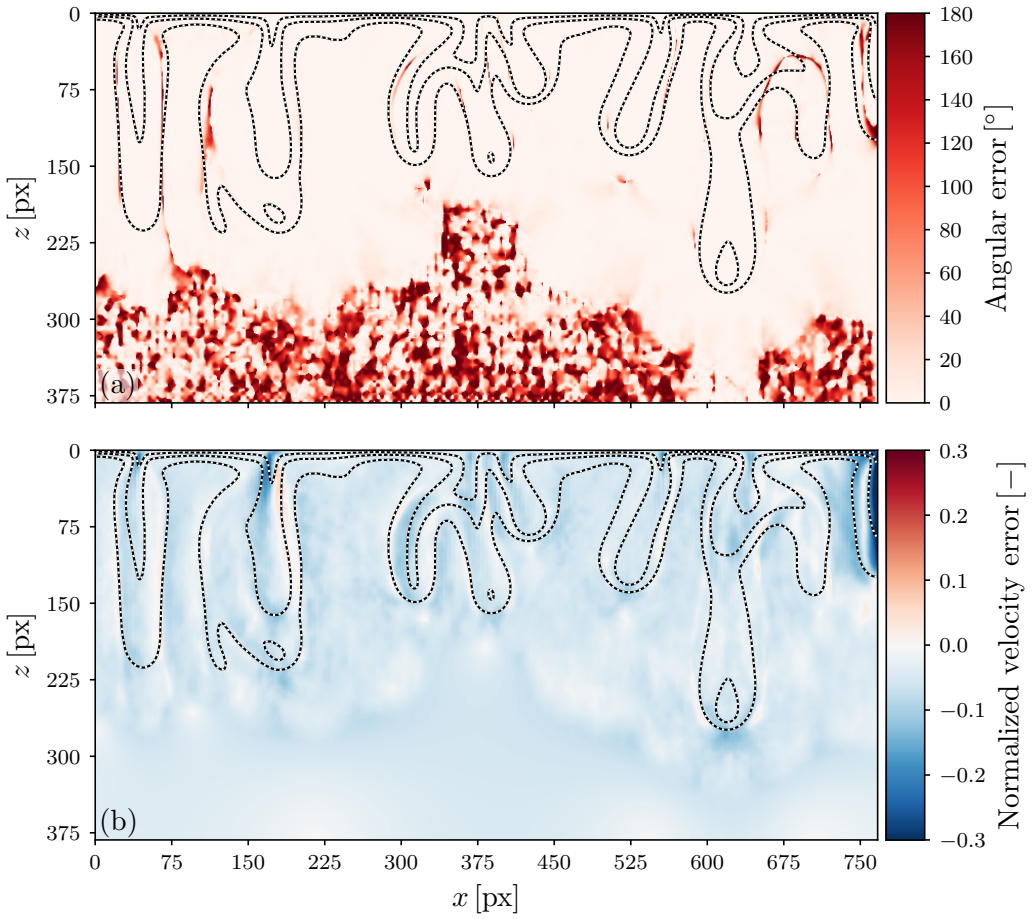

Figure 6.6: Error measures for the ValidateNE1 example: Angular error (a) and normalized velocity error (b) (negative for $|\tilde{\mathbf{u}}^{\text{est}}| < |\tilde{\mathbf{u}}^{\text{true}}|$). Concentration isolines are given for the levels $\tilde{C}_w = (0.25, 0.5, 0.75)$.

Figures 6.6(b) and 6.7(b) show that the method predominantly underestimates the flow velocity values. Apart from the boundary effects on the rightmost density finger in the ValidateNE1 example (Fig. 6.6(b)), large portions of the region where $\tilde{C}_w > 0$ exhibit a normalized velocity error roughly in the range between -13% to 6% also with a strong tendency to underestimate the flow. Locally, the error can exceed these values. For instance at the seeding point of the second density finger from the left the normalized velocity error ranges down to -23%. Also the flow velocities are underestimated in the regions with upward flow between the density fingers. There, the normalized velocity error roughly ranges from -7% to -4%. Figure 6.7(b) shows that the underestimation is generally less pronounced for the TestNE2 example. This is mainly due to the predominantly lower velocities (cf. Figs. 6.1(a) and 6.3(a)). However, the

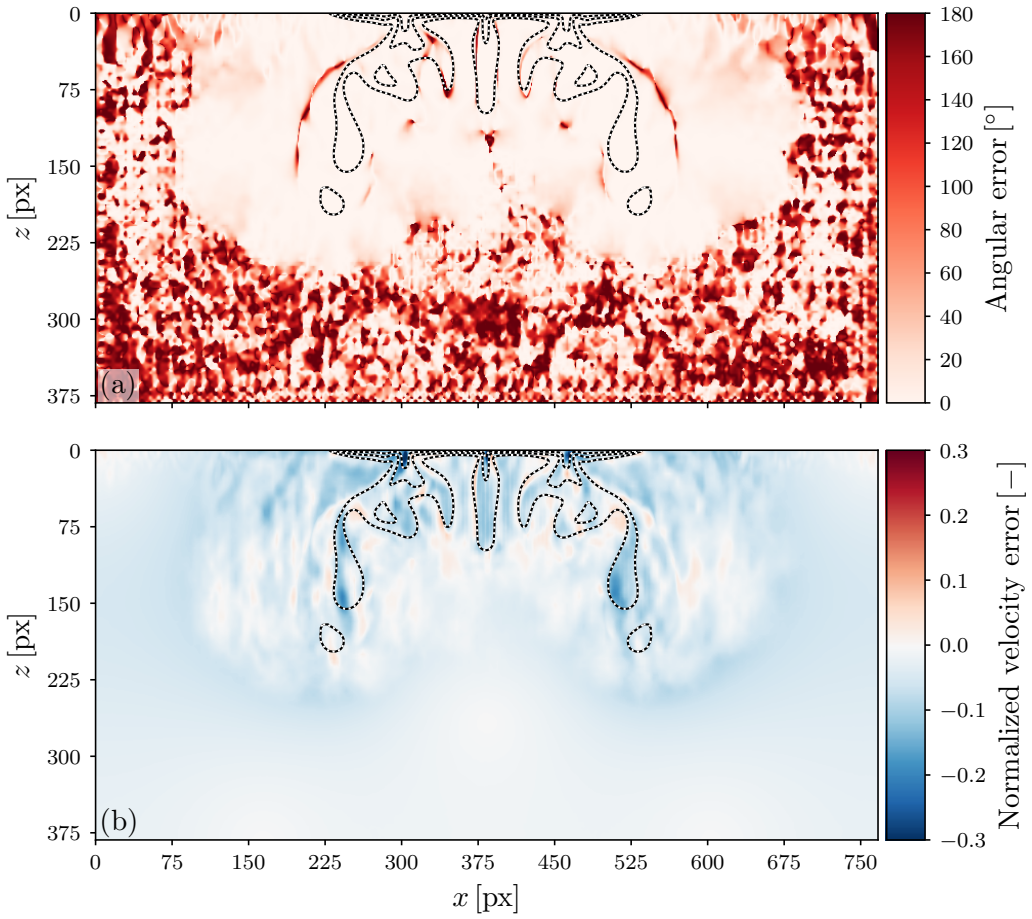

Figure 6.7: Error measures for the TestNE2 example. Same representation as in Fig. 6.6.

span of the normalized velocity error is similar to the validation dataset example with a range from −20% to 6%.

The incapability to estimate the absolute values of the flow velocities clearly shows the limits of the flow field estimation with the implementation of the CNN I used. The smooth concentration gradients that are encountered in the flow field estimation for active solute transport are quite different to the high contrast image data in typical applications of CNNs, like image classification. Together with the generally small displacements in the concentration field data, this seems to impede the quantitatively correct detection of the movement.

6.4 Results on the Laboratory Experiment

Also for the laboratory experiment, the flow fields are estimated using Architecture 1 trained on TrainNE1. Figure 6.8 shows the results on an exemplary concentration field pair. The quality of the flow field estimation is consistent with the estimation for the numerical experiment (cf. Fig. 6.1(b), and Fig. 6.3(b)). Predominantly, the flow structures seem to be estimated reliably. Also the resulting divergence of the estimated flow field (Fig. 6.8(b)) is consistent with the synthetic case (cf. Fig. 6.1(d)). Nevertheless, the ground truth for the laboratory experiment is inaccessible for direct performance evaluation. To assess the performance on the real data, I propagated the measured concentration fields forward in time using the estimated flow fields (see Section 6.2). The propagated concentration fields are then compared to the respective measured concentration fields.

As described in Section 6.2, I chose to optimize for the propagation time step and the Gaussian image filter standard deviation, although I am aware that this alleviates the expected underestimation of the flow field with the CNN. On the other hand, the physically scaled value for $\mathrm{d}\tilde{t}$ introduces estimation biases, that cannot be assessed. My choice results in $\mathrm{d}\tilde{t} = 0.0438$ and $\sigma = 2.6\,\mathrm{px}$ for Architecture 1 trained on TrainNE1 (results are illustrated in Fig. A.17(a)). Additionally, in the laboratory experiment I do not know the upper boundary condition for the concentration. Therefore, I took the upper ten pixel lines of the measured concentration field and set these pixel lines as the upper boundary condition before every warping step. Using these parameters for the concentration field propagation, I received the estimated concentration fields as shown in Fig. 6.9(d)-(f) together with the respective measured concentration fields in Fig. 6.9(a)-(c). The propagation times are $\tilde{t}_\mathrm{p} = 0.0438$, $\tilde{t}_\mathrm{p} = 0.0876$, and $\tilde{t}_\mathrm{p} = 0.219$, respectively. To the eye the concentration fields at $\tilde{t}_\mathrm{p} = 0.0438$ do agree in general, although slight deviations at the leftmost and rightmost density fingers already become apparent. At $\tilde{t}_\mathrm{p} = 0.0876$ and $\tilde{t}_\mathrm{p} = 0.219$ these deviations become even larger until the concentration fields do not agree very well anymore. In the center region the agreement in the shape is still very good despite the appearance of the slightly pointier finger tips. Given the evaporation boundary condition and the balancing upward flow in the laboratory experiment together with the observed strong lateral movement of the outer density finger seeding points (cf. Fig. 4.3), I assume that the outermost fingers are strongly affected by the background flow. As indicated by the blue dotted lines in Fig. 6.9, I divide the observed area into outer regions, where I expect the influence to be strong, and a center region with small, but still noticeable influence. In TrainNE1, only the effects of pure active solute transport are represented, the outer regions are coherently beyond the capabilities of the flow field estimation in this case. Hence, I focus on the discussion of the center region for the remainder of this section.

To be able to compare the quality of the results on real and synthetic data, I also used concentration field propagation as described in Section 6.2 on the synthetic data to obtain a reference. The upper boundary condition for the concentration is set to $\tilde{C}_w = 1$, resembling the boundary condition in the numerical simulations. I chose one

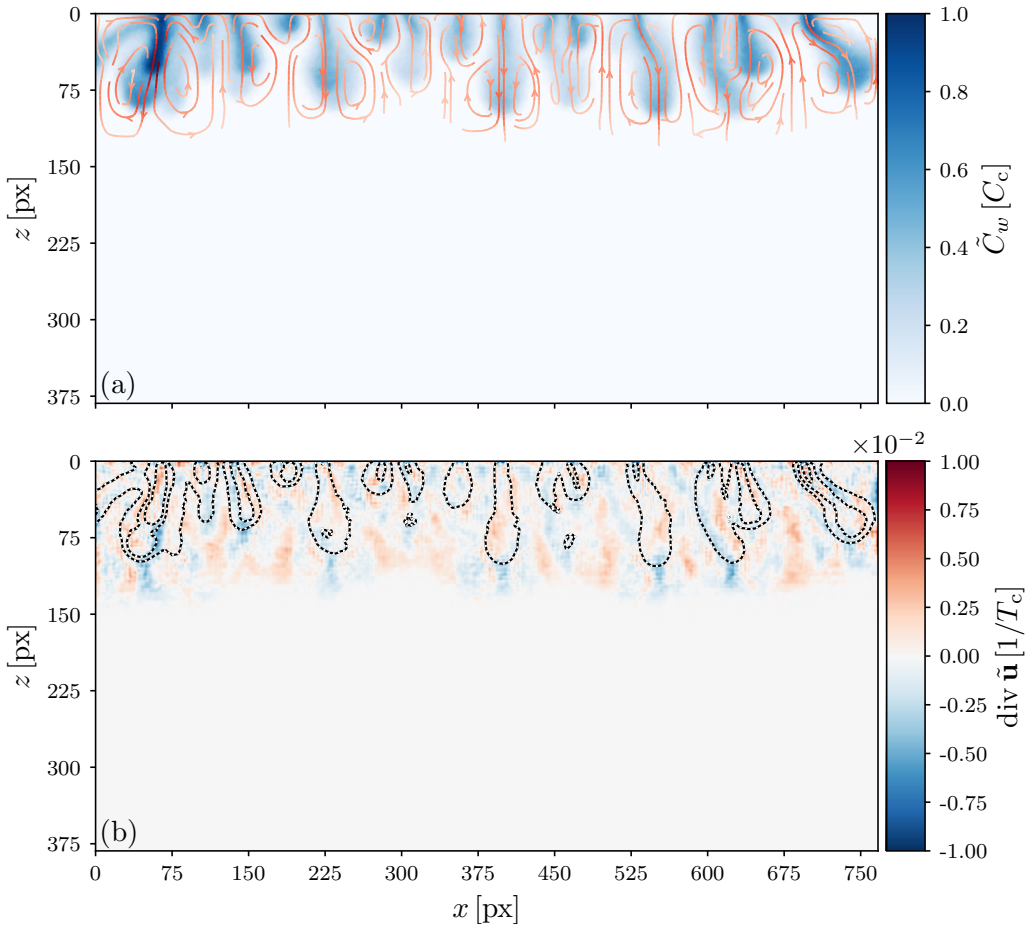

Figure 6.8: Estimated flow field for the laboratory experiment (a) shown as streamlines (red color intensity indicates absolute flow velocity) on top of the color coded prior concentration field. Flow field divergence of the estimated flow field (b). Concentration isolines are given at levels $\tilde{C}_w = (0.25, 0.5, 0.75)$.

distinct concentration field with the propagation time $\tilde{t}_\mathrm{p} = 0$ as the starting point. The concentration field is then iteratively propagated by the propagation time step $\mathrm{d}\tilde{t} = 0.02$, since the true concentration field is present in this temporal resolution. For the chosen concentration field, the given propagation time step results in $\sigma = 2.43\,\mathrm{px}$ for the Gaussian blurring. The examples of the synthetic reference, as presented in Fig. A.12, are chosen for the propagation times $\tilde{t}_\mathrm{p} = 0.04$, $\tilde{t}_\mathrm{p} = 0.08$, and $\tilde{t}_\mathrm{p} = 0.22$ to closely match the propagation times for the laboratory experiments as shown in Fig. 6.9.

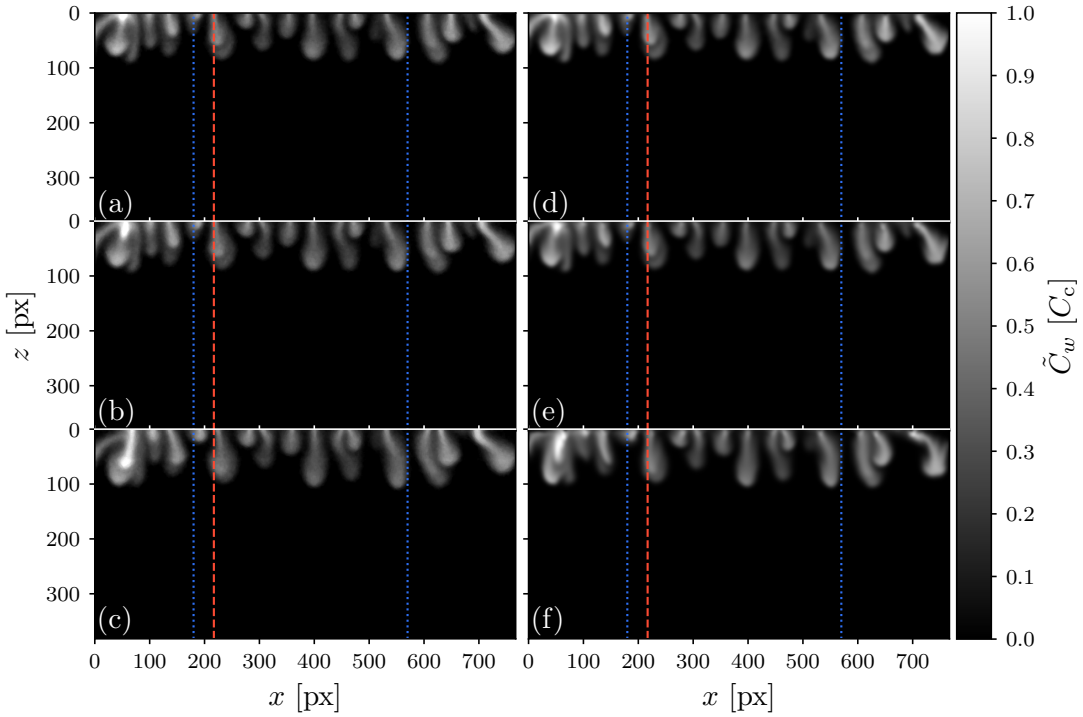

Figure 6.9: Concentration field propagation for the laboratory experiment: True concentration fields (a)-(c) and concentration fields propagated with estimated flow fields (d)-(f) at propagation times $\tilde{t}_\mathrm{p} = 0.0438$ (top row), $\tilde{t}_\mathrm{p} = 0.0876$ (center row), and $\tilde{t}_\mathrm{p} = 0.219$ (bottom row). The red dashed line marks the initial position of the respective density finger seeding point. The blue dotted lines indicate the region of concentration error evaluation presented in Figs. 6.10(d)-(f).

For the synthetic reference, a detailed look at the deviations between the concentration fields that are propagated with the estimated flow fields and the true concentration fields is presented in Fig. 6.10(a)-(c) as the normalized concentration errors $([\tilde{C}_w^\mathrm{est} - \tilde{C}_w^\mathrm{true}]/\max(\tilde{C}_w^\mathrm{true}))$ for the same propagation times $\tilde{t}_\mathrm{p} = 0.04$, $\tilde{t}_\mathrm{p} = 0.08$, and $\tilde{t}_\mathrm{p} = 0.22$, respectively. At $\tilde{t}_\mathrm{p} = 0.04$, the deviations in the middle region, where the flow velocities are relatively small, roughly range from -1.5% up to 1.5%. In regions with higher velocities on the left, the concentration field deviates stronger. Especially at the tip of the furthest developed density finger (at $x = 50$) the normalized concentration error is -3%. This is consistent with the already observed underestimation of the absolute flow velocities (cf. Figs. 6.6(b) and 6.7(b)). The solute there is not transported as far as in the true concentration field. For the subsequent propagation times the estimated and the true concentration fields diverge even further, also as expected due to the underestimation of the flow fields.

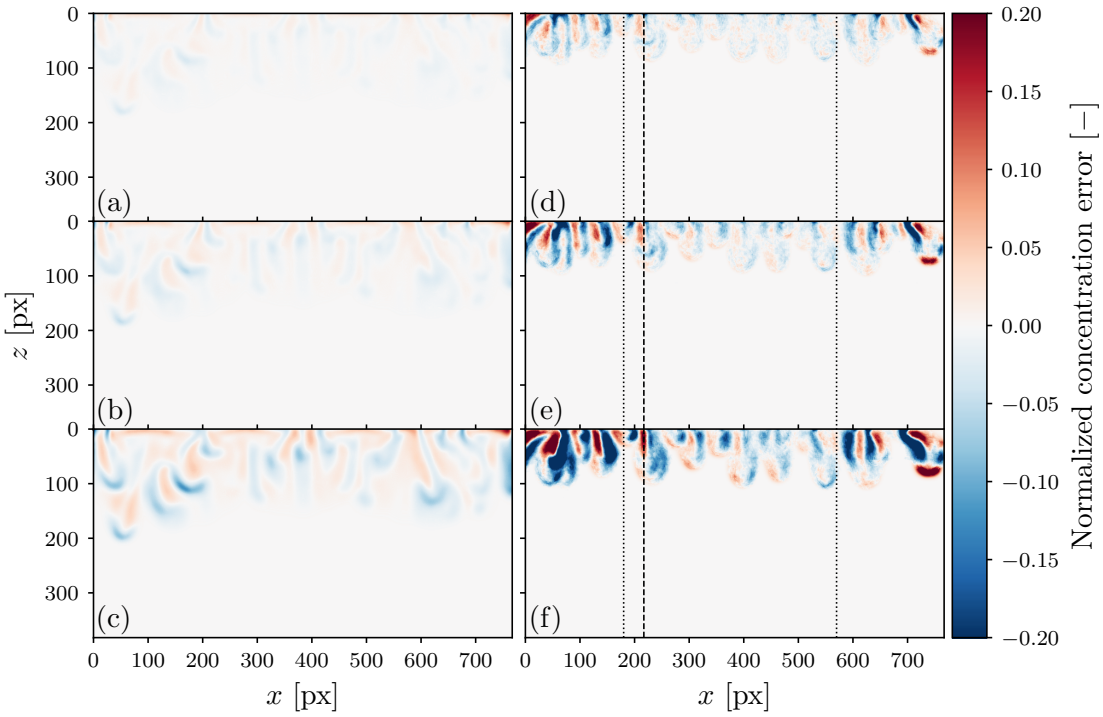

Figure 6.10: Normalized concentration errors for propagated concentration fields. Numerical experiment: (a) at $\tilde{t}_\mathrm{p} = 0.04$, (b) at $\tilde{t}_\mathrm{p} = 0.08$, and (c) at $\tilde{t}_\mathrm{p} = 0.22$. Laboratory experiment: (d) at $\mathrm{d}\tilde{t} = 0.0438$, (e) at $\tilde{t}_\mathrm{p} = 0.0876$, and (f) at $\tilde{t}_\mathrm{p} = 0.219$. Normalized concentration errors are chosen to be negative for $\tilde{C}_w^\mathrm{est} < \tilde{C}_w^\mathrm{true}$. The black dashed line indicates the initial position of the respective density finger seeding point (cf. red dashed line in Fig. 6.9). The black dotted lines indicate the center region with small influence of the background flow.

For the laboratory experiment, Figs. 6.10(d)-(f) show the normalized concentration errors at $\tilde{t}_\mathrm{p} = 0.0438$, $\tilde{t}_\mathrm{p} = 0.0876$, and $\tilde{t}_\mathrm{p} = 0.219$, respectively. At $\tilde{t}_\mathrm{p} = 0.0438$, the normalized concentration error is generally low, with values roughly ranging from -10% to 10% for the center region, where the influence of the background flow is small (black dotted lines). For the longer propagation times the errors increase, as the estimated concentration field diverges more and more from the measured one. When compared to the normalized concentration errors of the numerical experiment (Figs. 6.10(a)-(c)), the errors are typically larger for the laboratory experiment. Nevertheless, in regions with higher flow velocities, as seen in the region with $x < 200\,\mathrm{px}$ for the numerical experiment, errors can reach similar values as in the center region of the laboratory experiment. The qualitative structure of the errors differs, however. For the numerical experiment the concentrations are underestimated in front of the

density finger tips, whereas for the laboratory experiment the concentrations are underestimated on the right flank of the fingers, while being overestimated on left flank. This shows that for the numerical experiment the downward flow is too slow, whereas for the laboratory experiment the estimation is unable to predict a general rightward flow. As the rightward drift there seems to affect all the fingers quite uniformly, I attribute this to the earlier mentioned convection currents in the Hele-Shaw cell independent of the density-driven flow. This is also observed for fingers in the central region, as is seen in Figs. 6.9(a)-(c), where not only the seeding point of the density finger, initially located at the red dashed line, but the complete finger shifts to the right. The corresponding position in Figs 6.10(d)-(f) is indicated by the black dashed line. This kind of background convection is not represented in TrainNE1, hence, it is coherent that the method cannot estimate the flow fields in this respect.

6.5 Representation of Superscale Convection

The results presented in Section 6.3 show that the flow fields are estimated reliably for the synthetic test case using Architecture 1 trained on TrainNE1 and can generalize to variations in the concentration boundary condition, while the quantitative investigation showed a general underestimation of the absolute flow velocities. Applying the same CNN to the real test case (Section 6.4) showed that in some regions the estimated flow fields can describe the observed concentration fields reasonably well. However, in other regions the estimated flow fields qualitatively do not agree with the observations, revealing that important transport processes are not represented in TrainNE1. In this section, I present the results when using TrainNE1+3 as training data to include the representation of superscale convection.

As Architecture 1 showed improved, but still unsatisfactory estimations of the flow fields for the real world test case (results shown in Fig. A.15), I used Architecture 2 in this case. The idea is that the skip connections in Architecture 2 aid to propagate fine graded information through the network, which is important to account for the background flow. In addition, the larger capacity of the model can facilitate the incorporation of more transport processes. The results of the flow field estimation using Architecture 2 trained on TrainNE1+3 are presented for the synthetic test case first (Section 6.5.1), followed by the results on the laboratory test case (Section 6.5.2).

6.5.1 Results on the Numerical Experiments

To follow the same line as in Section 6.3, I chose a representative concentration field pair example from ValidateNE1+3, along with the already introduced example from TestNE2, to present the results with the represented background flow. Using Architecture 2 trained on TrainNE1+3 resulted in the estimated flow fields as depicted in Figs. 6.11 and 6.12 for the ValidateNE1+3 and TestNE2 examples, respectively.

The ValidateNE1+3 example exhibits strong background flow in the negative x-direction that is superposed to the flow structures of the density-driven instability. As

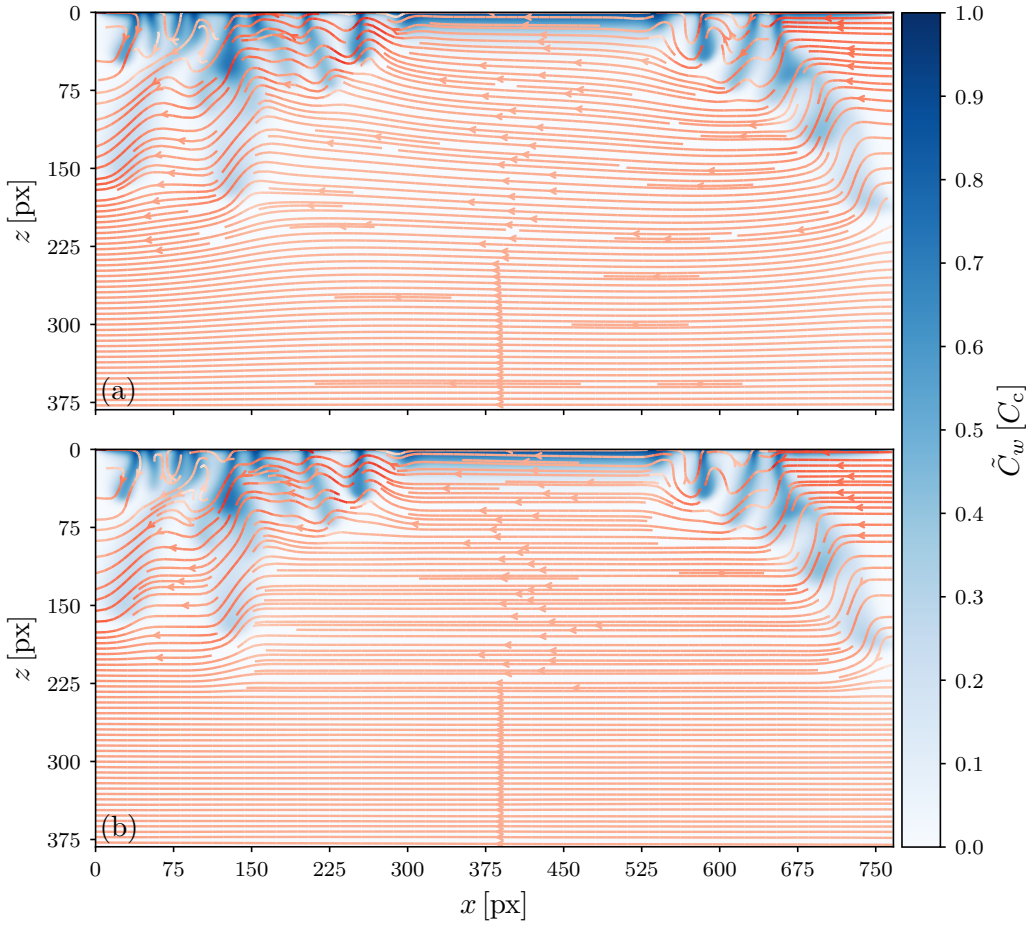

Figure 6.11: Estimated flow field for the ValidateNE1+3 example with represented superscale convection: truth (a) and estimation (b) of the flow field shown as streamlines (red color intensity indicates absolute flow velocity) on top of the color coded prior concentration field.

depicted in Fig. 6.11(a) for the true flow field, areas of the superposition are located in the outer regions of the upper half of the domain. There, the streamlines show wave like patterns that have an upward component in between the fingers and a downward component where the solute induces downward movement along the fingers. Right in the upper left corner, a quite interesting pattern is observed, as parts of the streamlines are directed opposite to the background flow direction. No fingers have developed in the center of the domain, hence, the flow there is horizontal at the top. Below, the flow exhibits a slight upward component for the upper half, but is again perfectly horizontal for the lower half of the domain.

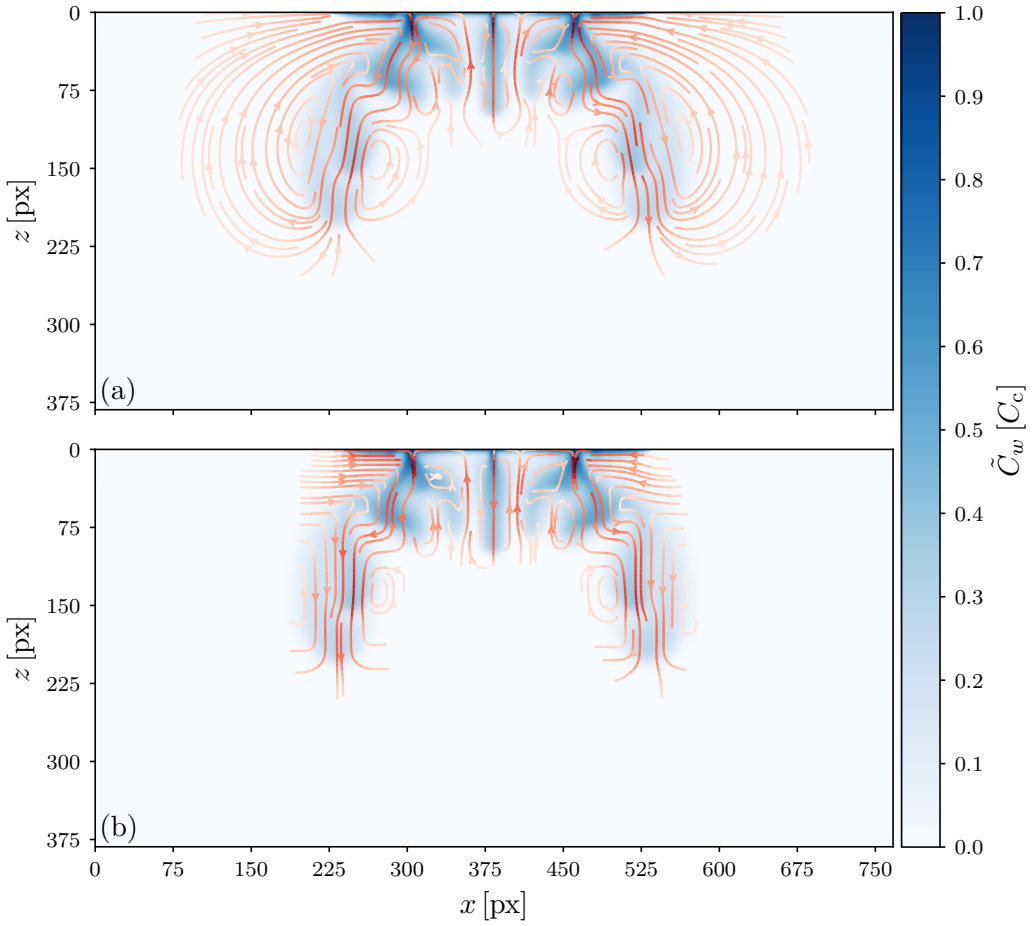

Figure 6.12: Estimated flow field for the TestNE2 example with represented superscale convection. Same representation as in Fig. 6.11.

The estimated flow field, as depicted in Fig. 6.11(b), is in good agreement with the true flow field. The same main flow direction to the left is predicted reliably, which is quite remarkable since for large parts of the domain no information by solute movement is given. In the outer regions, the same wave like patterns in the streamlines are observed and even the earlier mentioned flow in the opposite direction to the background flow in the upper left corner is predicted correctly. The main difference that is evident in the estimation is the absence of an small upward flow component in the upper half of the domain center.

The flow field estimation for the TestNE2 example is shown in Fig. 6.12(b), along with the true field in Fig. 6.12(a) for comparison. The estimation correctly detects the absence of the background flow in TestNE2, while in the regions of the density fingers, the shape of the depicted streamlines qualitatively still agrees with the truth.

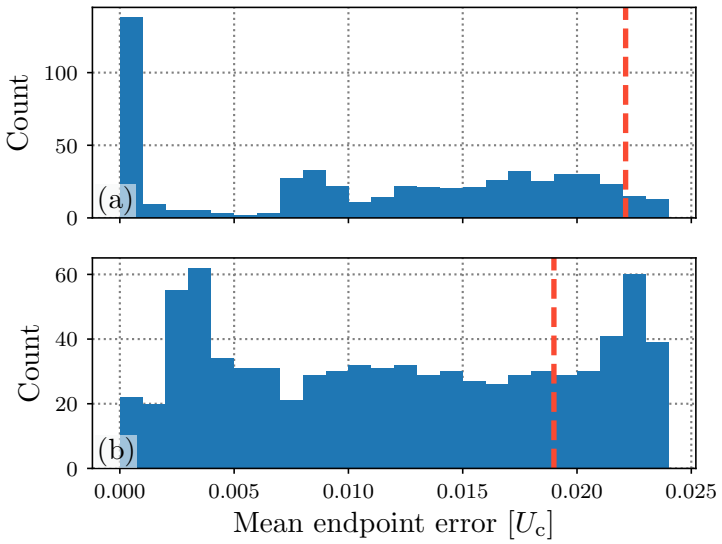

Figure 6.13: Error distributions of the mean endpoint error for ValidateNE1+3 (a) and TestNE2 (b) with represented superscale convection. Mean endpoint error values of the representative dataset examples, shown in Figs. 6.11 and 6.12, respectively, are indicated by the red dashed lines.

The flow points downward along the fingers, upward in small regions in between, and in some places the already observed circular patterns are apparent. When moving away from the fingers, the estimated flow velocities quickly drop to very small values close to 0. This is different to the estimated flow fields of Architecture 1 trained on TrainNE1 (cf. Fig. 6.3), where the estimation better described the true flow field in the surroundings of the fingers. Nonetheless, Architecture 2 trained on TrainNE1+3 still generalizes reasonably well for the qualitatively different concentration boundary condition in TestNE2.

For both, the ValidateNE1+3 and the TestNE2 examples, the investigation of the flow field divergences showed similar values as in Section 6.3 (cf. Figs. A.13 and A.14). Therefore, the introduced errors with respect to spurious fluid sources and sinks are comparable to the estimations using Architecture 1 trained on TrainNE1.

As a quantitative measure, I again calculated the MEPE for the estimated flow fields of the individual image pairs of ValidateNE1+3 and TestNE2. The respective error distributions are shown in Fig. 6.13(a) and (b). For ValidateNE1+3, the histogram shows the maximum value of 0.024, which is slightly higher than the observed 0.022 for ValidateNE1 without the representation of superscale convection (cf. Fig. 6.5(a)), but still suggests a good overall agreement between the estimation and the truth for this case. The same situation with a main part of the data count being below MEPE=0.001,

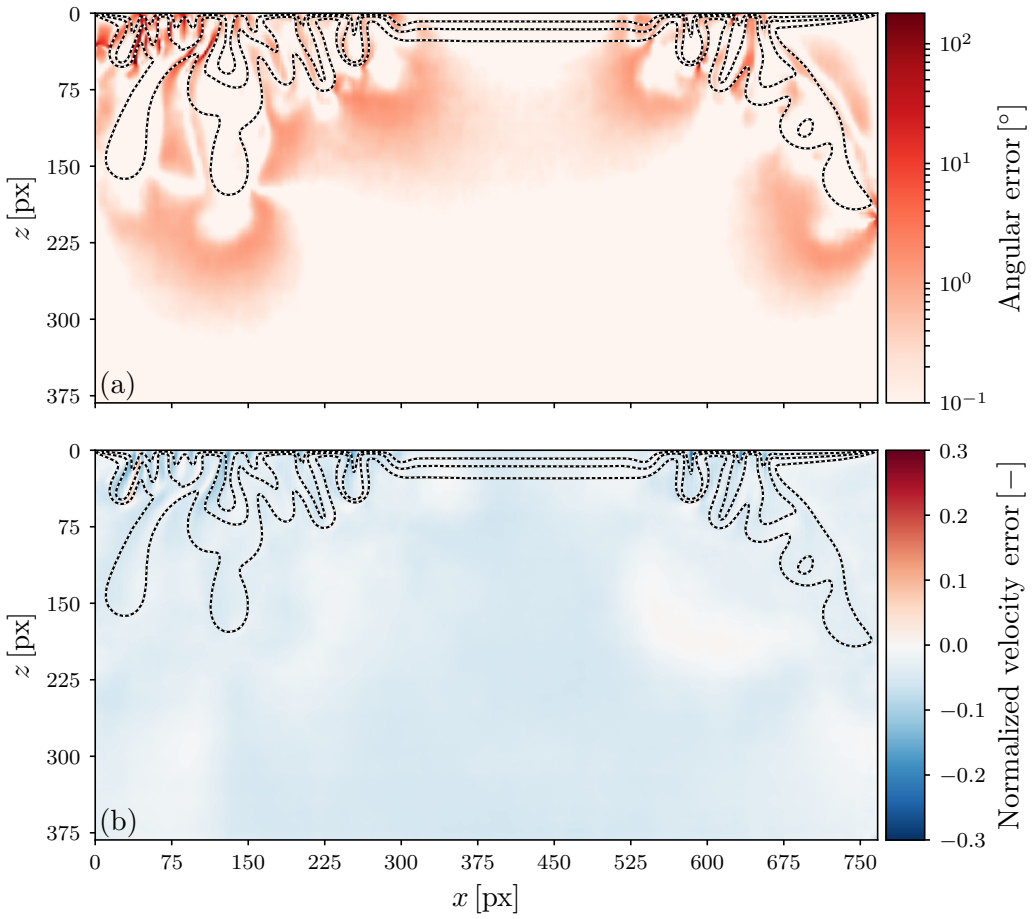

Figure 6.14: Error measures for the ValidateNE1+3 example with represented superscale convection: Angular error (a) and normalized velocity error (b) (negative for $|\tilde{\mathbf{u}}^{\text{est}}| < |\tilde{\mathbf{u}}^{\text{true}}|$). Concentration isolines are given for the levels $\tilde{C}_w = (0.15, 0.4, 0.65)$. Note the logarithmic scale for better visibility of small angular error values.

due to the little dynamics in these image pairs, is encountered as well. The error distribution for TestNE2, as depicted in Fig. 6.13(b), also shows higher MEPE values, when compared to the results for Architecture 1 trained on TrainNE1 (cf. Fig. 6.5(b)), with the maximum being at 0.024. This is expected when recalling the estimation, as shown in Fig. 6.12. There, the estimation of the correct flow field fails in the closer surroundings of the density fingers, while Architecture 1 trained on TrainNE1 is able to better extrapolate in these regions. Note that also the MEPE of the ValidateNE1+3

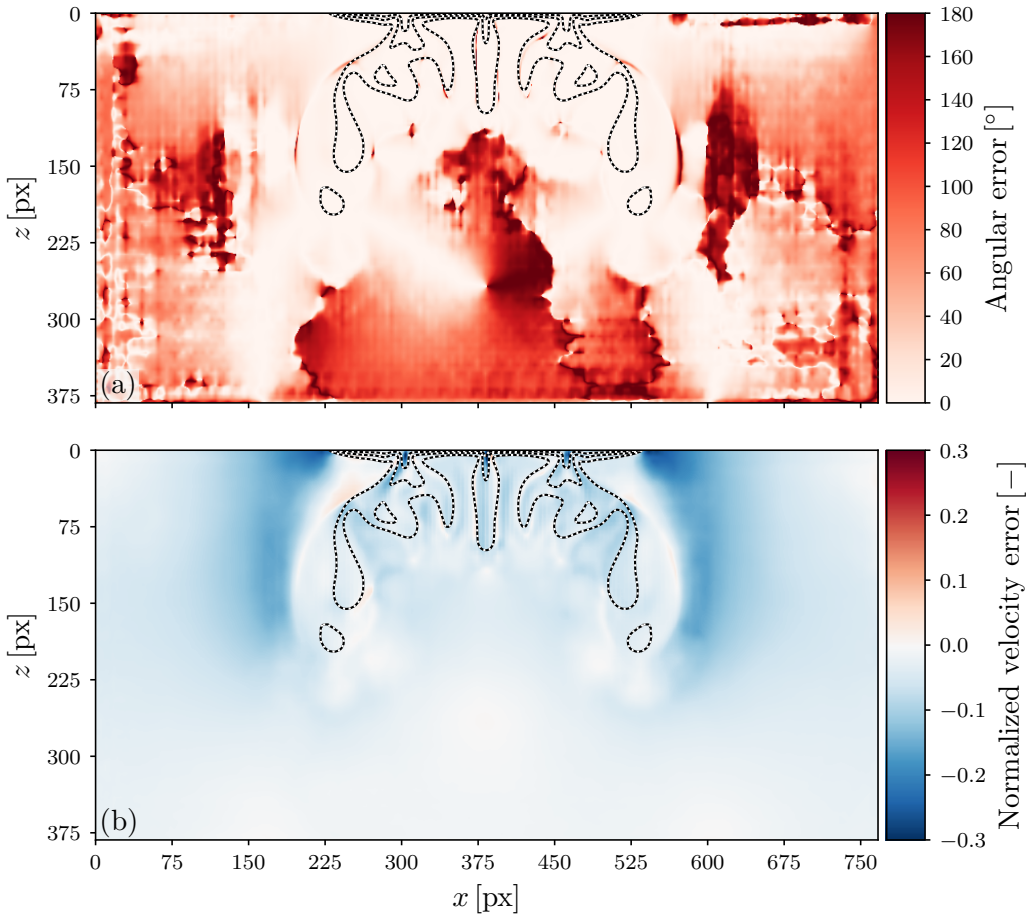

Figure 6.15: Error measures for the TestNE2 example with represented superscale convection: Angular error (a) and normalized velocity error (b) (negative for $|\tilde{\mathbf{u}}^{\text{est}}| < |\tilde{\mathbf{u}}^{\text{true}}|$). Concentration isolines are given for the levels $\tilde{C}_w = (0.25, 0.5, 0.75)$.

and TestNE2 examples are indicated in Fig. 6.13, which supports their choice as representative examples, in the sense that the estimation achieves a lower value on the majority of the image pairs as for the given examples.

For the assessment of the spatial distribution, the angular error and the normalized velocity error for the representative examples are depicted in Figs. 6.14 and 6.15. Their investigation supports the observations already deduced from Figs. 6.11 and 6.12. For the ValidateNE1+3 example, the agreement of the flow direction is reasonably good for the most part. The main deviations are located in the upper left corner, where relatively fine fingers are apparent. Note that in the presentation a logarithmic scale is chosen to improve the visibility of the low values. As already observed in Section 6.3

for Architecture 1 trained on TrainNE1, Architecture 2 also mainly underestimates the absolute flow velocities, which can be seen in Fig. 6.14(b). The observed values for the normalized velocity error are at around -5% in regions where only background flow is present. The same is true for large parts in the region of the density fingers, but values as low as -22.1% are encountered at their seeding points, which is coherent with the errors presented in Section 6.3.

The angular error distribution for the TestNE2 example is shown in Fig. 6.15(a). In this case, the flow direction is still predicted reasonably well where $\tilde{C}_w > 0$. Outside of these regions, the predicted directions generally do not agree anymore, as the angular error ranges up to 180° there. This is coherent with the composition of TrainNE1+3, since parts contain background flow, while others do not. No explicit information by lateral movement of whole density fingers is apparent in the TestNE2 example, hence the estimation of the direction seems to be ambiguous in this case. The normalized velocity errors for the TestNE2 example, as depicted in Fig. 6.15(b), are in agreement with the previous observations that the flow velocities are generally underestimated. For the most part the errors are contained in a similar range as before, but additional areas of strong underestimated flow velocities, which are already indicated in Fig. 6.12, become apparent. At the outsides of the leftmost and rightmost large plumes, where $\tilde{C}_w$ is close to 0, the encountered values range between -10% and -26%.

In summary, the application of the flow field estimation on the synthetic test case showed that the background flow can be predicted reliably, when the process is represented in the training data. The predictive capability for the TestNE2 example decreased, but remained reasonably good in regions, where information is given by the density fingers and plumes. As encountered before in Section 6.3, the limitations of the method to accurately estimate the absolute flow velocities persist with the same tendency to underestimate the values.

6.5.2 Results on the Laboratory Experiment

The flow field estimation on the laboratory experiment example using Architecture 2 trained on TrainNE1+3 is presented in Fig. 6.16. With the represented superscale convection in the training data, the estimated flow field is qualitatively different from the estimation using Architecture 1 trained on TrainNE1 (cf. Fig. 6.8). The flow field can be divided in three regions. The first ranges from $x = 0\,\text{px}$ to $450\,\text{px}$, where a strong background flow in the positive x-direction is predicted. In the upper part of this region, the background flow is superposed to the density-driven instability and the same wave like patterns in the streamlines as for the synthetic case do occur. Between $x = 450\,\text{px}$ and $600\,\text{px}$, the background flow is generally weaker and shifts from rightward to leftward. In the upper part, the flow seems to be hardly affected by this background flow, since the patterns of the streamlines resemble the patterns observed for the case without the representation of the superscale convection. For x between $600\,\text{px}$ and $768\,\text{px}$, the predicted background flow points in the negative

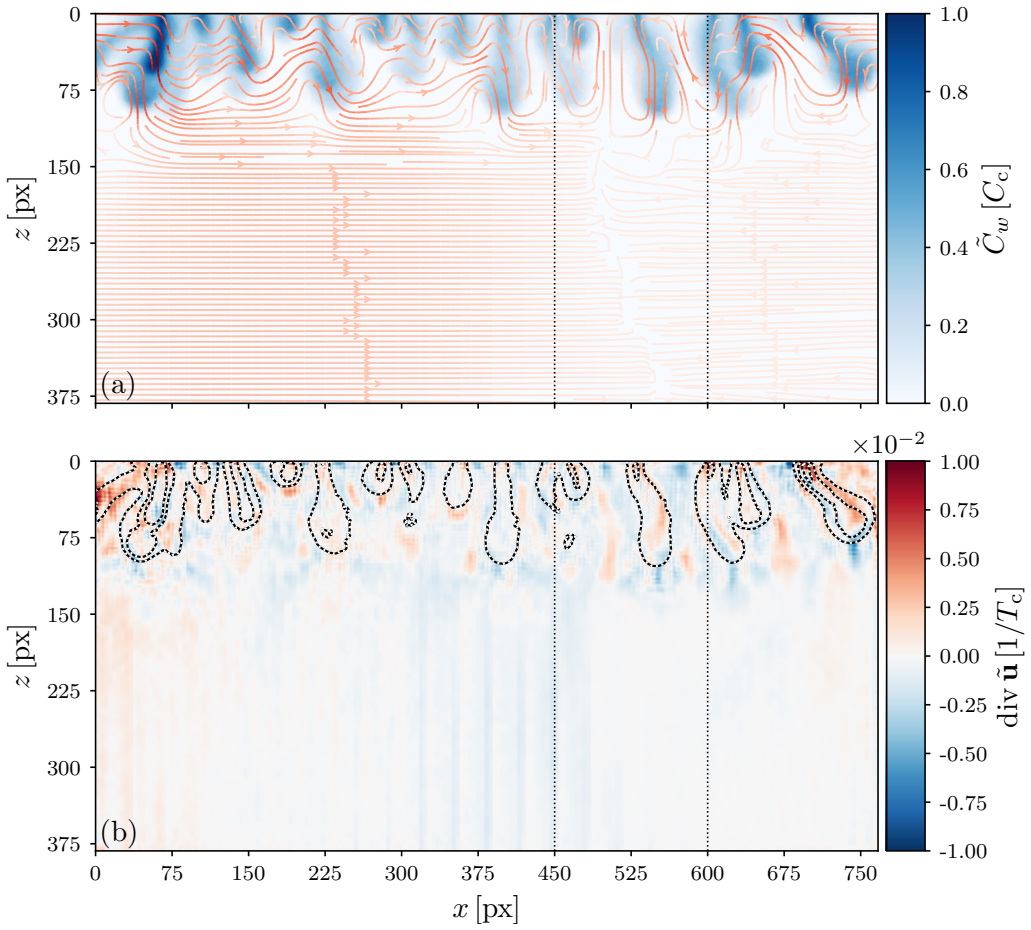

Figure 6.16: Estimated flow field for the laboratory experiment with represented superscale convection (a) shown as streamlines (red color intensity indicates absolute flow velocity) on top of the color coded prior concentration field. Flow field divergence of the estimated flow field (b). Concentration isolines are given at levels $\tilde{C}_w = (0.25, 0.5, 0.75)$. The black dotted lines indicate regions with different influence of the background flow.

x-direction and is weaker in comparison to the flow in the first region. There again, in the upper part, the wave like patterns in the streamlines are encountered, but the upward and downward components are more pronounced. Overall, the predictions are coherent with the observed lateral movement of the fingers in the temporal development of the laboratory experiment as presented in Section 4.2. This already indicates a more consistent estimation of the flow field in this case.

The corresponding flow field divergence values are shown in Fig. 6.16(b). In the region of the density fingers, the values are generally similar to the case without the

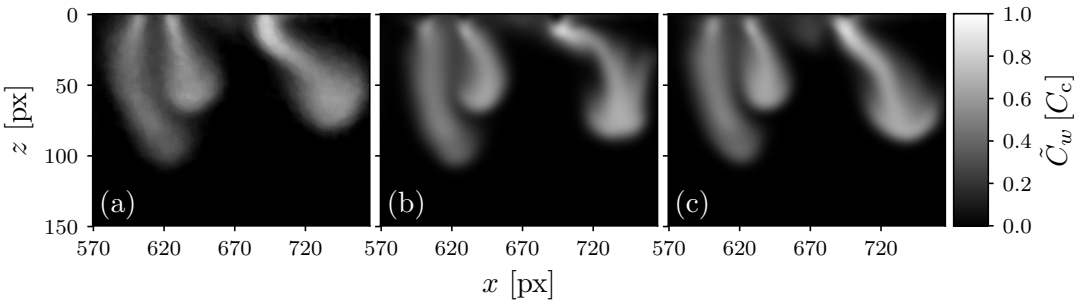

Figure 6.17: Detail of the propagated concentration fields for the three rightmost density fingers: the true concentration field (a) compared to the predicted concentration fields after five propagation steps for Architecture 1 trained on TrainNE1 (b) and for Architecture 2 trained on TrainNE1+3 (c).

representation of superscale convection. This is true, except for small areas at the outermost fingers, where values up to $1.3 \cdot 10^{-2}$ are found. Hence, for the most of the upper part the errors introduced in this respect are expected to be similar as before. As a qualitative difference, values between $-1.3 \cdot 10^{-3}$ and $1.0 \cdot 10^{-3}$ are found in the lower part, where the flow field divergence was 0 in the previous case. When going from left to right, this is due to the gradual increase and following gradual decrease of the parallel flow leading to the spurious fluid sources and sinks. However, the values encountered there are similar to the divergences in the upper part and the introduced errors are therefore similar.

Also for the flow field estimation with Architecture 2 trained on TrainNE1+3, I propagated the measured concentration fields using the estimated flow field as described in Section 6.2. The optimized values resulted in $\mathrm{d}\tilde{t} = 0.0336$ and $\sigma = 2.2\,\mathrm{px}$ for the dimensionless time step and the standard deviation of the Gaussian image filter, respectively (results are illustrated in Fig. A.17(b)). The expectation is that with the represented superscale convection the estimated flow fields better explain the measured concentration fields. A comparison is presented in Fig. 6.17, where the measured concentrations of the three rightmost fingers in the observation area are shown along with the propagated concentration fields with and without the represented superscale convection. Most prominently, the shape of the density finger on the right qualitatively agrees significantly better when using Architecture 2 trained on TrainNE1+3 for the estimation. Also the other two fingers better resemble the measured ones in their shape, but the improvements are more subtle there.

For a quantitative insight into the improvements, I present the comparison of the normalized concentration errors with and without the represented superscale convection in Fig. 6.18. The concentration fields are propagated using the respective optimized time steps and standard deviations for Gaussian image filtering for this comparison. Therefore, I used $\mathrm{d}\tilde{t} = 0.0438$ and $\sigma = 2.6\,\mathrm{px}$ for the case of Architecture 1 trained on TrainNE1 and $\mathrm{d}\tilde{t} = 0.0336$ and $\sigma = 2.2\,\mathrm{px}$ for the case of Architecture 2 trained on

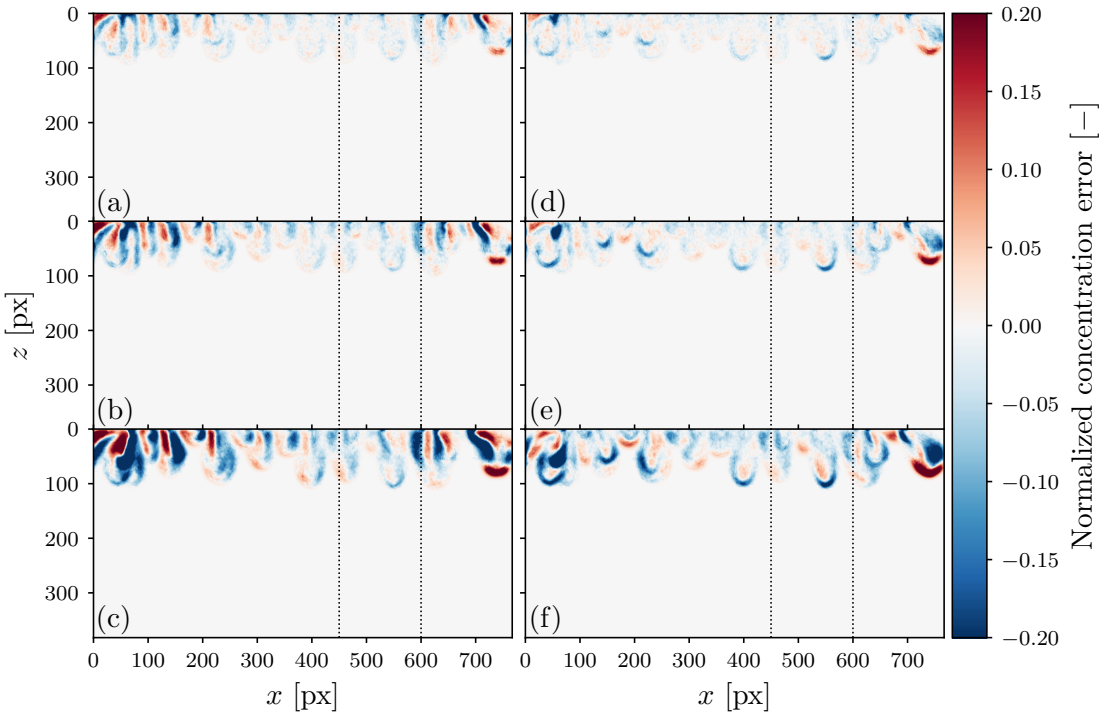

Figure 6.18: Comparison of the normalized concentration errors for the cases with and without represented superscale convection: Results for Architecture 1 trained on TrainNE1 ((a) to (c)) and Architecture 2 trained on TrainNE1+3 ((d) to (f)), both after one (top row), two (middle row), and five (bottom row) propagation steps using their respective optimized time steps $\mathrm{d}\tilde{t}$. Normalized concentration errors are chosen to be negative for $\tilde{C}_w^{\mathrm{est}} < \tilde{C}_w^{\mathrm{true}}$. The black dotted lines indicate regions with different influence of the background flow.

TrainNE1+3. Again, the three regions with different influence of the background flow, as described above, become recognizable.

In the region for $0\,\mathrm{px} < x < 450\,\mathrm{px}$ with a strong background flow to the right (cf. Fig. 6.16(a)), normalized concentration errors as low as -46% and as high as 17% already become noticeable after one propagation step, when the superscale convection is not represented in the training data. For the subsequent propagation steps the errors increase significantly with a typical deviation structure that the concentrations are too low on the right and too high on the left of the fingers. This is due to the lateral movement of the fingers that is apparent in the measurements, but not represented in the estimated flow fields. In contrast, for the case with the represented background flow in the training data, the errors are generally smaller, with few exceptions, and also qualitatively different. Since the errors at the individual fingers are approximately symmetric in the lateral direction, the estimated flow fields better represent the lateral

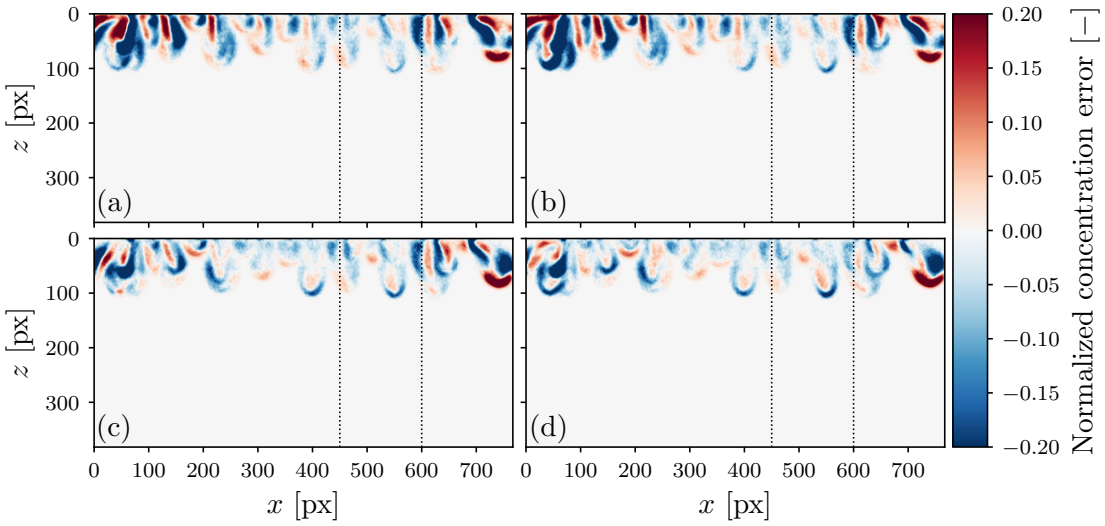

Figure 6.19: Comparison of the effects due to CNN architecture and represented superscale convection in the training data: normalized concentration errors after five propagation steps for Architecture 1 trained on TrainNE1 (a), Architecture 2 trained on TrainNE1 (b), Architecture 1 on TrainNE1+3 (c), and Architecture 2 trained on TrainNE1+3 (d).

movement of the fingers. Also, the concentrations are typically underestimated at the finger tips and overestimated at their cores, which is coherent with the already encountered tendency of the method to underestimate the absolute flow velocities. Despite the reasonable qualitative agreement with the measurements, still substantial errors are encountered, especially, at the leftmost fingers after five propagation steps. It seems that there the affect of the superscale convection is stronger than the estimation is able to detect.

In the region for $450\,\text{px} < x < 600\,\text{px}$, where the influence of the background flow is weak, the errors are generally smaller for the case where the background flow is not represented in the training data. This can be explained, as the relevant process is represented reasonably good in both cases, but the larger optimized time step for Architecture 1 trained on TrainNE1 compensates for the typical underestimation of the absolute flow velocities.

In the last region for $600\,\text{px} < x < 768\,\text{px}$, the background flow is directed to the left and again affects the movement of the density fingers. Apart from the fact that the flow is in the opposite direction compared to the first region, the observations are the same.

Overall, the following conclusions can be drawn from this. The flow field estimation with Architecture 2 trained on TrainNE1+3 better describes the lateral movement of the density fingers and is therefore more consistent with the measurements. While the optimization of the time step in the case of Architecture 1 trained on TrainNE1 allows

to compensate for the underestimated absolute flow velocities, the lateral flow in the case of represented superscale convection in the training data introduces an additional constraint for the optimization of the time step, such that the lateral position of the fingers has to match the measurements. With this, I argue that the optimized time step $\mathrm{d}\tilde{t} = 0.0336$ is more consistent with the measurements.

To get an assessment of the effect on the results due to the differences between Architecture 1 and Architecture 2, namely the introduction of skip connections to allow for fine grained information to propagate through the network and the larger model capacity, a comparison of the normalized concentration errors between the individual cases is shown in Fig. 6.19. From the left to the right column, Architecture 1 is swapped for Architecture 2 and from the top to the bottom row, superscale convection is included in the training data. When trained on TrainNE1, the normalized concentration errors for Architecture 1 and Architecture 2 are very similar, therefore, no significant effect of the architecture differences are encountered in this case. The smaller model capacity of Architecture 1 already seems to be large enough to incorporate the relevant processes and the skip connections in Architecture 2 seem to be irrelevant for the detection of the solute movement due to the density-driven flow. When introducing the representation of background flow the results also improve for Architecture 1, in the sense that the lateral movement is generally better estimated by the flow fields. Still, the lateral position is off for some density fingers in the outermost regions. The overall best results with respect to a consistent representation of the relevant physical processes are achieved with Architecture 2 trained on TrainNE1+3, although underestimated absolute flow velocities introduce underestimated concentrations at the finger tips in this case. This suggests that skip connections are important to better identify the small lateral movements, along with the larger model capacity to be capable to represent the associated background flow.

6.6 Summary & Discussion

I estimated the flow fields for a synthetic and a real world test case by using two versions, a simplified and an extended one, of the same CNN that I trained on purely synthetic data of active solute transport and showed the capabilities and limitations of the approach. In doing so, I successfully transferred the flow field information, contained in the numerical simulations, over to the laboratory experiment.

First, I used the simpler of the two CNN versions, Architecture 1, trained on the data that only account for the process of the density-driven instability (TrainNE1). For the synthetic test case (TestNE2), I chose a representative example to present the results. For this test case, the method was able to reliably predict the local flow directions, even in surrounding areas of the density fingers, where no direct information was given by solute movement. This is quite remarkable, since the concentration boundary condition in test case was chosen to be different from the boundary condition in the training data. Furthermore, this demonstrates that the CNN was able to learn

the representation of the physical process and transfer the information to data that exhibits some qualitative differences to the data it has encountered during the training process. However, limitations of the method to accurately estimate the absolute flow velocities were encountered. The CNN tends to underestimate the true values. In typical applications, like image classification and image segmentation, for which CNNs originally were developed, high contrast images are encountered. This is different to the characteristics of the concentration fields observed for active solute transport. The smooth concentration gradients in conjunction with small displacements seem to impede the quantitative correct detection of the movement and an improved adaptation of the method to the physical problem is needed at this point.

In a next step, the same version of the CNN was applied to the laboratory experiment and the flow fields were estimated successfully. This demonstrated exemplarily the information transfer to real data. The qualitative flow structures were consistent with the structures in the synthetic data. To investigate the correctness of the estimation, I propagated the concentration fields forward in time, using the displacement by the estimated flow fields and Gaussian image filtering, and compared the resulting concentration fields to the measured ones. To obtain the displacement, the flow fields needed to be scaled with the according time step. Calculation of the time step by using the characteristics of the experiment resulted in a value of $\mathrm{d}\tilde{t} = 0.034 \pm 0.07$ with a large uncertainty and which is also expected to introduce biases that cannot be assessed. Therefore, I chose to optimize the time step such that the root mean squared error between the propagated concentration fields and the measured ones were minimized. With this, I obtained a value of $\mathrm{d}\tilde{t} = 0.0438$, which is substantially larger than the experimentally predicted one. This implicitly scaled the flow fields and therefore alleviated the suspected and previously in the synthetic test case observed underestimation of the absolute flow velocities. However, I argue that the incorporation of this information is valid, since the interest is in the best estimate of the experimentally inaccessible flow fields in between two subsequent time steps. This is conceptually different from using the CNN as a surrogate forward model to predict the next unknown time steps.

The results on the propagated concentration fields of the laboratory experiment show that the estimated flow fields explain the measured concentration fields reasonably well in the center region of the observation area. Comparison to the synthetic reference showed that the encountered error values can be similar there, but are qualitatively different in their spatial distribution. According to the underestimated absolute flow velocities, the propagated concentrations were underestimated at the finger tips in the synthetic reference, while in the laboratory case, the concentrations were underestimated at the right side of the fingers and overestimated at their left sides. This is attributed to a lateral drift of whole density fingers, which was observed in the laboratory experiment, but was not represented in the training data, in this case. The errors due to the lateral drift of density fingers were substantially larger in the outer regions, where the flow field estimation failed to explain the measured concentration fields. This indicated the missing representation of relevant transport processes and

shows that the CNN, despite its capability to generalize to different situations, is only able to learn the physics presented in the training data.

To account for superscale convection in the synthetic training data, I introduced background flow by temporally altering the potential at the lateral boundary conditions in the numerical simulation. The extended version of the CNN, Architecture 2, incorporates more convolution layers and introduces skip connections to allow signal propagation of fine grained information through the network. I then trained this version of the CNN on the extended training data (TrainNE1+3) so that it learned the representation of the density-driven solute transport along with the superscale convection. In this case, the method was able to reliably distinguish between the presence and absence of background flow in the synthetic test data, while maintaining the predictive capability of the flow due to the density differences, where information was given by the solute. The method still showed the tendency to underestimate the absolute flow velocities, as for Architecture 1 trained on the data without represented superscale convection. As a difference, Architecture 2 trained on the extended training data lost some of the predictive capability in the surroundings of the density fingers for the synthetic test example. However, this seems to be attributed to the characteristics of the training data used in this case, since the flow direction in the bulk without solute is ambiguous if no explicit information by lateral movement of whole fingers is present.

For the application of Architecture 2 trained on the extended training data with represented superscale convection on the laboratory experiment, again, the time step for the concentration field propagation was optimized. In this case, the optimization resulted in $\mathrm{d}\tilde{t} = 0.0336$, which is substantially smaller than the previous value of $\mathrm{d}\tilde{t} = 0.0438$. This is coherent with the presence of the lateral movement of the density fingers. The objective of the time step optimization was to minimize the root mean squared error between the propagated and measured concentration fields. Hence, the lateral movement introduced additional constraints such that not only the finger lengths had to match but also their lateral positions. Accordingly, the compensation of the underestimated absolute flow velocities by the implicit scaling of the estimated flow fields by the time step is expected to be smaller here.

Architecture 2 trained on the extended training data predicted the expected background flow for the laboratory experiment, whereat regions with strong and weak influence were detected. Hence, the obtained normalized concentration errors showed improved qualitative agreement. The lateral positions of the density fingers were predicted reasonably well, while the propagated concentration values were typically underestimated at the finger tips in accordance to the observed tendency of the method to underestimate the absolute flow velocities. Especially in the regions with a strong influence of the background flow, the encountered errors improved in comparison to Architecture 1 trained on the training data without represented superscale convection (TrainNE1). In the regions where the influence of the background flow was small, smaller errors were encountered for Architecture 1 trained on the training data without represented superscale convection, which I attribute to the compensation by the larger time step. With this and since the predictions of Architecture 2 trained on the

extended training data were more consistent with the measurements, I argue that the optimized time step of $\mathrm{d}\tilde{t} = 0.0336$ is the more consistent one. This is also supported by the experimental time step $\mathrm{d}\tilde{t} = 0.034 \pm 0.07$, however, the reliability is limited due to the large uncertainties there.

The main limitation of the method to typically underestimate the absolute flow velocities remains at this point. However, using the presented approach I was able to estimate the otherwise inaccessible flow fields in the laboratory experiment, which were affected by superscale convection. The main advantage is that this was achieved without the explicit knowledge of neither the boundary condition with respect to the concentration, nor the boundary condition with respect to the flow.

7 | Conclusion & Outlook

Improving the representation of hydrological systems is challenging and usually relies on the availability of accurate and dense measurements. Often, researchers in this field encounter the situation, where a good understanding of the physical processes is given, but information about the observed system is incomplete. The presence of nonlinear processes in combination with uncertain boundary conditions typically impedes accurate predictions. Based on the process understanding and increasing computational power, large datasets generated by numerical simulations are available. These simulations offer the advantage of detailed information, also of quantities that are difficult to measure. To close the information gap in real systems, I propose to use recent deep learning methods that are capable of incorporating the process representation contained in synthetic datasets to transfer the information to the measurements. In this work, I demonstrated this approach on the example of flow field estimation for active solute transport observed in a Hele-Shaw cell experiment, where high resolution measurements of the solute concentration distribution and its temporal development are available, while the direct inference of the corresponding flow fields is impossible.

Using numerical simulation of the physical processes, I generated synthetic datasets over a wide range of Rayleigh numbers. I used a CNN that I adapted from work on optical flow estimation [*Ilg et al.*, 2017] to estimate the flow fields, after training it on the synthetic data.

The application of the trained CNN to a synthetic test case showed that the method was able to learn the structurally correct representation of the physical processes, hence, the flow fields were predicted reliably in their flow direction. Additionally, the method was able to generalize toward spatial and temporal variations in the upper concentration boundary condition. However, limitations were found in the estimation of the absolute flow velocities, for which the CNN tends to underestimate the true values.

To explore the capabilities on real data, I applied the flow field estimation with the CNN to a Hele-Shaw cell experiment. The estimated flow fields showed the same structural quality as for the synthetic test case. Temporal propagation of the concentration fields using the estimated flow fields was able to explain the measurements reasonably well in the center of the cell, while the underestimation of the absolute flow velocities also remained in this case. In the outer regions, the observed agreement between the propagated concentration fields and the measurements was poor and strong influences of background flow in the laboratory experiment were identified as being unrepresented in the synthetic data. By extending the training data to also incorporate superscale convection and using an extended version of the CNN, I resolved this issue. The method was able to reliably predict the direction of the background flow and hence the influence of superscale convection without explicit knowledge of the

boundary conditions, which is remarkable. This capability to estimate missing system quantities without explicit knowledge of boundary conditions is relevant to solve inverse problems in hydrology and is a major advantage compared to other methods used to address these problems. Alternatives, like inverse modeling, are typically very sensitive to boundary conditions, which are challenging to infer correctly.

To resolve the main limitation of the method to correctly estimate the absolute flow velocities, which I mainly attribute to ambiguous detection of the solute movement due to the encountered smooth concentration gradients in the input data, a better understanding of the input-output relation of the CNNs is needed. Approaches to gain insight into the network internal signal propagation for the problem of image recognition [e.g., *Zeiler and Fergus*, 2014; *Olah et al.*, 2018; *Carter et al.*, 2019] need to be adapted, utilized, and interpreted for physical problems and could help in this regard. Also, I expect that a more explicit incorporation of time information by employing principles of long short-term memory [*Hochreiter and Schmidhuber*, 1997] to enable inference of the flow fields from the entire temporal development of the experiments is advantageous over the input of only two subsequent concentration fields. Corresponding technical improvement of the CNNs possibly leads to a more robust relation between the movement observed in the input and the corresponding flow velocities.

The presented approach is not limited to flow field estimation of active solute transport, which I chose for the demonstration. The conceptual idea can be transferred to other inverse problems, such as estimation of soil material properties based on observations of passive solute transport. Other inverse problems will give rise to new challenges, which require further development of the utilized deep learning methods. First steps are already done, e.g. with Bayesian inference of the internal model parameters of CNNs [*Zhu and Zabaras*, 2018] to quantify the uncertainty and the reliability of the network estimations. In conjunction with the availability of suited training data, I expect a wide applicability of the presented approach.

A | Appendix

A.1 Additional Data: Numerical Experiments

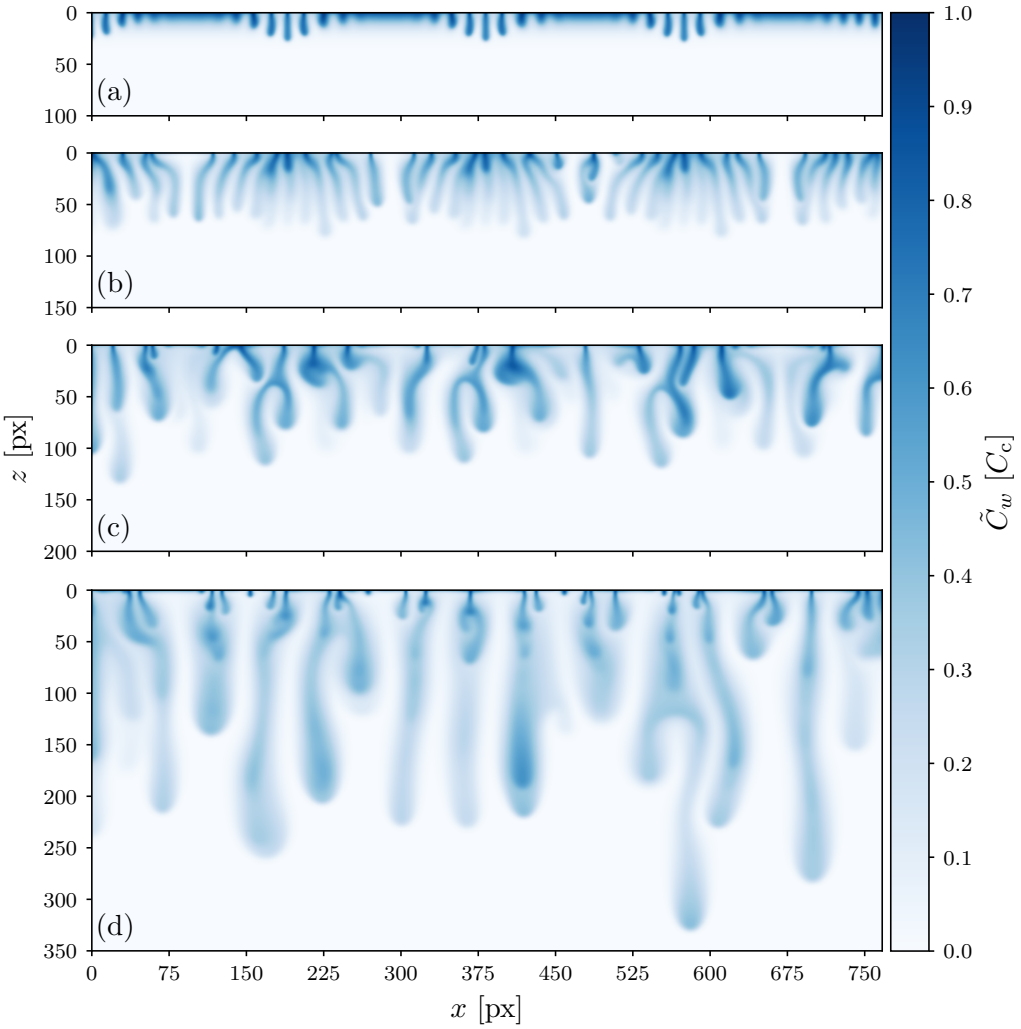

Figure A.1: Concentration fields of NE1 for $\mathrm{Ra}_{\mathrm{sim}} = 12{,}000$ at $\tilde{t} = 1.4$ (a), $\tilde{t} = 2.0$ (b), $\tilde{t} = 2.6$ (c), and $\tilde{t} = 3.9$ (d). The complete width of the domain is presented, while the shown portion of the depth is adapted to the density fingers.

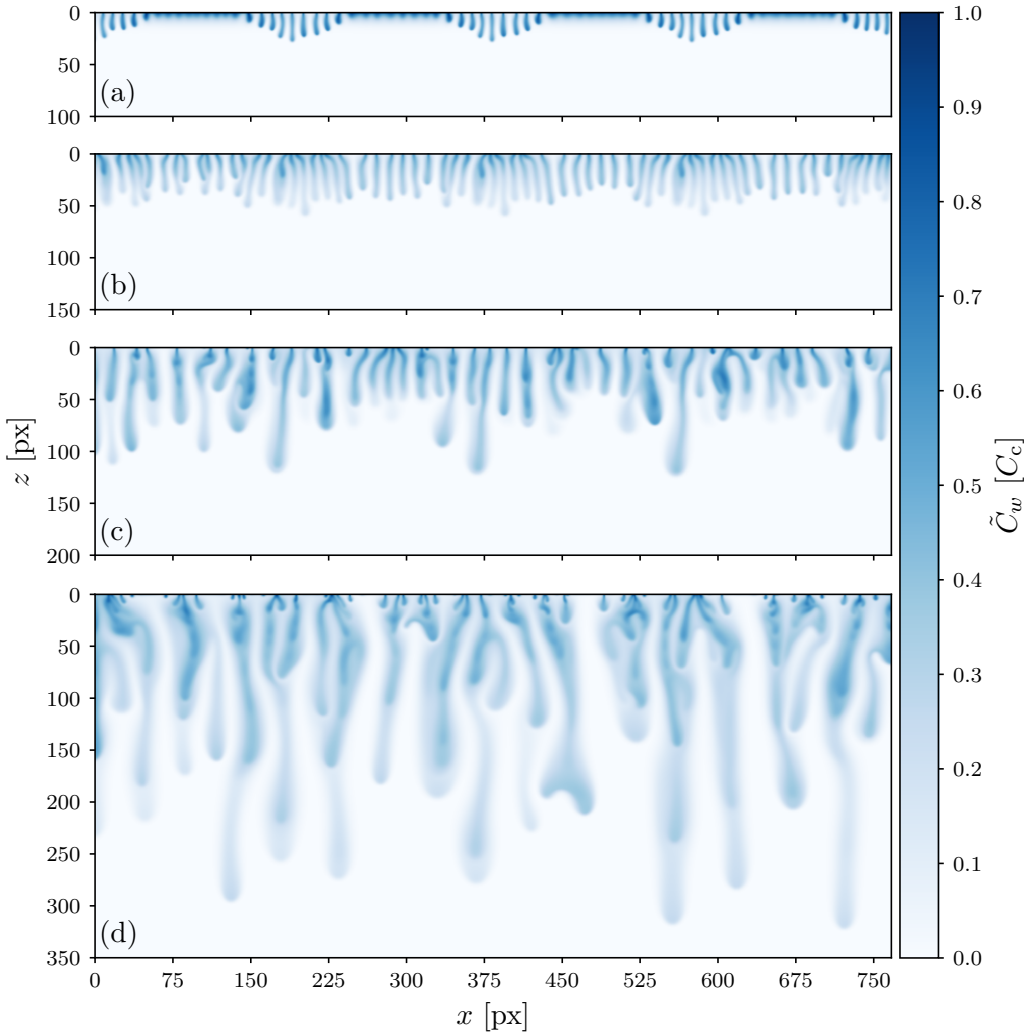

Figure A.2: Concentration fields of NE1 for $\mathrm{Ra}_{\mathrm{sim}} = 26{,}000$ at $\tilde{t} = 1.0$ (a), $\tilde{t} = 1.3$ (b), $\tilde{t} = 2.0$ (c), and $\tilde{t} = 3.6$ (d). Same representation as in Fig. A.1.

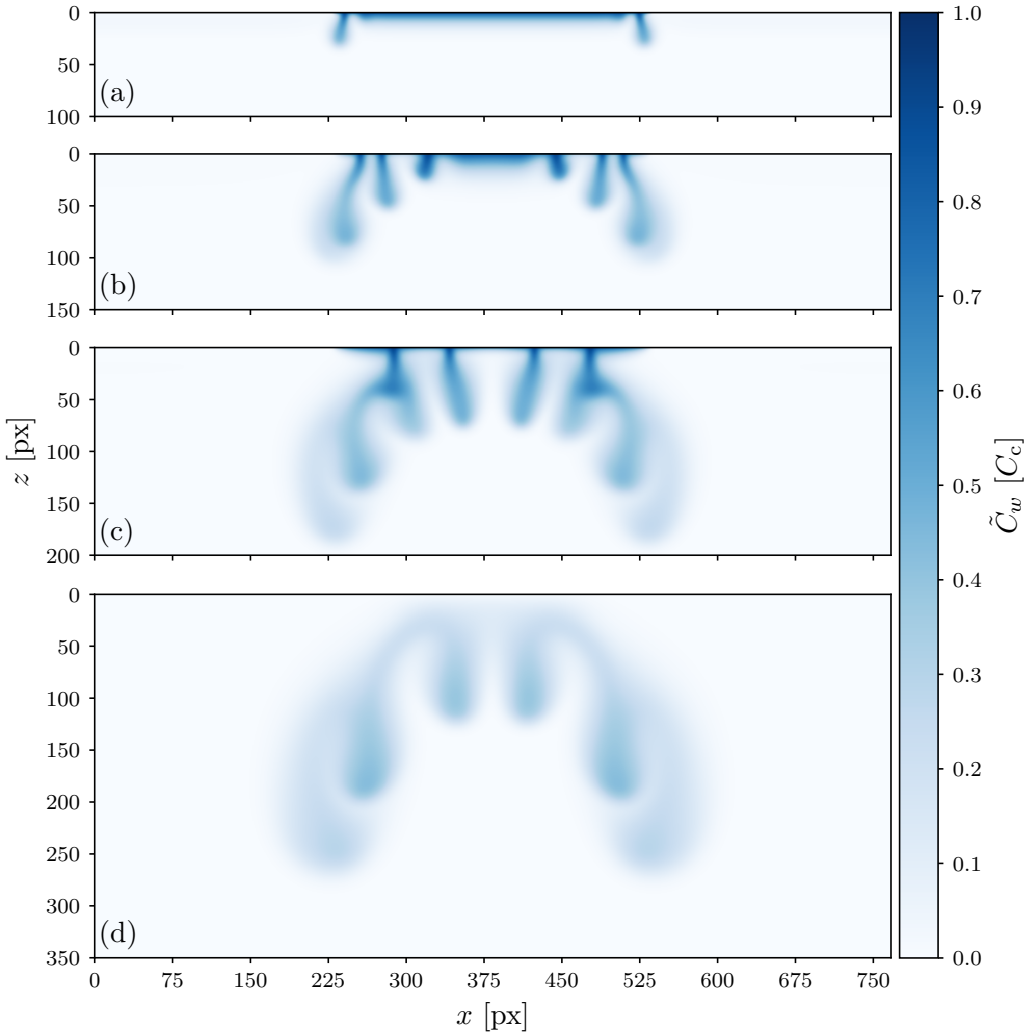

Figure A.3: Concentration fields of NE2 for $\mathrm{Ra}_{\mathrm{sim}} = 3{,}750$ at $\tilde{t} = 0.3$ (a), $\tilde{t} = 1.0$ (b), $\tilde{t} = 1.8$ (c), and $\tilde{t} = 2.8$ (d). Same representation as in Fig. A.1.

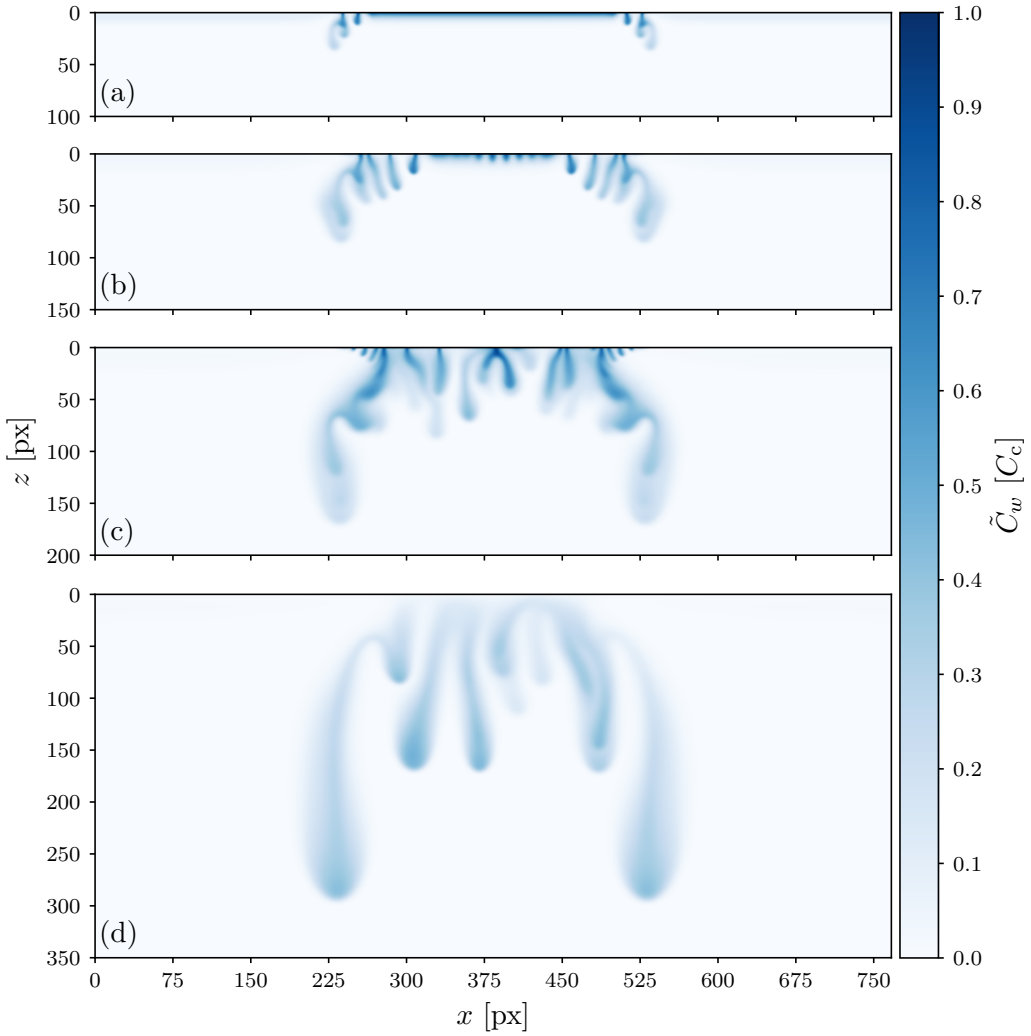

Figure A.4: Concentration fields of NE2 for $\mathrm{Ra}_{\mathrm{sim}} = 13{,}750$ at $\tilde{t} = 0.3$ (a), $\tilde{t} = 0.8$ (b), $\tilde{t} = 1.6$ (c), and $\tilde{t} = 2.7$ (d). Same representation as in Fig. A.1.

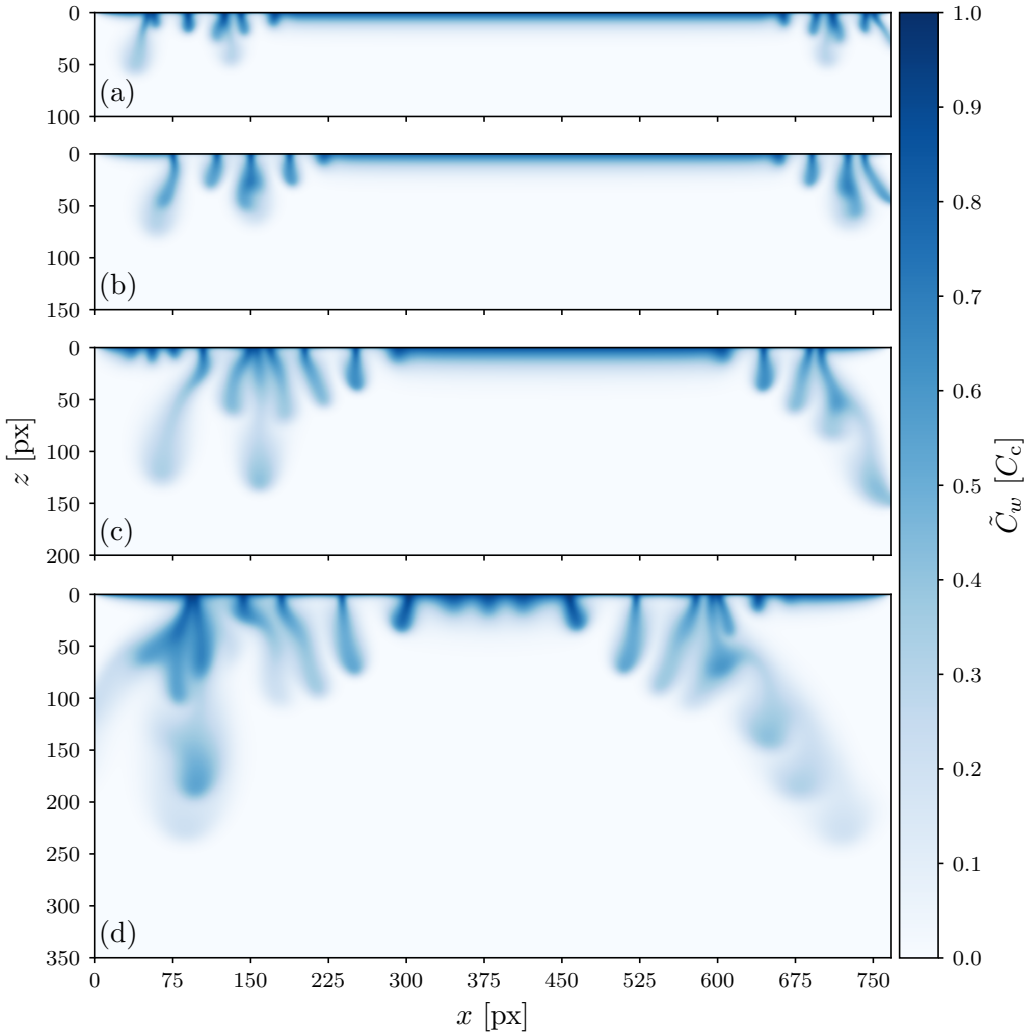

Figure A.5: Concentration fields of NE3 for $\mathrm{Ra}_{\mathrm{sim}} = 4{,}500$ at $\tilde{t} = 0.5$ (a), $\tilde{t} = 0.7$ (b), $\tilde{t} = 1.2$ (c), and $\tilde{t} = 2.0$ (d). Same representation as in Fig. A.1.

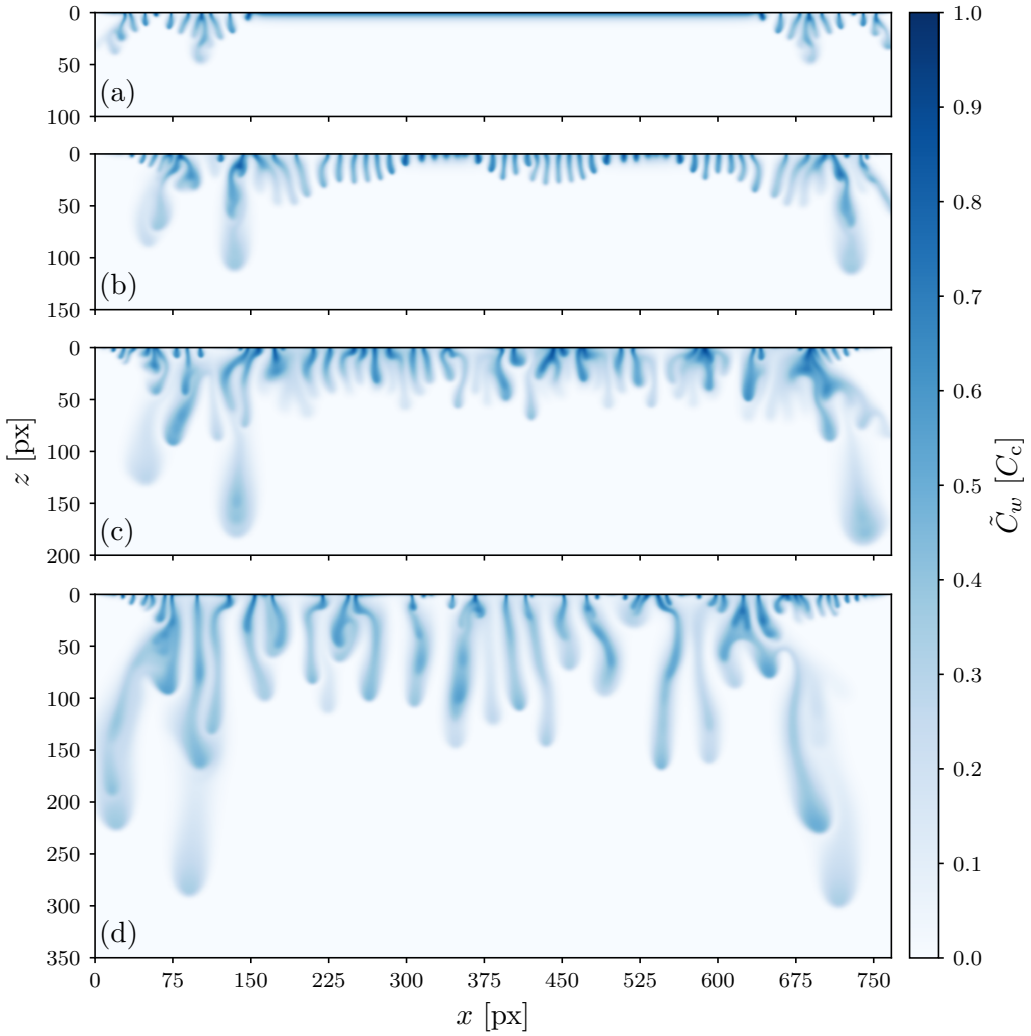

Figure A.6: Concentration fields of NE3 for $\mathrm{Ra}_{\mathrm{sim}} = 16{,}000$ at $\tilde{t} = 0.4$ (a), $\tilde{t} = 0.9$ (b), $\tilde{t} = 1.4$ (c), and $\tilde{t} = 2.1$ (d). Same representation as in Fig. A.1.

A.2 Network Architecture Details

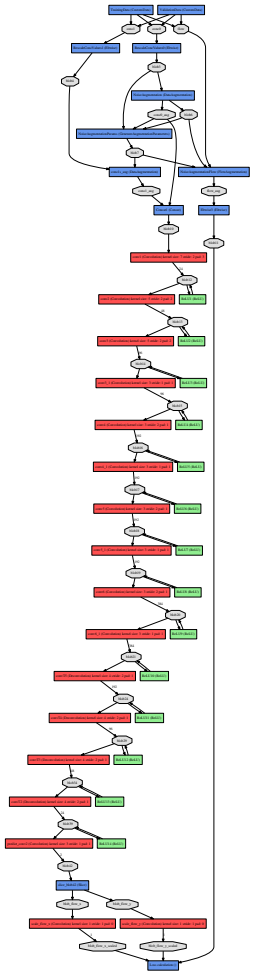

Figure A.7: Details of Architecture 1 (not optimized for print): The feature maps (gray) are connected with convolution and transposed convolution layers (red) that are followed by leaky ReLU activation functions (green). Utility layers like data augmentation, slicing and concatenation, and the calculation of the loss (cf. Fig. A.11) are depicted in blue.

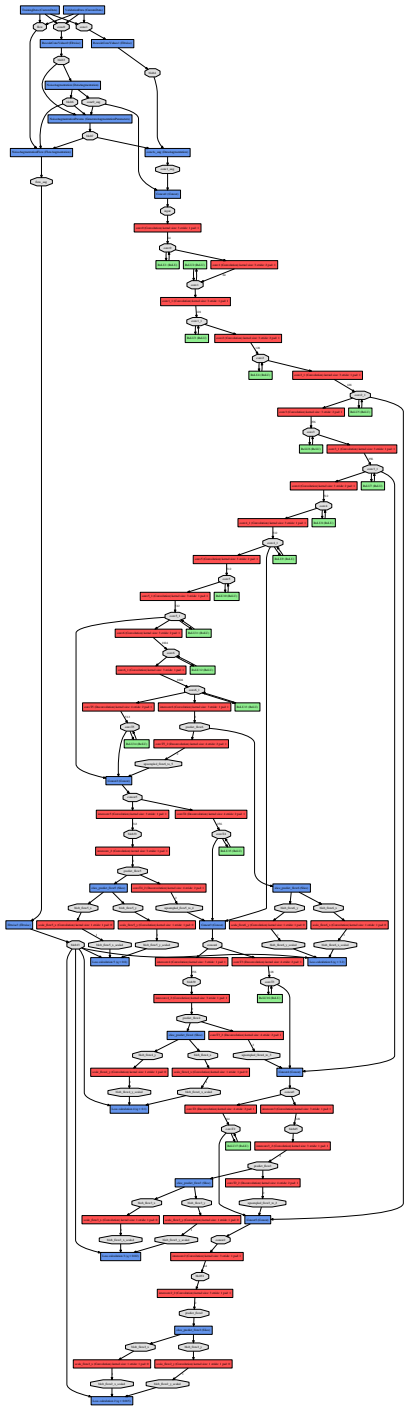

Figure A.8: Details of Architecture 2 (not optimized for print): The loss contributions at the coarse resolutions are calculated as depicted in Fig. A.11, which are combined by a weighted sum according to the weights q. Same representation as Fig. A.7.

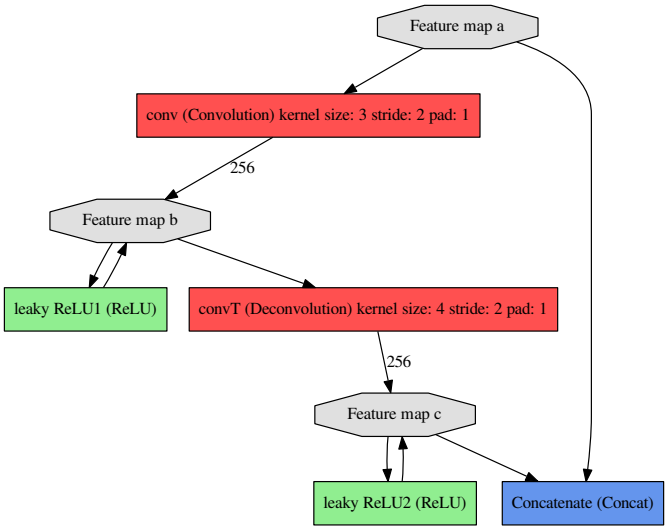

Figure A.9: Detail of a skip connection: convolution and transposed convolution layers are bypassed by concatenating a shallower feature map a with a deeper feature map c.

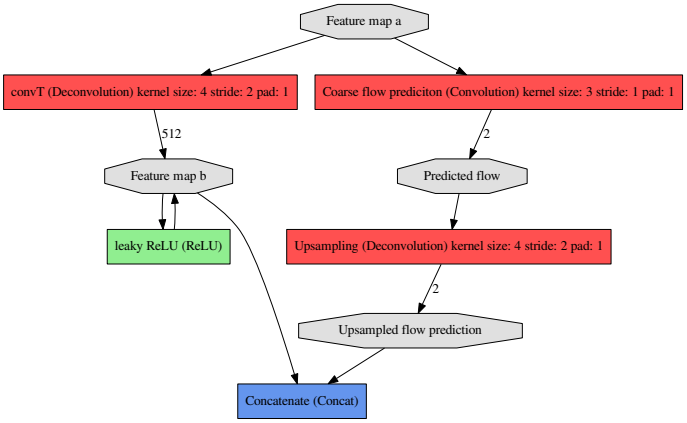

Figure A.10: Detail of an upsampling connection: similar to the skip connection (cf. Fig. A.9), the transposed convolution layers is bypassed by flow prediciton using a convolution layer and subsequent upsamling using a transposed convolution layer (both without activation functions).

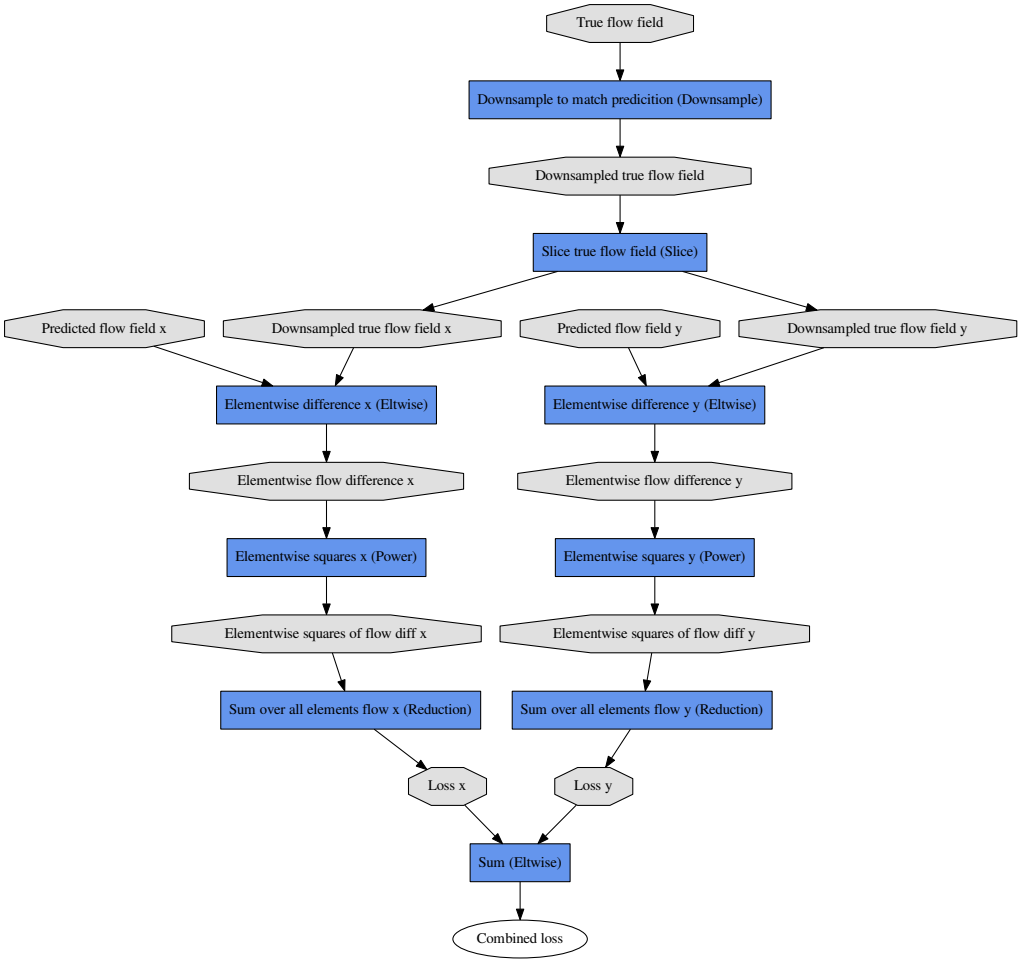

Figure A.11: Illustration of the loss calculation as defined in Eq. (5.39): several utility layers (blue) are used to calculate the elementwise differences and squares of the flow components (gray), before the sum over all elements is calculates as the loss.

A.3 Additional Data: Application

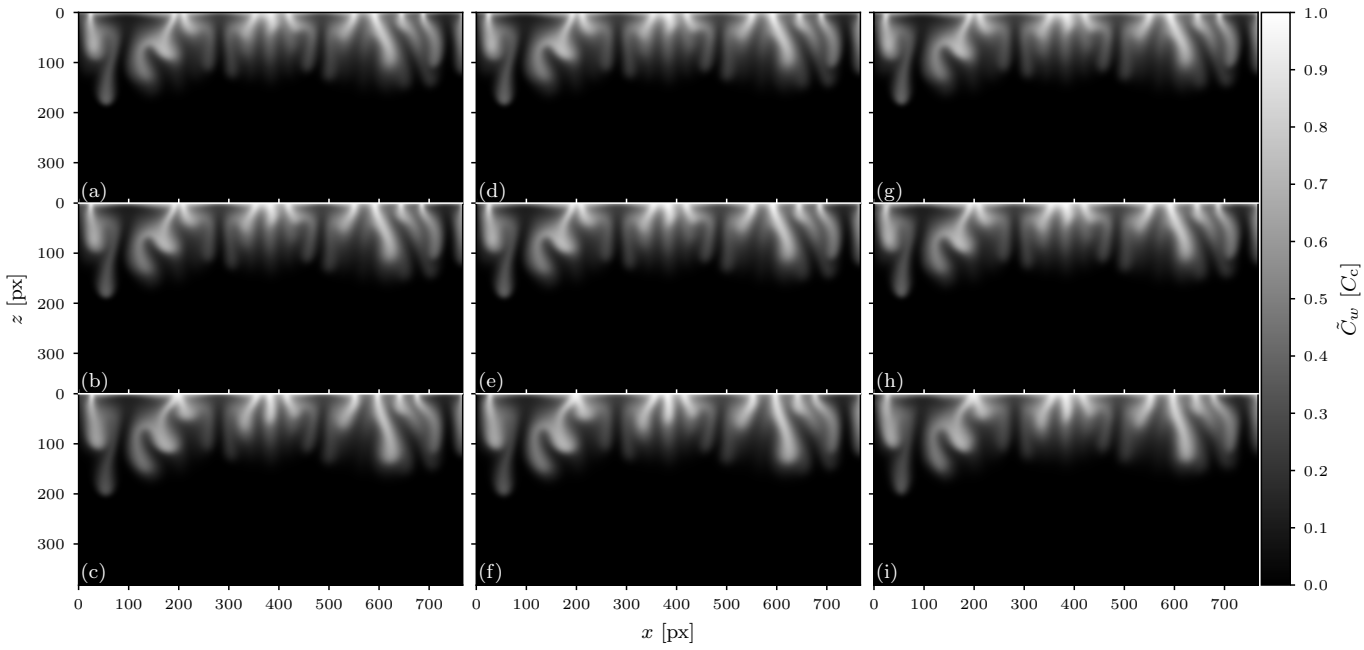

Figure A.12: Concentration field propagation by $\tilde{t}_\mathrm{p} = 0.04$ (top row), 0.08 (center row), and 0.22 (bottom row) for TrainNE1: true concentration fields (a)-(c), concentration fields propagated with true flow fields (d)-(f), and concentration fields propagated with estimated flow fields (g)-(i).

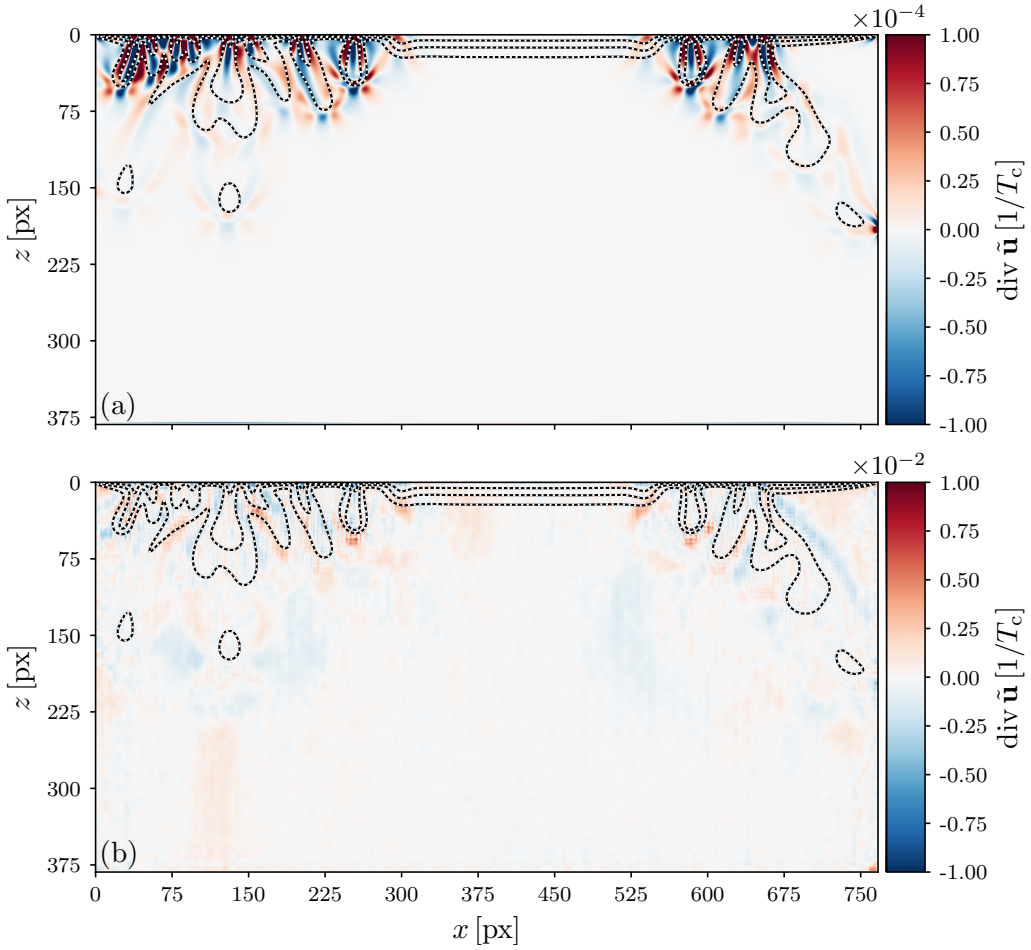

Figure A.13: Flow field divergence on the ValidateNE1+3 example with represented superscale convection for the true (a) and the estimated (b) flow fields. Concentration isolines are given at levels $\tilde{C} = (0.25, 0.5, 0.75)$.

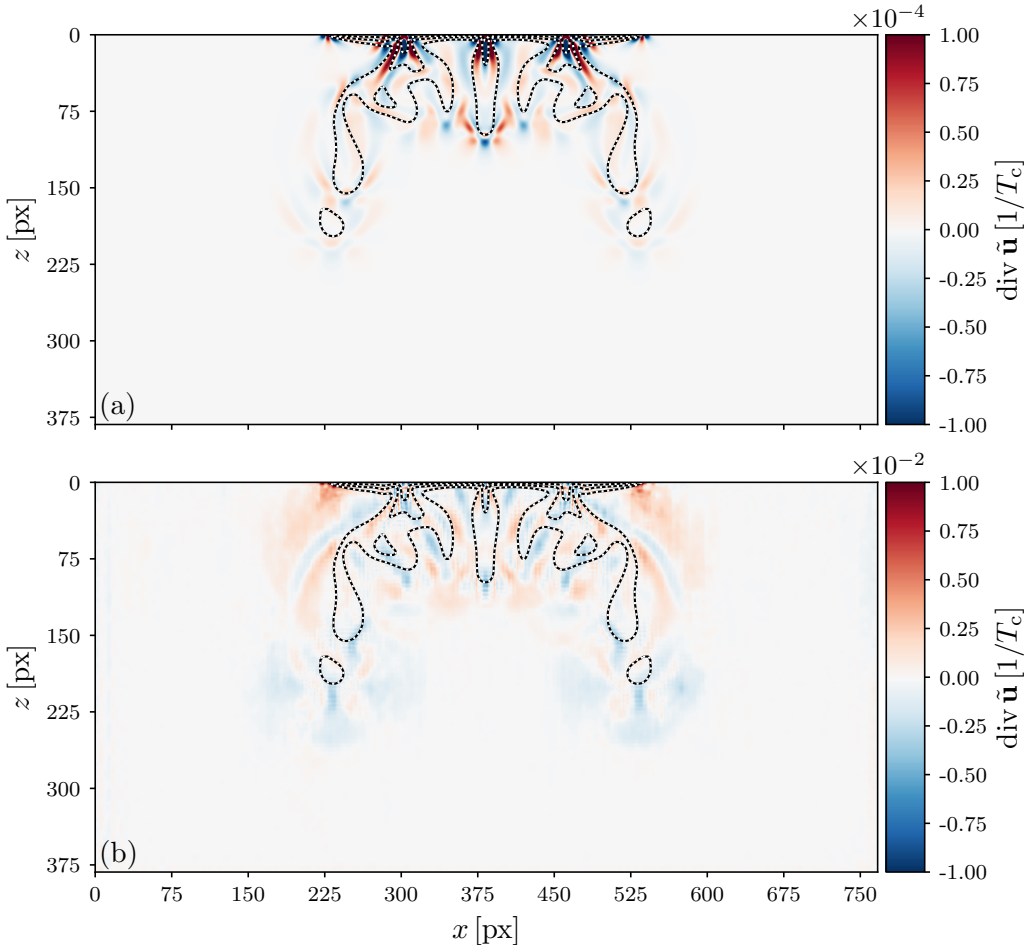

Figure A.14: Flow field divergence on the TestNE2 example with represented superscale convection for the true (a) and the estimated (b) flow fields. Same representation as in Fig. A.13.

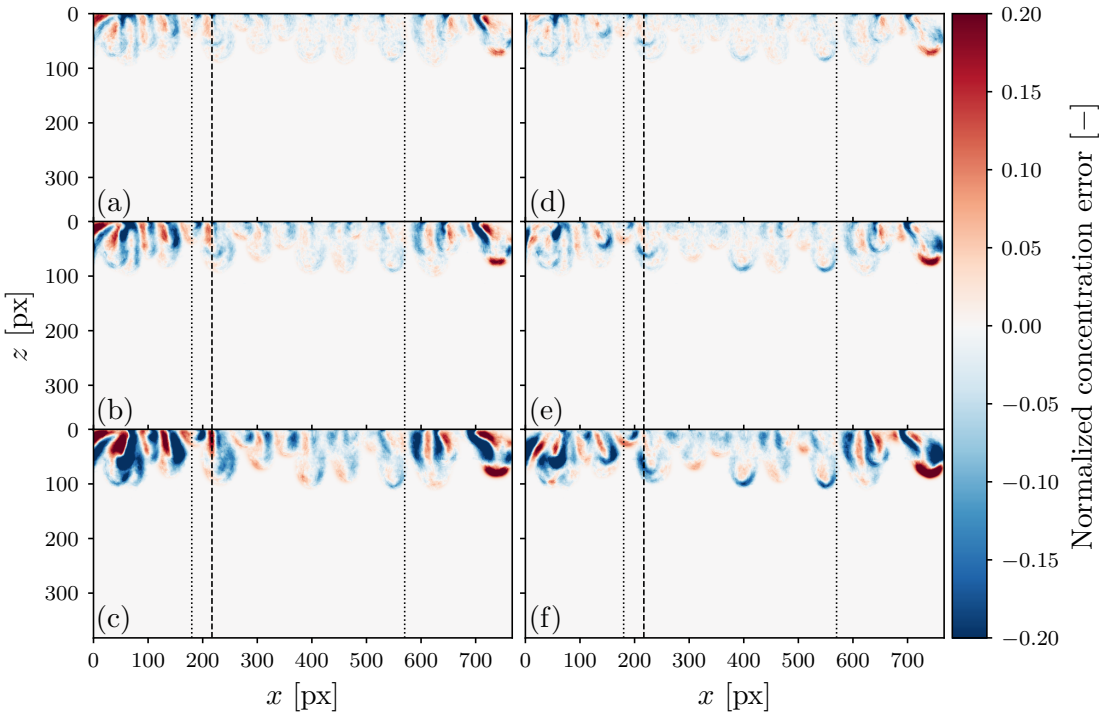

Figure A.15: Comparison of the normalized concentration errors for the cases with and without represented superscale convection using only Architecture 1: Results for training on TrainNE1 ((a) to (c)) and training on TrainNE1+3 ((d) to (f)), both after one (top row), two (middle row), and five (bottom row) propagation steps using their respective optimized time steps $\mathrm{d}\tilde{t}$. Normalized concentration errors are chosen to be negative for $\tilde{C}_w^{\mathrm{est}} < \tilde{C}_w^{\mathrm{true}}$. The black dotted lines indicate regions with strong and weak influence of the background flow. The black dashed line indicates the initial position of the respective density finger seeding point.

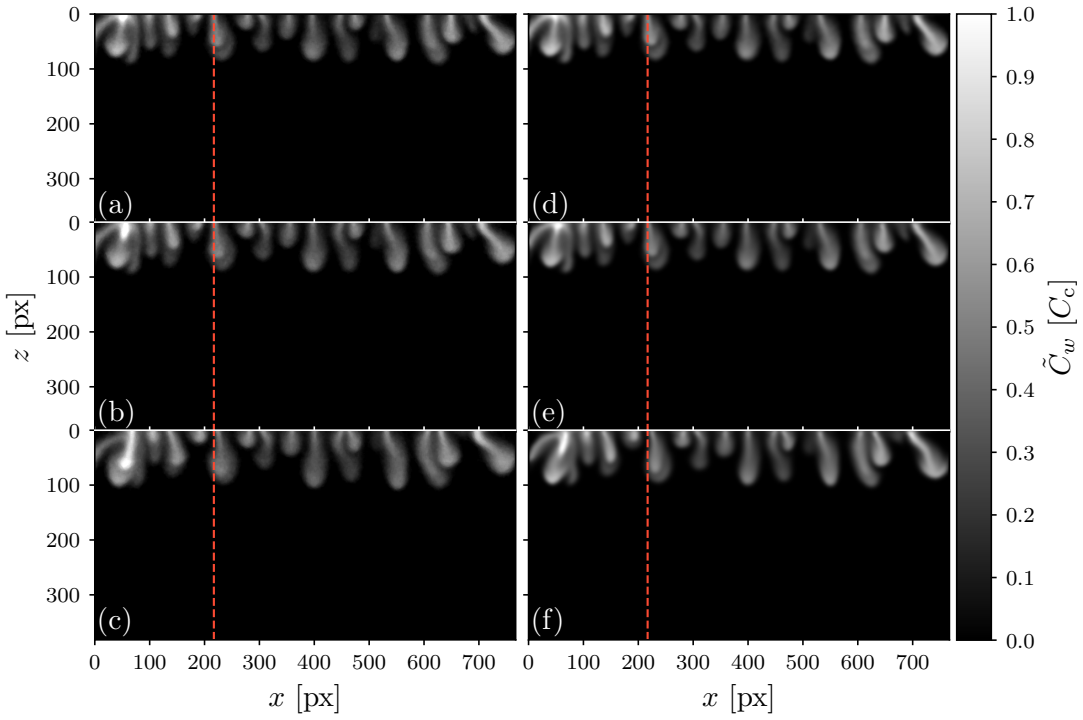

Figure A.16: Concentration field propagation for the laboratory experiment with represented superscale convection using Architecture 2: True concentration fields (a)-(c) and concentration fields propagated with estimated flow fields (d)-(f) at propagation times $\tilde{t}_p = 0.0336$ (top row), $\tilde{t}_p = 0.0672$ (center row), and $\tilde{t}_p = 0.168$ (bottom row). The red dashed line marks the initial position of the respective density finger seeding point.

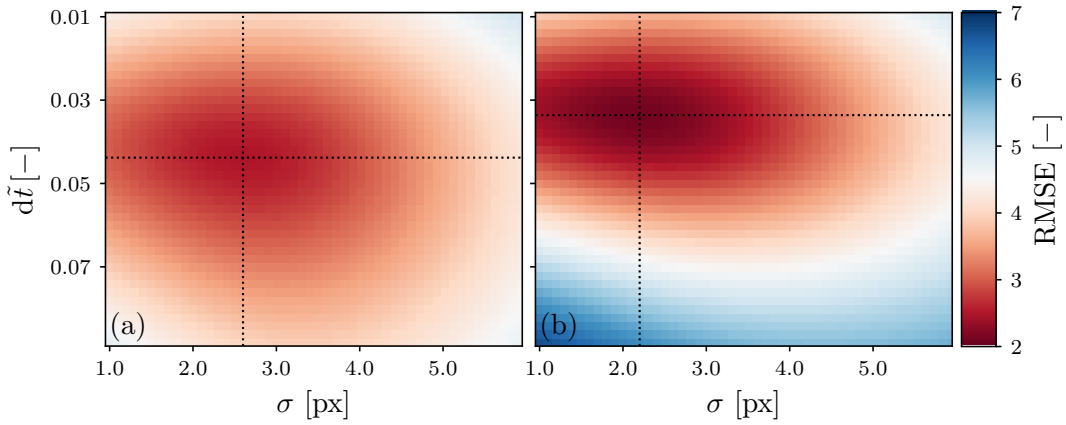

Figure A.17: Results of the parameter scans for the optimized dimensionless time step and standard deviation of the Gaussian image filter: The root mean squared error (RMSE) of the propagated and measured concentration fields, $\sqrt{\left[\sum_{j=1}^{N}\left[\tilde{C}_j^{\mathrm{est}}-\tilde{C}_j^{\mathrm{true}}\right]^2\right]/N}$ with N being the number of pixels, for Architecture 1 trained on TrainNE1 (a) and Architecture 2 trained on TrainNE1+3 (b). The black dotted lines indicate the resulting values $\mathrm{d}\tilde{t} = 0.0438$, $\sigma = 2.6\,\mathrm{px}$ (a) and $\mathrm{d}\tilde{t} = 0.0336$, $\sigma = 2.2\,\mathrm{px}$ (b). Note that the minimum for Architecture 2 trained on TrainNE1+3 is lower showing a overall better agreement with the measurements.

B | List of Figures

C | List of Tables

Acknowledgments

I would like to thank all the people, without whom the work on the presented project would not have been possible in the same way.

For giving me the opportunity to work on the project I would like to thank my advisor Prof. Dr. Kurt Roth. Thank you for your inspiration and always being present when discussion about the arising challenges was needed. Also I would like to thank Prof. Dr. Werner Aeschbach for immediately agreeing to the role as referee of this thesis. For providing the implementation of the numerical solver, I would like to express my honest appreciation to Prof. Dr. Peter Bastian.

Although I didn't have official mentors, there are people that I have definitely perceived as such. Especially, I would like to thank Dr. Hannes Bauser, who was always there to discuss the concepts and often provided smart ideas on how to approach the challenges. Also I would like to thank Dr. Daniel Berg, who helped a lot to resolve any computer related issues, and Dr. Lisa Hantschel for the discussions of preliminary results at the different phases of the work, which helped to generate new ideas.

Furthermore, I would like to thank Angelika Gassama for all the help to make the laboratory experiments possible and the creativity, when issues needed to be fixed quickly. Also I want to thank Jule Thome, who was always there for discussions, when I was not entirely sure on how to solve especially technical but also other general issues.

My thanks also go to all the members of the TS-CCEES group. It was a very nice time with you and I will certainly keep the intense, but also fun retreats in memory. Besides the work group there were also many other nice people around the institute. Thank you all, it was very nice that you always were in for a chat.

For the discussions and your comments, when finishing this thesis, special thanks go to Jule Thome, Dr. Lisa Hantschel, Dr. Hannes Bauser, Dr. Johannes Windschuh, and Sabrina Ebenhoch.

I want to thank all of my friends and the people who were around in the last four years. Thank you Vera, Silvan, Julian, Sarah, Lena, Robert, Jil, Dominik, Philipp, Johnny, Therese, Jan, Marian, Christopher, and Maria for all the time we spend together and the laughs we shared. Without you, this phase of my life would have been a lot less enjoyable.

And then, I want to thank my mother and my father, who seem to have done a very decent job in promoting my curiosity while they raised me. Thank you for that and for just being there.

Bibliography of Own Publications

In parts, the results presented in this dissertation have already been published in a peer reviewed journal. The publication, which I wrote as first author, is listed below. Identical figures, tables, and wording occur in this work.

Kreyenberg, P. J., H. H. Bauser, and K. Roth, Velocity field estimation on density-driven solute transport with a convolutional neural network, *Water Resources Research*, doi:10.1029/2019WR024833, 2019.

Parts of the datasets supporting this work have been made publicly available.

Kreyenberg, P. J., H. H. Bauser, and K. Roth, Velocity field estimation on density-driven solute transport with a convolutional neural network [Dataset], *heiDATA*, doi:10.11588/data/7NEEKF, 2019.

References

Abadi, M., et al., TensorFlow: Large-scale machine learning on heterogeneous distributed systems, *arXiv preprint arXiv:1603.04467*, 2016.

Alexander, R., Diagonally implicit Runge–Kutta methods for stiff ODE's, *SIAM Journal on Numerical Analysis*, *14*(6), 1006–1021, doi:10.1137/0714068, 1977.

Aris, R., On the dispersion of a solute in a fluid flowing through a tube, *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, *235*(1200), 67–77, doi:10.1098/rspa.1956.0065, 1956.

Backhaus, S., K. Turitsyn, and R. E. Ecke, Convective instability and mass transport of diffusion layers in a Hele-Shaw geometry, *Physical Review Letters*, *106*(10), 104,501, doi:10.1103/PhysRevLett.106.104501, 2011.

Bastian, P., M. Blatt, A. Dedner, C. Engwer, R. Klöfkorn, M. Ohlberger, and O. Sander, A generic grid interface for parallel and adaptive scientific computing. Part I: Abstract framework, *Computing*, *82*(2), 103–119, doi:10.1007/s00607-008-0003-x, 2008a.

Bastian, P., M. Blatt, A. Dedner, C. Engwer, R. Klöfkorn, R. Kornhuber, M. Ohlberger, and O. Sander, A generic grid interface for parallel and adaptive scientific computing. Part II: Implementation and tests in Dune, *Computing*, *82*(2), 121–138, doi:10.1007/s00607-008-0004-9, 2008b.

Bear, J., On the tensor form of dispersion in porous media, *Journal of Geophysical Research*, *66*(4), 1185–1197, doi:10.1029/JZ066i004p01185, 1961.

Bénard, H., Les tourbillons cellulaire dans nappe liquide transportant de la chaleur purconvections en regime permanent, *Rev. Gen. Sci. Pures Appl. Bull. Assoc.*, *11*, 1309–1328, 1900.

Bengio, Y., Learning deep architectures for AI, *Foundations and trends® in Machine Learning*, *2*(1), 1–127, doi:10.1561/2200000006, 2009.

Bengio, Y., Practical recommendations for gradient-based training of deep architectures, in *Neural networks: Tricks of the trade*, pp. 437–478, Springer, doi:10.1007/978-3-642-35289-8_26, 2012.

Bengio, Y., P. Lamblin, D. Popovici, and H. Larochelle, Greedy layer-wise training of deep networks, in *Advances in neural information processing systems*, pp. 153–160, 2007.

Blatt, M., and P. Bastian, The iterative solver template library, in *Applied parallel computing. State of the art in scientific computing*, vol. 4699, edited by B. Kågström, E. Elmroth, J. Dongarra, and J. Waśniewski, pp. 666–675, Springer Berlin Heidelberg, Berlin, Heidelberg, doi:10.1007/978-3-540-75755-9_82, 2007.

Blatt, M., and P. Bastian, On the generic parallelisation of iterative solvers for the finite element method, *Int. J. Comput. Sci. Eng.*, *4*(1), 56–69, doi:10.1504/IJCSE.2008.021112, 2008.

Blatt, M., et al., The distributed and unified numerics environment, version 2.4, *Archive of Numerical Software*, *4*(100), 13–29, doi:10.11588/ans.2016.100.26526, 2016.

Boussinesq, J., Théorie analytique de la chaleur (analytic theory of heat). Tome, Paris, Gauthier-Villars, 1903.

Carter, S., Z. Armstrong, L. Schubert, I. Johnson, and C. Olah, Exploring neural networks with activation atlases, *Distill*, *4*(3), e15, doi:10.23915/distill.00015, 2019.

Cauchy, A., Méthode générale pour la résolution des systemes d'équations simultanées, *Comp. Rend. Sci. Paris*, *25*(1847), 536–538, 1847.

Cireşan, D., U. Meier, J. Masci, and J. Schmidhuber, Multi-column deep neural network for traffic sign classification, *Neural networks*, *32*, 333–338, doi:10.1016/j.neunet.2012.02.023, 2012.

de Bezenac, E., A. Pajot, and P. Gallinari, Deep learning for physical processes: Incorporating prior scientific knowledge, *arXiv:1711.07970 [cs, stat]*, 2017.

Diersch, H.-J., and O. Kolditz, Variable-density flow and transport in porous media: Approaches and challenges, *Advances in Water Resources*, *25*(8-12), 899–944, doi:10.1016/S0309-1708(02)00063-5, 2002.

Dosovitskiy, A., P. Fischer, E. Ilg, P. Häusser, C. Hazirbaş, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, FlowNet: Learning optical flow with convolutional networks, in *2015 IEEE international conference on computer vision (ICCV)*, pp. 2758–2766, IEEE, Santiago, doi:10.1109/ICCV.2015.316, 2015.

Ecke, R. E., and S. Backhaus, Plume dynamics in Hele-Shaw porous media convection, *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *374*(2078), 20150,420, doi:10.1098/rsta.2015.0420, 2016.

Ennis-King, J., and L. Paterson, Rate of dissolution due to convective mixing in the underground storage of carbon dioxide, in *Greenhouse gas control technologies - 6th international conference*, edited by J. Gale and Y. Kaya, pp. 1653–1656, Pergamon, Oxford, doi:10.1016/B978-008044276-1/50268-3, 2003.

Ennis-King, J. P., and L. Paterson, Role of convective mixing in the long-term storage of carbon dioxide in deep saline formations, *SPE Journal*, *10*(03), 349–356, doi:10.2118/84344-PA, 2005.

Faisal, T. F., S. Chevalier, and M. Sassi, Experimental and numerical studies of density driven natural convection in saturated porous media with application to CO_2 geological storage, *Energy Procedia*, *37*, 5323–5330, doi:10.1016/j.egypro.2013.06.450, 2013.

Faisal, T. F., S. Chevalier, Y. Bernabe, R. Juanes, and M. Sassi, Quantitative and qualitative study of density driven CO_2 mass transfer in a vertical Hele-Shaw cell, *International Journal of Heat and Mass Transfer*, *81*, 901–914, doi:10.1016/j.ijheatmasstransfer.2014.11.017, 2015.

Farabet, C., C. Couprie, L. Najman, and Y. LeCun, Learning hierarchical features for scene labeling, *IEEE transactions on pattern analysis and machine intelligence*, *35*(8), 1915–1929, doi:10.1109/TPAMI.2012.231, 2012.

Farajzadeh, R., H. Salimi, P. L. Zitha, and H. Bruining, Numerical simulation of density-driven natural convection in porous media with application for CO_2 injection projects, *International Journal of Heat and Mass Transfer*, *50*(25-26), 5054–5064, doi:10.1016/j.ijheatmasstransfer.2007.08.019, 2007.

Fernandez, J., P. Kurowski, P. Petitjeans, and E. Meiburg, Density-driven unstable flows of miscible fluids in a Hele-Shaw cell, *Journal of Fluid Mechanics*, *451*, doi:10.1017/S0022112001006504, 2002.

Fick, A., Über Diffusion, *Annalen der Physik*, *170*(1), 59–86, doi:10.1002/andp.18551700105, 1855.

Glorot, X., and Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

Goodfellow, I., Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, ISBN: 0262035618, `http://www.deeplearningbook.org`, 2016.

Graf, F., E. Meiburg, and C. Haertel, Density-driven instabilities of miscible fluids in a Hele-Shaw cell: linear stability analysis of the three-dimensional Stokes equations, *Journal of Fluid Mechanics*, *451*, 261–282, doi:10.1017/S0022112001006516, 2002.

Guo, Y., Deep learning for visual understanding, Ph.D. thesis, Leiden University, the Netherlands, 2017.

Hassanzadeh, H., M. Pooladi-Darvish, and D. W. Keith, Modelling of convective mixing in CO_2 storage, *Journal of Canadian Petroleum Technology*, *44*(10), doi:10.2118/05-10-04, 2005.

He, K., X. Zhang, S. Ren, and J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

He, K., X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hele-Shaw, H. S., Investigation of the nature of surface resistance of water and of stream-line motion under certain experimental conditions, *Trans. Instn Nav. Archit., Lond.*, *40*, 21, 1898.

Hinton, G. E., N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, *arXiv preprint arXiv:1207.0580*, 2012.

Hochreiter, S., and J. Schmidhuber, Long short-term memory, *Neural computation*, *9*(8), 1735–1780, doi:10.1162/neco.1997.9.8.1735, 1997.

Horton, C., and F. Rogers Jr, Convection currents in a porous medium, *Journal of Applied Physics*, *16*(6), 367–370, doi:10.1063/1.1707601, 1945.

Ilg, E., N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, FlowNet 2.0: Evolution of optical flow estimation with deep networks, in *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 1647–1655, IEEE, Honolulu, HI, doi:10.1109/CVPR.2017.179, 2017.

IPCC, IPCC special report on carbon dioxide capture and storage, *UK: Cambridge University Press*, 2005.

Janai, J., F. Güney, A. Behl, and A. Geiger, Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art, *arXiv:1704.05519 [cs]*, 2017.

Jia, Y., E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, Caffe: Convolutional architecture for fast feature embedding, *arXiv:1408.5093 [cs]*, 2014.

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

Kneafsey, T. J., and K. Pruess, Laboratory flow experiments for visualizing carbon dioxide-induced, density-driven brine convection, *Transport in Porous Media*, *82*(1), 123–139, doi:10.1007/s11242-009-9482-2, 2010.

Kneafsey, T. J., and K. Pruess, Laboratory experiments and numerical simulation studies of convectively enhanced carbon dioxide dissolution, *Energy Procedia*, *4*, 5114–5121, doi:10.1016/j.egypro.2011.02.487, 2011.

Kolditz, O., R. Ratke, H.-J. G. Diersch, and W. Zielke, Coupled groundwater flow and transport: 1. Verification of variable density flow and transport models, *Advances in Water Resources*, *21*(1), 27–46, doi:10.1016/S0309-1708(96)00034-6, 1998.

Kreyenberg, P. J., Experimental studies on evapo-induced density-driven flow in Hele-Shaw cells, Master's thesis, Heidelberg University, Germany, 2015.

Kreyenberg, P. J., H. H. Bauser, and K. Roth, Velocity field estimation on density-driven solute transport with a convolutional neural network, *Water Resources Research*, doi:10.1029/2019WR024833, 2019.

Krizhevsky, A., I. Sutskever, and G. E. Hinton, Imagenet classification with deep convolutional neural networks, in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Lapwood, E., Convection of a fluid in a porous medium, in *Mathematical proceedings of the Cambridge Philosophical Society*, vol. 44, pp. 508–521, Cambridge University Press, doi:10.1017/S030500410002452X, 1948.

Lide, D. R., *CRC handbook of chemistry and physics*, vol. 85, CRC press, 2004.

Lindeberg, E., and D. Wessel-Berg, Vertical convection in an aquifer column under a gas cap of CO_2, *Energy Conversion and Management*, *38*, S229–S234, doi:10.1016/S0196-8904(96)00274-9, 1997.

Liu, Q., Stein variational gradient descent as gradient flow, in *Advances in neural information processing systems*, pp. 3115–3123, 2017.

Liu, Q., and D. Wang, Stein variational gradient descent: A general purpose Bayesian inference algorithm, in *Advances in neural information processing systems*, pp. 2378–2386, 2016.

Maas, A. L., A. Y. Hannun, and A. Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in *Proc. icml*, vol. 30, p. 3, 2013.

Marçais, J., and J.-R. de Dreuzy, Prospective interest of deep learning for hydrological inference, *Groundwater*, *55*(5), 688–692, doi:10.1111/gwat.12557, 2017.

Mo, S., Y. Zhu, N. Zabaras, X. Shi, and J. Wu, Deep convolutional encoder-decoder networks for uncertainty quantification of dynamic multiphase flow in heterogeneous media, *Water Resources Research*, *55*(1), 703–728, doi:10.1029/2018WR023528, 2019.

Nguyen, A., J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski, Plug & play generative networks: Conditional iterative generation of images in latent space, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4467–4477, 2017.

Nield, D. A., and A. Bejan, *Convection in porous media, 3. ed*, Springer, 2006.

Oberbeck, A., Über die Wärmeleitung der Flüssigkeiten bei Berücksichtigung der Strömungen infolge von Temperaturdifferenzen, *Annalen der Physik*, *243*(6), 271–292, doi:10.1002/andp.18792430606, 1879.

Olah, C., A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev, The building blocks of interpretability, *Distill*, *3*(3), e10, doi:10.23915/distill.00010, 2018.

Oltean, C., C. Felder, M. Panfilov, and M. A. Buès, Transport with a very low density contrast in Hele-Shaw cell and porous medium: Evolution of the mixing zone, *Transport in Porous Media*, *55*(3), 339–360, doi:10.1023/B:TIPM.0000013332.08029.af, 2004.

Paszke, A., et al., Automatic differentiation in PyTorch, in *NIPS autodiff workshop*, 2017.

Pau, G. S., J. B. Bell, K. Pruess, A. S. Almgren, M. J. Lijewski, and K. Zhang, High-resolution simulation and characterization of density-driven flow in CO_2 storage in saline aquifers, *Advances in Water Resources*, *33*(4), 443–455, doi:10.1016/j.advwatres.2010.01.009, 2010.

Petersen, P., and F. Voigtlaender, Equivalence of approximation by convolutional neural networks and fully-connected networks, *arXiv preprint arXiv:1809.00973*, 2018.

Pruess, K., and K. Zhang, Numerical modeling studies of the dissolution-diffusion-convection process during CO_2 storage in saline aquifers, (LBNL-1243E, 944124), doi:10.2172/944124, 2008.

Rasmusson, M., F. Fagerlund, K. Rasmusson, Y. Tsang, and A. Niemi, Refractive-light-transmission technique applied to density-driven convective mixing in porous media with implications for geological CO_2 storage, *Water Resources Research*, *53*(11), 8760–8780, doi:10.1002/2017WR020730, 2017.

Rayleigh, L., LIX. On convection currents in a horizontal layer of fluid, when the higher temperature is on the under side, *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *32*(192), 529–546, doi:10.1080/14786441608635602, 1916.

Riaz, A., M. Hesse, H. A. Tchelepi, and F. M. Orr, Onset of convection in a gravitationally unstable diffusive boundary layer in porous media, *Journal of Fluid Mechanics*, *548*, 87–111, doi:10.1017/S0022112005007494, 2006.

Ronneberger, O., P. Fischer, and T. Brox, U-net: Convolutional networks for biomedical image segmentation, in *International conference on medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

Roth, K., Physics of terrestrial systems. Lecture notes, v0.2., *Institure of Environmental Physics, Heidelberg University*, 2017.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams, Learning representations by back-propagating errors, *Nature*, *323*, 533–536, doi:10.1038/323533a0, 1986.

Scheidegger, A. E., General theory of dispersion in porous media, *Journal of Geophysical Research*, *66*(10), 3273–3278, doi:10.1029/JZ066i010p03273, 1961.

Sermanet, P., K. Kavukcuoglu, S. Chintala, and Y. LeCun, Pedestrian detection with unsupervised multi-stage feature learning, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3626–3633, 2013.

Shen, C., A transdisciplinary review of deep learning research and its relevance for water resources scientists, *Water Resources Research*, *54*(11), 8558–8593, doi:10.1029/2018WR022643, 2018.

Shen, C., et al., Hess opinions: Incubating deep-learning-powered hydrologic science advances as a community, *Hydrology and Earth System Sciences*, *22*(11), doi:10.5194/hess-22-5639-2018, 2018.

Simonyan, K., and A. Zisserman, Very deep convolutional networks for large-scale image recognition, *arXiv preprint arXiv:1409.1556 [cs]*, 2014a.

Simonyan, K., and A. Zisserman, Two-stream convolutional networks for action recognition in videos, *arXiv:1406.2199 [cs]*, 2014b.

Simonyan, K., A. Vedaldi, and A. Zisserman, Deep inside convolutional networks: Visualising image classification models and saliency maps, *arXiv preprint arXiv:1312.6034*, 2013.

Slim, A. C., Solutal-convection regimes in a two-dimensional porous medium, *Journal of Fluid Mechanics*, *741*, 461–491, doi:10.1017/jfm.2013.673, 2014.

Slim, A. C., and T. S. Ramakrishnan, Onset and cessation of time-dependent, dissolution-driven convection in porous media, *Physics of fluids*, *22*(12), 124,103, doi:10.1063/1.3528009, 2010.

Slim, A. C., M. M. Bandi, J. C. Miller, and L. Mahadevan, Dissolution-driven convection in a Hele-Shaw cell, *Physics of Fluids*, *25*(2), 024,101, doi:10.1063/1.4790511, 2013.

Springenberg, J. T., A. Dosovitskiy, T. Brox, and M. Riedmiller, Striving for simplicity: The all convolutional net, *arXiv preprint arXiv:1412.6806*, 2014.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, *The Journal of Machine Learning Research*, *15*(1), 1929–1958, 2014.

Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, Going deeper with convolutions, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Tang, Y., and C. Eliasmith, Deep networks for robust visual recognition, in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 1055–1062, 2010.

Taylor, G. I., Dispersion of soluble matter in solvent flowing slowly through a tube, *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, *219*(1137), 186–203, doi:10.1098/rspa.1953.0139, 1953.

Thomas, C., L. Lemaigre, A. Zalts, A. D'Onofrio, and A. De Wit, Experimental study of CO_2 convective dissolution: The effect of color indicators, *International Journal of Greenhouse Gas Control*, *42*, 525–533, doi:10.1016/j.ijggc.2015.09.002, 2015.

Thomas, C., S. Dehaeck, and A. De Wit, Convective dissolution of CO_2 in water and salt solutions, *International Journal of Greenhouse Gas Control*, *72*, 105–116, doi:10.1016/j.ijggc.2018.01.019, 2018.

Weir, G. J., S. P. White, and W. M. Kissling, Reservoir storage and containment of greenhouse gases, *Energy Conversion and Management*, *36*(6), 531–534, doi:10.1016/0196-8904(95)00060-Q, 1995.

Wooding, R. A., S. W. Tyler, and I. White, Convection in groundwater below an evaporating salt lake: 1. Onset of instability, *Water Resources Research*, *33*(6), 1199–1217, doi:10.1029/96WR03533, 1997a.

Wooding, R. A., S. W. Tyler, I. White, and P. A. Anderson, Convection in groundwater below an evaporating salt lake: 2. Evolution of fingers or plumes, *Water Resources Research*, *33*(6), 1219–1228, doi:10.1029/96WR03534, 1997b.

Yang, C., and Y. Gu, Accelerated mass transfer of CO_2 in reservoir brine due to density-driven natural convection at high pressures and elevated temperatures, *Industrial & Engineering Chemistry Research*, *45*(8), 2430–2436, doi:10.1021/ie050497r, 2006.

Yarotsky, D., Universal approximations of invariant maps by neural networks, *arXiv preprint arXiv:1804.10306*, 2018.

Zeiler, M. D., and R. Fergus, Visualizing and understanding convolutional networks, in *European conference on computer vision*, pp. 818–833, Springer, 2014.

Zeiler, M. D., D. Krishnan, G. W. Taylor, and R. Fergus, Deconvolutional networks, in *2010 IEEE Computer Society conference on computer vision and pattern recognition*, pp. 2528–2535, doi:10.1109/CVPR.2010.5539957, 2010.

Zeiler, M. D., G. W. Taylor, R. Fergus, et al., Adaptive deconvolutional networks for mid and high level feature learning., in *ICCV*, vol. 1, p. 6, 2011.

Zhou, D.-X., Universality of deep convolutional neural networks, *arXiv preprint arXiv:1805.10769*, 2018.

Zhou, P., and J. Feng, Understanding generalization and optimization performance of deep CNNs, in *Proceedings of the 35th international conference on machine learning*, *Proceedings of Machine Learning Research*, vol. 80, edited by J. Dy and A. Krause, pp. 5960–5969, PMLR, Stockholmsmässan, Stockholm Sweden, 2018.

Zhou, Y.-T., and R. Chellappa, Computation of optical flow using a neural network, in *IEEE International Conference on Neural Networks*, vol. 1998, pp. 71–78, 1988.

Zhu, Y., and N. Zabaras, Bayesian deep convolutional encoder-decoder networks for surrogate modeling and uncertainty quantification, *Journal of Computational Physics*, *366*, 415–447, doi:10.1016/j.jcp.2018.04.018, 2018.