FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of master's thesis

Title:	Improving deep learning precipitation nowcasting by using prior knowledge
Student:	Bc. Matej Choma
Supervisor:	Mgr. Petr Šimánek
Study program:	Informatics
Branch / specialization:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2022/2023

Instructions

State-of-the-art models for precipitation nowcasting are based on weather radar data and recurrent neural networks with convolutional cells. This thesis aims to improve the performance of deep learning precipitation nowcasting by using more prior knowledge about the atmosphere.
Explore human knowledge about atmosphere and precipitation.
Explore possibilities of enhancing neural networks with prior knowledge of physical processes.
Propose and train a deep learning rainfall nowcasting model.
Evaluate the model, analyze and discuss the results.

Electronically approved by Ing. Karel Klouda, Ph.D. on 12 September 2021 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Improving Deep Learning Precipitation Nowcasting by Using Prior Knowledge

Bc. Matej Choma

Department of Applied Mathematics

Supervisor: Mgr. Petr Šimánek

May 5, 2022

Acknowledgements

My deep thanks to my supervisor, Mgr. Petr Šimánek, for his leadership during the work on this thesis. By showing me the direction but letting me work the way there alone, he truly helped me grow during the last year. I appreciate our discussions about the influence and physical interpretation of various neural network layers. They enabled me to understand deep learning on a new level.

I would like to express my sincere gratitude to the company Meteopress for the resources and support provided during the writing of this thesis. I am grateful for the opportunity to work on a research topic with real-life impact in an environment where every problem is solvable.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on May 5, 2022 . . . . . .

Czech Technical University in Prague

Faculty of Information Technology

Citation of this thesis

Choma, Matej. *Improving Deep Learning Precipitation Nowcasting by Using Prior Knowledge*. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2022.

Abstract

Deep learning methods dominate short-term high-resolution precipitation nowcasting in terms of prediction error. However, their operational usability is limited by difficulties explaining dynamics behind the predictions, which are smoothed out and missing the high-frequency features due to optimizing for mean error loss functions. This thesis summarizes our progress in addressing these issues.

Firstly, we present *Intensity Classification Loss* to improve the prediction of severe rainfall. The model is trained to predict the probability of precipitation with an intensity over 40 dBZ as a secondary output, which is compared to binary ground truth. Experiments have shown that this approach helps predict severe rainfall but does not predict precipitation with higher intensities than the selected threshold.

Secondly, we experiment with hand-engineering of the *advection-diffusion* differential equation into a PhyCell to introduce more accurate physical prior to a PhyDNet model that disentangles physical and residual dynamics. Results indicate that while PhyCell can learn the intended dynamics, training of PhyDNet remains driven by loss optimization, resulting in a model with the same prediction capabilities.

Keywords Precipitation Nowcasting, Physics Prior, Deep Learning, Loss Engineering

Abstrakt

Pri krátkodobých predpovediach zrážok s vysokým rozlíšením z hľadiska chyby predpovede dominujú metódy hlbokého učenia. Avšak, ich operatívne používanie je obmedzené problémami s vysvetliteľnosťou dynamiky za predpoveďami. Tieto sú zároveň vyhladené a chýbajú im vysokofrekvenčné prvky v dôsledku optimalizácie pre stratové funkcie založené na strednej chybe. V tejto práci je zhrnutý náš pokrok pri riešení týchto problémov.

V prvej časti predstavujeme *Intensity Classification Loss* na zlepšenie predpovede silných zrážok. Model je natrénovaný vytvárať sekundárny výstup predpovedajúci pravdepodobnosť zrážok s intenzitou nad 40 dBZ, ktorý sa porovnáva s binárnou skutočnosťou. Experimenty ukázali, že tento prístup pomáha predpovedať silné zrážky, ale nepredpovedá zrážky s vyššou intenzitou, ako je zvolený prah.

V druhej časti experimentujeme s ručným vkladaním diferenciálnej rovnice *advekcie-difúzie* do PhyCell. Cieľom je vniesť lepšiu apriornú znalosť o fyzike do modelu PhyDNet, ktorý oddeľuje fyzikálnu a reziduálnu dynamiku. Výsledky naznačujú, že zatiaľ čo sa PhyCell dokáže naučiť zamýšľanú dynamiku, tréning modelu PhyDNet zostáva riadený optimalizáciou stratovej funkcie. Toto vedie k modelu s nezmenenými predikčnými vlastnosťami.

Kľúčové slová nowcasting zrážok, apriorná znalosť fyziky, hlboké učenie, modelovanie stratových funkcií

Contents

List of Figures

List of Tables

Introduction

"Our vision is to protect people everywhere from the effects of dangerous weather," Meteopress.[1]

Human civilization is intertwined with the weather. It is normal to adapt day-to-day activities with respect to temperature, wind, and precipitation outside. Homes are built as a shelter from the weather, and its effects on food production inspired cultures around the world. Thus, it is beneficial to know in advance what the weather may be like, and adjust according to it to increase comfort, safety, and profit. However, weather may sometimes be severe, changing in tens of minutes and destroying anything standing in its path. The tornado in Moravia, which happened on June 24, 2021, is a tragic example still in the living memory [6]. In these cases, weather prediction becomes a critical tool for protection.

Precipitation is not only dictating clothes, transport, or moisture for crops, but in our latitudes, it accompanies most of the short-term storm-based severe weather as well. Each time a dark cloud forms on the horizon, a question regarding its future development and severity arises. Luckily, precipitation may be monitored in real-time and in high resolution with weather radars. It may be argued that the observations are sufficient for taking individual protective measures. Nevertheless, humans have many activities when it is impossible to monitor their surroundings actively, and a localized short-term prediction may be game-changing.

For the past three years, we have been exploring the use of deep learning (DL) techniques for short-time high-resolution rainfall prediction in cooperation with the company Meteopress [7]. Building on the PhyDNet architecture

[1] https://www.meteopress.com

disentangling physical from unknown dynamics [4], we have achieved unparalleled quantitative performance of an operational precipitation nowcasting system [8].

However, this is not a competition, and there is still room for improvement. The difficulty of explaining dynamics learned by a DL model lowers the trustworthiness of the predictions in the eyes of meteorologists. The regression formulation of the learning problem, guided by mean error loss functions, results in the ignorance of hardly predictable high-frequency features, which are the most important ones during storm events. Last but not least, the performance decays quickly with prolonged forecast times.

PhyDNet is a neural network (NN) developed for a general video prediction, where the underlying dynamics governing the system are unknown. However, with the long history of weather forecasting [2], this is not the case for precipitation. In this thesis, we aim to progress in addressing the issues mentioned above by exploiting the prior knowledge of precipitation physics. This work will explore the human knowledge of the atmosphere and how it may be used to enhance the physical part of the prediction in PhyDNet. Subsequently, models incorporating the proposed changes will be trained on a radar echo dataset and compared to a PhyDNet baseline. The results will be thoroughly analyzed and discussed.

The first two chapters (1, 2) introduce the reader to precipitation nowcasting, providing necessary theoretical background and exploring related work. In Chapter 3, the architecture of PhyDNet is described in detail, and the precipitation nowcasting problem is formulated, alongside the description of data used, in Chapter 4. Chapter 5 contains proposed changes to PhyDNet. Finally, the results of the concluded experiments are summarized and discussed in Chapter 6.

Chapter 1

Theoretical Background

This chapter provides theoretical background necessary for the thesis. In Section 1.1 we introduce the reader to precipitation, followed by an introduction to weather prediction in Sections 1.2 and 1.3 and weather radars in Section 1.4. The Section 1.5 summarized differential equation concepts needed in this thesis, followed by deep learning concepts in Section 1.6.

1.1 Atmosphere, Precipitation and Convection

The atmosphere is the gaseous envelope of air around the Earth that shields us from outer space (e.g., ultraviolet radiation from the sun or objects coming to Earth from interplanetary space). While the height of the atmosphere is hundreds of kilometers, the weather we are familiar with is happening near the surface, in the troposphere. The troposphere, which is the lowest part, stretches up to the height of 11 km in middle latitudes and contains around 75% of the air mass. It consists of permanent gases, mostly 78% of nitrogen (N_2) and 21% of oxygen (O_2), and variable gases as water vapor (H_2O) and carbon dioxide (CO_2). [1]

Water is present in the atmosphere in all three states – as invisible water vapor, liquid droplets, and solid ice crystals. It continuously circulates in the atmosphere, evaporating from the surface and oceans, condensing into clouds, and falling back to Earth as precipitation. The amount of water vapor in the air is called humidity and can be measured in several ways. Considering an enclosed volume of air in a thin elastic container, called a parcel, the humidity may be measured as:

- *absolute humidity* – the ratio of the weight of the water vapor in the parcel to the volume of the parcel,

- *mixing ratio* – the ratio of the weight of the water vapor to the weight of the remaining dry air in the parcel,

- *relative humidity* – the ratio of the amount of water vapor present in the air to the water vapor capacity at a particular temperature. [1]

Air has a limited capacity of how much water vapor it can take in before it saturates, generally meaning that adding more vapor would lead to its condensation. This capacity changes with air temperature – hotter air may hold more water vapor. Based on this, it is common to refer to humidity in terms of a *dew point* – the temperature to which the air needs to be cooled to start the condensation of the water vapor currently in it. [1]

During condensation, water creates small visible droplets, called fog or clouds. This process releases latent heat necessary for phase change from gas to liquid into the atmosphere and consequently heats it. The opposite, evaporation, is a cooling process as the heat is consumed by water molecules changing to the gas state. Thus, water transportation is an important factor for heat transfer in the atmosphere. [1]

Clouds form when the air is cooled to its dew point, and condensation nuclei are present (aerosol particles that become centers of water droplets), typically as a parcel of air rises through the atmosphere. It behaves similarly to an adiabatic process – air pressure decreases with height, which allows the rising parcel to expand (increase its volume), lowering pressure and temperature. Likewise, sinking parcel compresses, increasing its pressure and temperature. While the parcel contains only water vapor, it changes its temperature by 1°C per 100 m of height, which is called *dry adiabatic lapse rate* (DALR). If the rising parcel is cooled below its dew point, part of the cooling is offset by the latent heat released by condensation, and it consequently cools by *saturated adiabatic lapse rate* (SALR) – approximately 0.6°C per 100 m of height. A typical cloud droplet has an average diameter of 0.02 mm, requiring only slight upward air currents (updraft) to prevent its fall. Through the diffusion, collision-coalescence, and ice-crystal processes (ice crystal riming and aggregation), the droplets may grow enough to overcome the updraft and fall as precipitation, such as rain, snow, sleet, or hail. [1]

Common causes for the rise of an air parcel and cloud formation are shown in Figure 1.1. When moving air encounters an obstacle, such as a mountain range, it is forced to lift along the slopes. The ascending air on the upwind side cools, and the water inside condenses. If it falls as precipitation, the descending air on the downwind side will be warmer due to the latent heat released. Secondly, when two different masses of air encounter, they push the

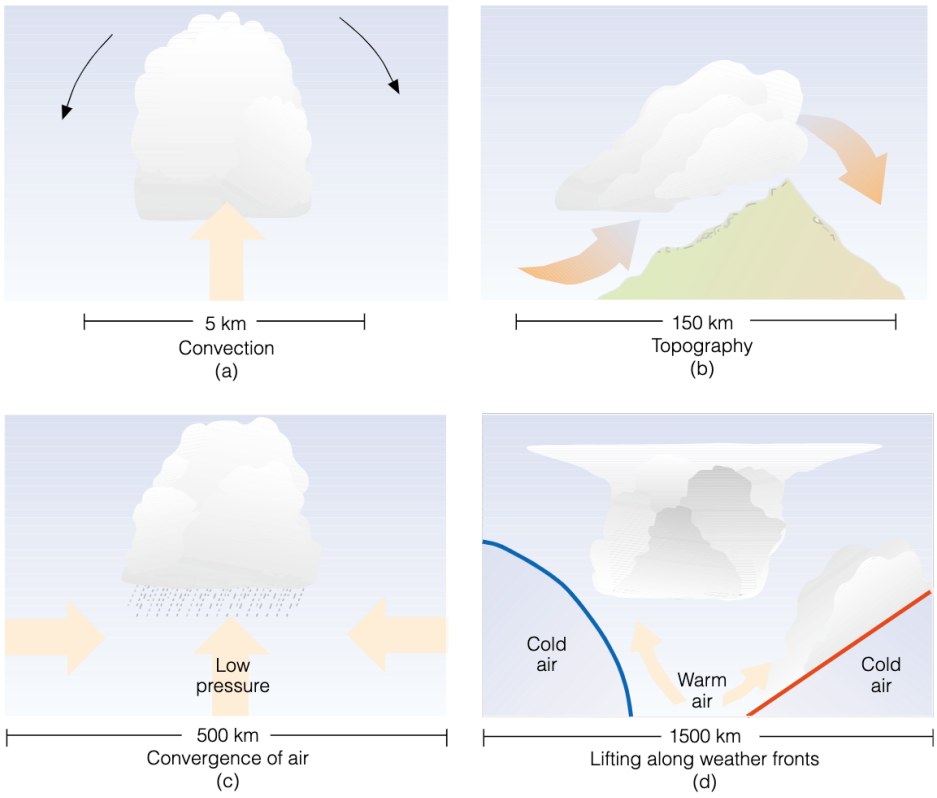


Figure 1.1: The main causes of air ascension and cloud forming. [1]

air between them up. Last but not least, the clouds can be created through *convection*. [1]

Convection is a process of vertical heat transfer that is most important on hot sunny days. When the sun heats the surface unevenly (e.g., a concrete parking lot will absorb more energy than the neighboring forest), an air parcel hotter than its surroundings is created. It expands, becomes less dense, and consequently starts to rise, cooling along its way by dry adiabatic rate. The altitude to which the air parcel ascends depends on the atmosphere's environmental lapse rate and vertical stability. The environmental lapse rate describes the vertical profile of temperature in the atmosphere in 1°C/100 m. The atmosphere is stable if the environmental lapse rate is smaller than SALR – ascending air parcel cools faster than the atmosphere and consequently sinks back to the surface. In an unstable atmosphere, the environmental lapse rate is higher than DALR, meaning a dislocated air parcel will remain hotter than the surrounding air and rise. Finally, the atmosphere may be conditionally unstable, meaning that a parcel lifted enough to start condensation continues its way upwards but returns down if it remains dry. Cooler, heavier air flows

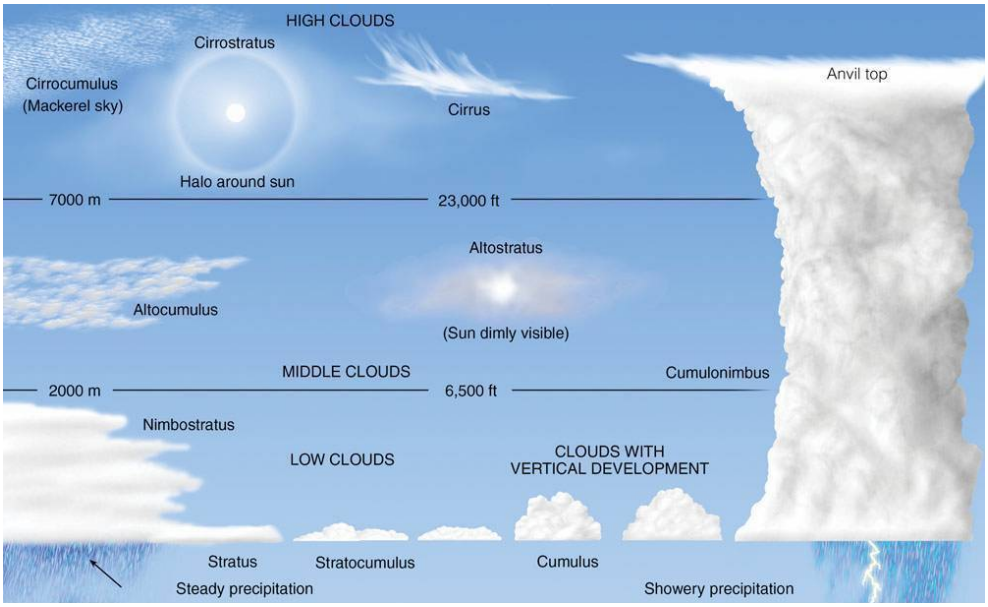


Figure 1.2: Illustration of general cloud types. Precipitation occurs in Cumulonimbus and Nimbostratus clouds. [1]

in to replace the lifted air parcel, creating a *convective circulation*. It is a very localized process that happens only on a scale of a few kilometers (Figure 1.1). They may be several neighboring convective currents during a hot day. The term convection is sometimes confused with *advection* – horizontal movement of air (wind). [1]

Clouds forming through convection in an unstable atmosphere (e.g., cumulonimbus) produce heavy short-duration rain showers that may turn into thunderstorms with hail and lightning. On the other hand, clouds forming in a stable atmosphere tend to spread horizontally (e.g., nimbostratus), resulting in long-lasted steady precipitation. [1]

1.2 Numerical Weather Prediction

Traditional multi-day weather forecasts are computed using numerical weather prediction (NWP) models. The basic concept of NWP is that future weather can be predicted by applying the laws of physics to the current weather observations. It is an initial value problem where forecasts are simulated from the observed state of the atmosphere by integrating partial differential equations (PDEs) governing the dynamics in it. This audacious idea dates back to the beginning of the 20th century when there were too few routine atmospheric observations and no computers. The first real-time forecasts were computed in 1954 in Stockholm, but only in the 1970s, the supercomputing power has

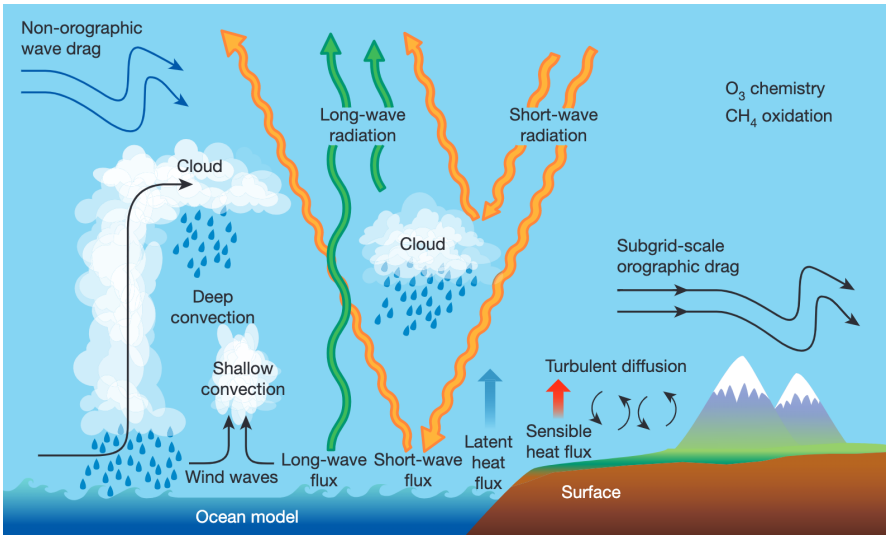


Figure 1.3: Processes that cannot be resolved on traditional model grid scale but are important for accurate weather prediction. They are represented via parameterizations in the NWP. [2]

made it feasible to solve the full set of originally proposed equations. Over the past decades, various technological and scientific advancements led to increased forecast skills, but the paradigm stayed the same. [2]

According to [9], there are seven basic equations governing the speed, temperature, pressure, mixing ratio of water vapor, and density of the atmosphere. These equations, sometimes referred to as *prognostic equations*, should, in theory, fully describe processes in the atmosphere. However, it is mathematically intractable to obtain their analytical solutions, and spatial and temporal discretizations are needed to solve them numerically. Only part of the dynamics is represented in the selected discrete step and scale, also called resolved. The processes, which occur on smaller scales, such as evaporation and condensation, or heat and momentum originating from friction, are unresolved in the discretization. To account for the effect of the unresolved processes, they enter the equations as a source or sink terms, parametrized using resolved variables. [2, 9]

To simulate weather for the whole globe in a reasonable time, the size of the grid cell needs to be 10 km or larger. These scales do not resolve the radiative, convective, and diffusive processes in the atmosphere or at the atmosphere-surface interface. However, their effects are important for the prediction (Figure 1.3). Thus, representing the basic physics in the model through parameterization is an important step for accurate predictions of the atmosphere. The degree of parametric formulation needed differs across processes. While

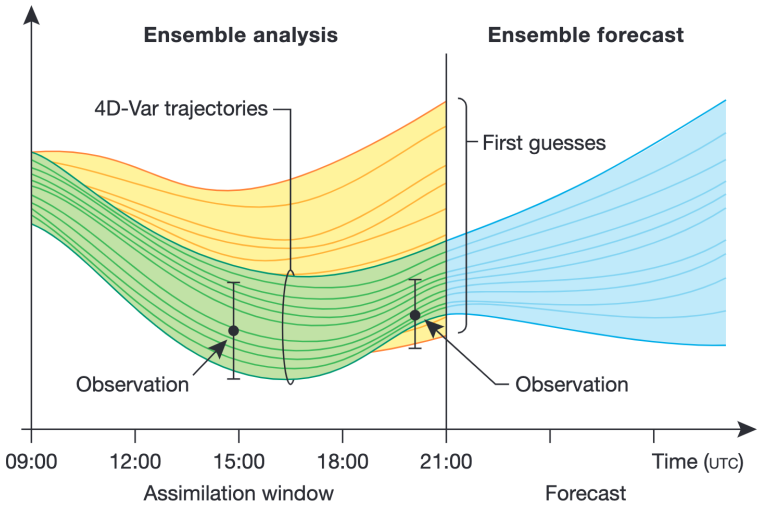


Figure 1.4: 4D-Var data assimilation technique and ensemble forecasting. A small uncertainty in the initial conditions may lead to completely different forecasts. [2]

cloud microphysics is similar, independent of the spatial scale, processes like deep convection occur only in one grid point or between them. Thus, the respective parametrization needs to be adjusted for every scale. [2]

For regional NWP models, predicting the weather for a smaller area, the resolution may be increased to $\sim$ 1 km per grid cell. Most of the mesoscale processes are resolved here, allowing their simulation with much higher accuracy. However, regional model computation times still range from tens of minutes to hours. [9, 10]

1.2.1 Model Initialization and Ensemble Forecasts

The numerical simulations of the atmosphere described above are deterministic given the input state. However, the system is chaotic, as discovered by Edward Lorenz in the 1960s. Lorenz found that exceedingly small perturbations in the input state lead after two weeks to predictions as different as if they were randomly chosen system states. Thus, two tasks arise. Obtaining an as good representation of the input state as possible and quantifying the uncertainty in the prediction. [9]

The chaotic nature of the atmosphere results in a decreasing skill of the forecasts with prediction time. As it is naive to assume a perfect measurement of the state of the atmosphere, performing deterministic forecasts gives only very limited information about the future. Moreover, due to the non-linear complexity of the system, assigning uncertainty to predictions via statistical

methods is inadequate. The accepted and operationally deployed solution is to compute an ensemble of predictions. A set of complete simulations of the system is started from several slightly perturbed initial states. Based on the predictability of the current weather, the obtained forecasts will be more or less different, providing a notion of the uncertainty of the predictions. This set of deterministic forecasts is then used to estimate the probability distribution of the state of the atmosphere at some point in the future (Figure 1.4). [9]

The main problem in determining the initial conditions for the forecast model is that there are observations available only for part of it, and they are distributed non-uniformly in space and time. The number of observations is several orders of magnitude lower than the number of model state degrees of freedom, which are decided by the domain, grid-scale, and modeled variables. The first approaches to filling these gaps in the grid were simple interpolation methods that evolved into complex data assimilation techniques, such as 4D-Var, used in operational forecasts today. [9, 2]

The four-dimensional variational data assimilation technique (4D-Var) uses ensemble modeling in two steps – analysis and forecasts. Short-range forecasts are first computed (e.g., for 6 hours), starting from some set of initial states[2], obtaining an ensemble of possible trajectories during this time window. Analysis of the current state of the atmosphere is selected from these trajectories based on their difference from the newly available observations (Figure 1.4). Afterward, a set of these analyses initialize the actual ensemble forecast. Ensemble analyses and forecasts are used in cycles, always keeping a model representation of the current state of the atmosphere and assimilating observations into it. [2]

1.3 Weather Nowcasting

Nowcasting is short-term forecasting of the weather, defined by the World Meteorological Organization (WMO) as: "*forecasting with local detail, by any method, over a period from the present to 6 hours ahead, including a detailed description of the present weather,*" [11].

Thanks to the recent advances in communication technologies, nowcasts may be effectively utilized by air traffic control, power utilities, or anyone conducting their work outdoors. However, the primary motivation behind nowcasting is to provide tools for informed decision-making during high-impact small-scale weather events, such as thunderstorms and related hazards. The time of an issued warning is an essential factor, as a warning is only useful if it pro-

[2]Commonly taken from a previous run of this or some larger domain NWP model.

vides the receiver with enough lead time to take action. Thus, high resolution and real-time availability are important traits of nowcasting tools.[11]

1.4 Weather Radars

Radar is a system for measuring the distance to an object based on the object's ability to reflect electromagnetic pulses. Weather radar transmits pulses for every rotation and several elevations, always waiting and receiving reflected signals. As the surface of water droplets is electrically leading, it reflects a portion of the energy from the transmitted pulse. When the radar receives the reflected signal, the exact position of the water droplets in the atmosphere may be determined based on the pulse direction and time. Thus, weather radars provide a tool for real-time and high-resolution precipitation monitoring. Radars have been detailly discussed in the Bachelor's thesis [7] that serves as a prelude to this work.

1.5 Differential Equations

The objective of many physical problems is to understand the relationship between changing quantities – to find out how is a target variable changing with respect to others on which it is dependent. In mathematics, the rate of change is expressed through derivatives. Thus, mathematical models of physical systems often involve derivatives of the unknown function, and when they do, they are called differential equations. [12]

Considering an unknown function y of a variable x the simplest differential equation is of the form

$$\frac{\partial y}{\partial x} = f(x), \tag{1.1}$$

describing a relation between the two variables, based on a known function f of x. This equation is called an *ordinary differential equation* (ODE) as the unknown function y depends only on one variable x. If it depended on several variables and the equation contained partial derivatives of them, it would be called *partial differential equation* (PDE). The order of the highest derivative included is the order of the equation. [12]

The desirable outcome of analysis of differential equation

$$\frac{\partial^n y}{\partial x^n}(x) = f\left(x, \frac{\partial y}{\partial x}, \ldots, \frac{\partial^{n-1} y}{\partial x^{n-1}}\right), \tag{1.2}$$

where y is an n times differentiable unknown function of interest and f is a known function, is to find an analytical solution. A function y satisfying

Equation 1.2 for all x in an open interval $x \in (a, b)$. Considering an example first-order ODE

$$\frac{\partial y}{\partial x} = x^3, \tag{1.3}$$

the analytical solution is

$$y = \frac{x^4}{4} + c, \tag{1.4}$$

for an arbitrary constant c. Equation like these are called *initial value problems* – the constant c can be exactly determined by the initial conditions $y(0) = \alpha$, where α is a know value.[12]

When it is not possible to find an analytical solution, numerical methods need to be used, out of which most simple is *forward Euler's method*. Consider an initial value problem

$$\frac{\partial y}{\partial x} = f(x, y), \quad y(x_0) = y_0. \tag{1.5}$$

The value of y at point x_n can be obtained by computing a sequence $(y_0, \ldots, y_n)$

$$y_{i+1} = y_i + hf(x_i, y_i), \tag{1.6}$$

at points $(x_0, \ldots, x_n)$, equally spaced by h. [12]

This work deals with PDEs. A higher order PDE of an unknown function $y(t, x)$ is called *linear*, if it can be written in the form

$$a_0(t,x)y + a_1(t,x)\frac{\partial y}{\partial t} + a_2(t,x)\frac{\partial y}{\partial x} + \cdots + a_n(t,x)\frac{\partial^{i+j} y}{\partial t^i \partial x^j} = b(t,x), \tag{1.7}$$

where $a_k(t, x), k \in \{0, \ldots, n\}$ and $b(t, x)$ are arbitrary differentiable functions. PDEs not writable in this form are called *non-linear*. [12]

1.5.1 Advection-Diffusion Equation

The advection-diffusion equation, sometimes called convection-diffusion, is the simplest model describing a chemical carried by motion in a fluid and diffusing along the way. The concentration of the observed quantity $h(t, x)$, dependent on time t and location x, is modeled as

$$\frac{\partial h}{\partial t} = \underbrace{-u\frac{\partial h}{\partial x}}_{\text{advection}} + \underbrace{\kappa\frac{\partial^2 h}{\partial x^2}}_{\text{diffusion}}, \tag{1.8}$$

where u is a velocity guiding the advection term and κ is diffusivity. [13]

1.5.2 Navier-Stokes Equations

Navier-Stokes equations are PDEs describing motion of the fluids. The unknown variable is velocity vector field u, describing speed and direction of flow in every point of it. Given the assumption of incompressible flow, the momentum equation may be written as

$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u - \nu\nabla^2 u = -\frac{1}{\rho}\nabla p + g, \tag{1.9}$$

where ν is viscosity of the fluid, ρ is density, p is pressure and g represents external sources. [14]

1.6 Deep Learning

We assume that the reader is familiar with DL, and we summarize here only the concepts most important for the thesis, referencing [3]. In case more information about the topic is requested, it can be studied in [3].

Deep learning is a part of machine learning using neural networks consisting of large number of layers and neurons. The task of supervised deep learning is to automatically learn weights needed for modeling an arbitrary function, represented by training data. Given a batch of training examples $\mathcal{B} = \{(x_i, y_i)|i \in \{1, \ldots, N\}\}$ and weights of the model θ, gradient descent algorithm is used for updating the weights toward minimization of error as

$$\theta \leftarrow \theta - \frac{\alpha}{N}\sum_{i\in\mathcal{B}} \frac{\partial \mathcal{L}^{(i)}(\theta)}{\partial \theta}, \tag{1.10}$$

where $\mathcal{L}^{(i)}(\theta)$ is the error for i-th training sample and α is a learning rate. [3]

The default building block of neural networks is a perceptron. Given a vector of its learned weights $\mathbf{w}$, learned bias b and a vector of input features $\mathbf{x}$, the output o is computed as

$$o = \sigma(\mathbf{x}^\intercal \mathbf{w} + b), \tag{1.11}$$

where σ is a non-linear activation function, such as *sigmoid, ReLU* or *tanh*. As the rest of the computation is linear, the activation is the key component in learning complex functions. [3]

1.6.1 Convolutional Layers

Convolutional layers are designed to work efficiently with image data. They are processing 3D tensors of shape $C \times H \times W$ with learned 3D kernels $C \times k \times k$, where C is the number of image channels, and the rest describes its size. The kernel is slid across the input image, computing cross-correlation at each

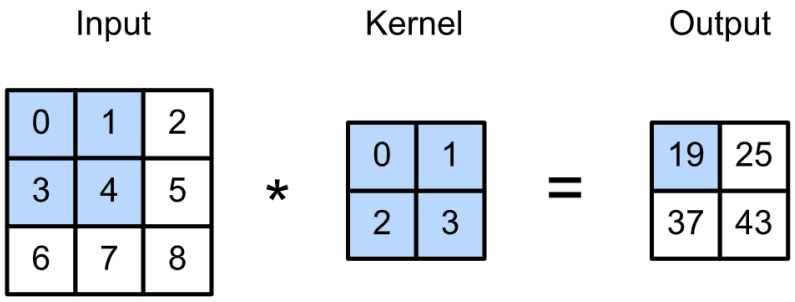


Figure 1.5: An example of 2D convolutional layer. The cross-correlation for the blue output cell is computed as $0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19$. [3]

location (Figure 1.5). It should be noted that from a mathematical point of view, the operation performed is better described as cross-correlation. [3]

1.6.2 Recurrent Neural Networks

Recurrent neural networks (RNN) keep an internal state h_t that allows them to process sequences. Their task is to predict the input state x_t at time t, given a sequence $x_{t-1}, \ldots, x_{t-\tau}$ of past τ measurements. A summary of the past is represented in h_t, which is updated at each step as $h_t = g(h_{t-1}, x_{t-1})$. Thus, the following input state x_t is estimated as $\widehat{x}_t = \arg\max_{x_t} P(x_t|h_t)$. Performing regression on themselves, using the latent representation h_t, these models are called *latent autoregressive models* (Figure 1.6). [3]

One of the most popular RNN architectures is Long Short-Term Memory (LSTM) cell [15], which can learn both long-term dependencies in the processed sequence and between the consecutive steps in it. ConvLSTM, their version for processing spatio-temporal data (a sequence of spatial data – e.g., images), has been introduced in [16] and will be discussed in detail in Section 3.3.

1.6.3 Group Normalization

Group normalization (GN) [17] is an alternative standardization technique to the famous batch normalization [18]. The role of these techniques is to make training faster and more stable to the hyper-parameter changes. When dealing with image data, memory constraints limits the number of samples in a batch. GN's advantage over batch normalization is that it computes statistics over groups of channels, not requiring large batch sizes to work. C channels of the input x are split into g groups, each containing C/g channels. The input is

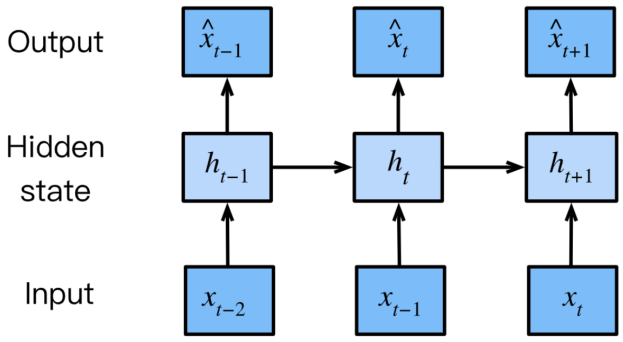


Figure 1.6: RNN – a latent autoregressive model. [3]

then standardized as

$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} \cdot \gamma + \beta, \tag{1.12}$$

where the mean $\mathrm{E}[x]$ and variance $\mathrm{Var}[x]$ are computed over each group of channels and γ, β are learnable parameters. [17]

Chapter 2

Related Work

This chapter contains a brief overview of other work related to precipitation nowcasting and enhancing DL models with prior knowledge of physical processes.

2.1 Precipitation Nowcasting

Regional NWP models (Section 1.2) such as AROME-NWC [10] can explicitly describe convection in the atmosphere and may be used in nowcasting for longer time range $1 - 6$ h. AROME-NWC 6 h forecast is run every hour, on a grid resolution 2.5 km, with a 60 s time step. The objective is to have the predictions available 30 minutes after the analysis time, achieved by a data assimilation window (-45 min, +15 min) and a short 15 min cut-off time. Only observations that become available within the cut-off time window are used in the analysis, making its selection a crucial balancing act between quickly available forecasts and enough data necessary for a good performance.

Real-time high-resolution radar and satellite observations make accurate NWP initializations possible. However, the cost of data assimilation and limitations on the model resolution to maintain computability cause not an optimal use of this data for short-range $0 - 2$ h nowcasting. A traditional approach is to compute nowcasts as an extrapolation on a sequence of radar or satellite measurements. [19]

In *Lagrangian persistence* models, it is assumed that the intensity of precipitation does not change. An advection field (optical flow) is estimated from a sequence of past observations, and the future ones are predicted by advecting the present rainfall. An open-source library containing these models is `rainymotion` [20]. There have been advances, building on the Lagrangian persistence, allowing probabilistic, more accurate nowcasts, such as models

from the library `pySTEPS` [21]. However, the nowcasting of convective initiation, development, and decay remains difficult. [19]

"*Machine learning provides an opportunity to capture complex non-linear spatio-temporal patterns and to combine heterogeneous data sources for use in prediction,*" [19]. The ConvLSTM architecture [16] was initially designed for precipitation nowcasting, and improvements to spatio-temporal prediction were introduced in PredRNN [22]. A Deep Generative Model may be used to predict high-frequency features in the precipitation [23].

2.2 Physics and Deep Learning

Enhancing DL models with a physics prior or a combination of physical modeling and DL can improve the ability of models to generalize to unseen samples, reduce the size of models or help training when not enough training data is available. A good overview of the topic may be found in [14].

The following work influenced this thesis:

- [24] – *Physics informed neural networks* are constrained by physical laws, expressed as general non-linear PDEs. These can learn solutions to supervised training problems data-efficiently while respecting the given laws.
- [25] – The authors present *hidden fluid mechanics*, a DL framework for inference of hidden quantities, like fluid pressure and velocity, from spatio-temporal visualizations of a passive scalar. Passive scalar is transported by the fluid but has no dynamical effect on the fluid motion.
- [4] – PhyDNet is RNN for general video prediction that learns disentanglement between physical and unknown dynamics governing the studied system. It will be discussed in detail in Chapter 3.
- [26] – APHYNITY is a framework for augmenting physical models with DL. The authors present a formulation of the learning problem that allows the physical model to learn as much dynamics as possible.

Chapter 3

PhyDNet Architecture

Our work in precipitation nowcasting models is based on the ideas of Vincent Le Guen and Nicolas Thome, which are presented in [4]. The authors propose to model video data for future frame prediction with a recurrent cell (PhyDNet) that disentangles physical dynamics governing the data from the unknown factors. The approach proposed in the paper builds on the idea of approximation of partial differential equations with convolutional layers and creates a framework for including these equations in DL models. This chapter contains a detailed description of PhyDNet's architecture and design concepts as understood from [4].

3.1 Disentanglement

PhyDNet is a recurrent cell designed for a general prediction of future video frames. Given a frame $\mathbf{u}^{(t)}$, the PhyDNet cell is trained to predict the following frame $\mathbf{u}^{(t+\Delta)}$. Authors work with an assumption that the system, which is captured in the video, can be at least partially described by some physical laws. Following it, they design architecture with two branches (Figure 3.1). The first branch consists of a novel PhyCell that models a broad class of PDEs and handles physical dynamics in the prediction. The second one is a deep ConvLSTM [16] cell handling all the residual dynamics. As the physical laws may not apply at the pixel level of the video, this disentanglement is preceded by a transformation into a conceptual latent space $\mathcal{H}$ that is learned end-to-end by deep convolutional encoder E and decoder D. [4]

The main benefit of this disentangling architecture is the advanced cooperation between physically constrained modeling and prediction by a deep model. Thus, PhyCell leverages physical prior to improve generalization and allows the model to learn prediction dynamics describable by PDE more effectively (with less trainable parameters). ConvLSTM learns the complex unknown

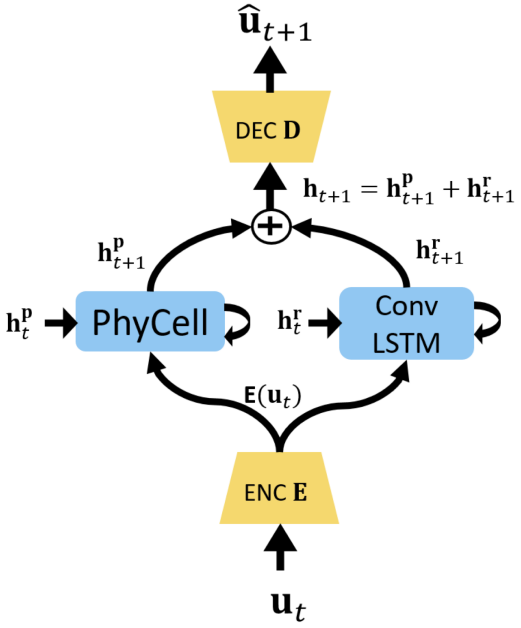


Figure 3.1: Illustration of the PhyDNet cell architecture with two prediction branches. [4]

factors necessary for pixel-level prediction. [4]

In the latent space $\mathcal{H}$, the memory of the PhyDNet cell remembers video up to a time t, in a domain with coordinates $\mathbf{x} = (x, y)$, represented as $\mathbf{h}(t, \mathbf{x}) = \mathbf{h}^{(t)} \in \mathcal{H}$ and linearly disentangled into physical and residual components as $\mathbf{h}^{(t)} = \mathbf{h}_p^{(t)} + \mathbf{h}_r^{(t)}$. Dynamics of the video are then governed by the following PDE:

$$\frac{\partial \mathbf{h}^{(t)}}{\partial t} = \frac{\partial \mathbf{h}_p^{(t)}}{\partial t} + \frac{\partial \mathbf{h}_r^{(t)}}{\partial t} := \mathcal{M}_p(\mathbf{h}_p^{(t)}, E(\mathbf{u}^{(t)})) + \mathcal{M}_r(\mathbf{h}_r^{(t)}, E(\mathbf{u}^{(t)})), \tag{3.1}$$

where $\mathcal{M}_p$ is modeled by PhyCell and $\mathcal{M}_r$ by ConvLSTM. Prediction of the next frame, discretized according to the forward Euler method (Section 1.5), is computed as:

$$\begin{aligned} \hat{\mathbf{u}}^{(t+\Delta)} &= D(\mathbf{h}_p^{(t+\Delta)} + \mathbf{h}_r^{(t+\Delta)}) \\ &= D(\mathbf{h}_p^{(t)} + \mathcal{M}_p(\mathbf{h}_p^{(t)}, E(\mathbf{u}^{(t)})) \\ &\quad + \mathbf{h}_r^{(t)} + \mathcal{M}_r(\mathbf{h}_r^{(t)}, E(\mathbf{u}^{(t)}))), \end{aligned} \tag{3.2}$$

remembering the newly computed hidden states $\mathbf{h}_p^{(t+\Delta)}$ and $\mathbf{h}_r^{(t+\Delta)}$. [4]

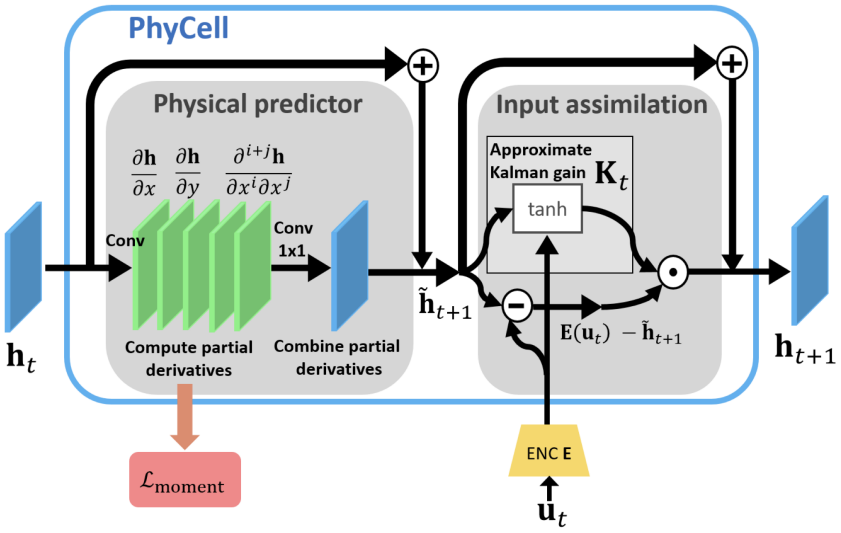


Figure 3.2: Illustration of the PhyCell architecture. The left gray block displays prediction step, the right one is correction. [4]

3.1.1 Data Dimensions

The input video frames are $\mathbf{u}^{(t)} \in \mathbb{R}^{C_u \times H \times W}$, where H, W describe size of the frame and C_u is number of input channels (typically $C_u \in \{1, 3, 4\}$). The dimensions change after the learned transformation into the latent space to $E(\mathbf{u}^{(t)}) \in \mathcal{H} = \mathbb{R}^{C_h \times H_h \times W_h}$. In the default settings $C_h = 64$, height of the transformed domain is $H_h = H/4$ and width $W_h = W/4$. PhyDNet design is fully convolutional; thus, it can handle input images with arbitrary H and W.

3.2 Physical Model – PhyCell

PhyCell (Figure 3.2) is a novel physically constrained recurrent cell introduced in [4] that models the dynamics in two steps:

$$\mathcal{M}_p(\mathbf{h}_p, E(\mathbf{u})) \coloneqq \Phi(\mathbf{h}_p) + C(\mathbf{h}_p, E(\mathbf{u})). \tag{3.3}$$

The first step is prediction in the latent space $\Phi(\mathbf{h}_p)$ (Equation 3.4) that follows the implemented PDE. Then, correction step $C(\mathbf{h}_p, E(\mathbf{u}))$ (Equation 3.8) handles the assimilation of input data into the latent representation similarly as in a Kalman filter [27].

3.2.1 Prediction Step

The physical predictor $\Phi(\mathbf{h}_p)$ models a generic class of **linear** PDEs as

$$\Phi(\mathbf{h}_p^{(t)}) \coloneqq \sum_{i,j<k} c_{i,j} \mathcal{D}_{i,j}(\mathbf{h}_p^{(t)}) = \sum_{i,j<k} c_{i,j} \frac{\partial^{i+j} \mathbf{h}_p}{\partial x^i \partial y^j}(t, \mathbf{x}), \tag{3.4}$$

computing a linear combination of spatial derivatives using learned coefficients $c_{i,j}$. Following the forward Euler discretization, the latent prediction is computed as

$$\tilde{\mathbf{h}}_p^{(t+\Delta)} = \mathbf{h}_p^{(t)} + \Phi(\mathbf{h}_p^{(t)}). \tag{3.5}$$

As this step relies solely on the hidden representation $\mathbf{h}_p^{(t)}$, prediction $\tilde{\mathbf{h}}_p^{(t+\Delta)}$ can be computed even if the input frame $\mathbf{u}^{(t)}$ is not available. [4]

The operation in Equation 3.4 is implemented using two convolutional layers as displayed in the Figure 3.2. The first one θ_1 computes k^2 spatial derivatives as $\phi_1 = \theta_1 \circledast \mathbf{h}_p$, resulting in a tensor $\phi_1 \in \mathbb{R}^{k^2 \times H_h \times W_h}$. This operation is described in detail in the following section 3.2.2. The second layer θ_2 performs the linear combination as convolution $\phi_2 = \theta_2 \circledast \phi_1$. The C_h kernels of θ_2 are sized 1×1, assigning a scalar $c_{i,j} \in \mathbb{R}$ to each partial derivative, represented as a channel in ϕ_1 and performing combination for each spatial position.

3.2.2 Approximation of Partial Derivatives

The partial differential operators $\mathcal{D}_{i,j}(\mathbf{h}_p) = \frac{\partial^{i+j}\mathbf{h}_p}{\partial x^i \partial y^j} = q_{i,j} \circledast \mathbf{h}_p$ are learned through constrained convolutional kernels $q_{i,j}$. The $k \times k$ moment matrix $\mathbf{M}(q_{i,j}) = (m_{a,b})_{k\times k}$ of a $k \times k$ convolutional kernel $q_{i,j}$ is defined as

$$m_{a,b} \coloneqq \frac{1}{a!b!} \sum_{u,v=-\frac{k-1}{2}}^{\frac{k-1}{2}} u^a v^b q_{i,j}[u,v]. \tag{3.6}$$

It is shown in [4] that if $m_{a,b} = 1$ for $a = i, b = j$ and $m_{a,b} = 0$ otherwise, the convolutional kernel $q_{i,j}$ approximates differential operator $\mathcal{D}_{i,j}$ through finite difference coefficients. [4]

Thus, the correct kernels are learned through $\mathcal{L}_m$ moment loss regularization. Defining a $k \times k$ matrix $\mathbf{\Delta}_{i,j}^k$, which equals 1 at position (i,j) and 0 elsewhere, the regularization term is computed as

$$\mathcal{L}_m = \sum_{i,j \le k} ||\mathbf{M}(q_{i,j}) - \mathbf{\Delta}_{i,j}^k||_F, \tag{3.7}$$

where $||\cdot||_F$ is Frobenius norm. [4]

3.2.3 Correction Step

The correction step $C(\mathbf{h}_p, E(\mathbf{u}))$, guiding the assimilation of latent prediction and input data, is defined as

$$C(\mathbf{h}_p^{(t)}, E(\mathbf{u}^{(t)})) \coloneqq \mathbf{K}^{(t)} \odot (E(\mathbf{u}^{(t)}) - (\mathbf{h}_p^{(t)} + \Phi(\mathbf{h}_p^{(t)}))). \tag{3.8}$$

Using this equation discretized according to forward Euler method, it is possible to derive the whole computation of the new hidden state of PhyCell as

$$\begin{aligned}\mathbf{h}_p^{(t+\Delta)} &= \mathbf{h}_p^{(t)} + \Phi(\mathbf{h}_p^{(t)}) + C(\mathbf{h}_p^{(t)}, E(\mathbf{u}^{(t)})) \\ &= \tilde{\mathbf{h}}_p^{(t+\Delta)} + \mathbf{K}^{(t)} \odot (E(\mathbf{u}^{(t)}) - \tilde{\mathbf{h}}_p^{(t+\Delta)}) \\ &= (1 - \mathbf{K}^{(t)}) \odot \tilde{\mathbf{h}}_p^{(t+\Delta)} + \mathbf{K}^{(t)} \odot E(\mathbf{u}^{(t)}).\end{aligned} \tag{3.9}$$

The gating factor $\mathbf{K}^{(t)} \in [0, 1]$ in these two equations, can be interpreted as a Kalman gain controlling the trade-off between the prediction and correction steps. When $\mathbf{K}^{(t)} = 0$, the input frame $\mathbf{u}^{(t)}$ has no contribution to the computed hidden state $\mathbf{h}_p^{(t+\Delta)}$. On the contrary, if $\mathbf{K}^{(t)} = 1$, the whole latent prediction $\tilde{\mathbf{h}}_p^{(t+\Delta)}$ is discarded and the hidden state $\mathbf{h}_p^{(t+\Delta)}$ is reseted according to the input. [4]

The Kalman gain $\mathbf{K}^{(t)}$ is computed using two convolutional layers θ_3, θ_4 as $\mathbf{K}^{(t)} = \theta_3 \circledast \tilde{\mathbf{h}}_p^{(t+\Delta)} + \theta_4 \circledast E(\mathbf{u}^{(t)})$.

3.3 Residual Model – ConvLSTM

The unknown phenomena in the video dynamics that are not corresponding to prior models are learned entirely from the data as the residual dynamics $\mathcal{M}_r(\mathbf{h}_r, E(\mathbf{u}))$ (Equation 3.1). This task is handled by a deep neural network, such as ConvLSTM [16], which was used by the PhyDNet authors in [4].

ConvLSTM [16] extends the idea of LSTM recurrent cell to spatio-temporal data. Originally designed for radar echo precipitation fields, ConvLSTM works with sequences of images, where each image is a 3D tensor – first dimension denotes number of channels, the other two are spatial. In the cell, vector products of the originally fully connected layers are replaced by convolutions as ($\circledast$ denotes the convolution operation, $\odot$ denotes Hadamard product and σ is the sigmoid activation function)

$$\begin{aligned}i^{(t)} &= \sigma\left(W_{xi} \circledast \mathbf{u}^{(t)} + W_{hi} \circledast \mathbf{h}_r^{(t)} + W_{ci} \odot \mathcal{C}^{(t)} + b_i\right) \\ f^{(t)} &= \sigma\left(W_{xf} \circledast \mathbf{u}^{(t)} + W_{hf} \circledast \mathbf{h}_r^{(t)} + W_{cf} \odot \mathcal{C}^{(t)} + b_f\right) \\ \mathcal{C}^{(t+\Delta)} &= f^{(t)} \odot \mathcal{C}^{(t)} + i^{(t)} \odot \tanh\left(W_{xc} \circledast \mathbf{u}^{(t)} + W_{hc} \circledast \mathbf{h}_r^{(t)} + b_c\right) \\ o^{(t)} &= \sigma\left(W_{xo} \circledast \mathbf{u}^{(t)} + W_{ho} \circledast \mathbf{h}_r^{(t)} + W_{co} \odot \mathcal{C}^{(t+\Delta)} + b_o\right) \\ \mathbf{h}_r^{(t+\Delta)} &= o^{(t)} \odot \tanh\left(\mathcal{C}^{(t+\Delta)}\right)\end{aligned} \tag{3.10}$$

These equations follow notation of this chapter, where $\mathbf{u}^{(t)}$ is the observed image, $\mathcal{C}^{(t)}$ is the cell state and $\mathbf{h}_r^{(t+\Delta)}$ is both the new hidden state and

prediction of the ConvLSTM. $i^{(t)}, f^{(t)}, o^{(t)}$ are gates deciding how is the data processed inside the cell. Finally, $W_{\square}$ are weights and $b_{\square}$ are biases of the particular layers. [16]

Chapter 4

Problem Setup

Precipitation nowcasting, formulated as a spatio-temporal prediction of atmospheric measurement sequences, poses an ideal problem for ML models, thanks to the large amounts of real-time, well-defined, and relatively clean data. This chapter formally defines the precipitation nowcasting problem explored and elaborates on the data and tools used.

4.1 Nowcasting Problem Formulation

We formulate the precipitation nowcasting task as a sequence to sequence prediction of tensors $\mathbf{\Psi}^{(T)} \in \mathbb{R}^{C \times H \times W}$, describing the state of the atmosphere at a given time T, with a constant time step Δ. $\mathbf{\Psi}^{(T)}$ is a $3D$ tensor, where H and W are respectively the height and width of the prediction domain, and C is the number of different data channels.

Given a sequence of past τ_I measurements $(\mathbf{\Psi}^{(T-(\tau_I-1)\Delta)}, \ldots, \mathbf{\Psi}^{(T)})$ up to a time T, the task is to predict τ_O future ones as

$$(\widehat{\mathbf{\Psi}}^{(T+\Delta)}, \ldots, \widehat{\mathbf{\Psi}}^{(T+\tau_O\Delta)}) = \tag{4.1}$$
$$\underset{(\mathbf{\Psi}^{(T+\Delta)}, \ldots, \mathbf{\Psi}^{(T+\tau_O\Delta)})}{\arg\max} P((\mathbf{\Psi}^{(T+\Delta)}, \ldots, \mathbf{\Psi}^{(T+\tau_O\Delta)}) | (\mathbf{\Psi}^{(T-(\tau_I-1)\Delta)}, \ldots, \mathbf{\Psi}^{(T)})),$$

where $\widehat{\mathbf{\Psi}}^{(t)} \in \mathbb{R}^{C_O \times H \times W}$ is a prediction of future precipitation fields for each timestamp $t \in (T+\Delta, \ldots, T+\tau_O\Delta)$. The number of input C and output C_O data channels may differ.

4.1.1 Deep Learning Approach

We approach this task using convolutional RNN $\mathcal{F}$ predicting the sequence of future states as a regression

$$\mathcal{F}((\mathbf{\Psi}^{(T-(\tau_I-1)\Delta)}, \ldots, \mathbf{\Psi}^{(T)}), \theta_F) = (\widehat{\mathbf{\Psi}}^{(T+\Delta)}, \ldots, \widehat{\mathbf{\Psi}}^{(T+\tau_O\Delta)}), \tag{4.2}$$

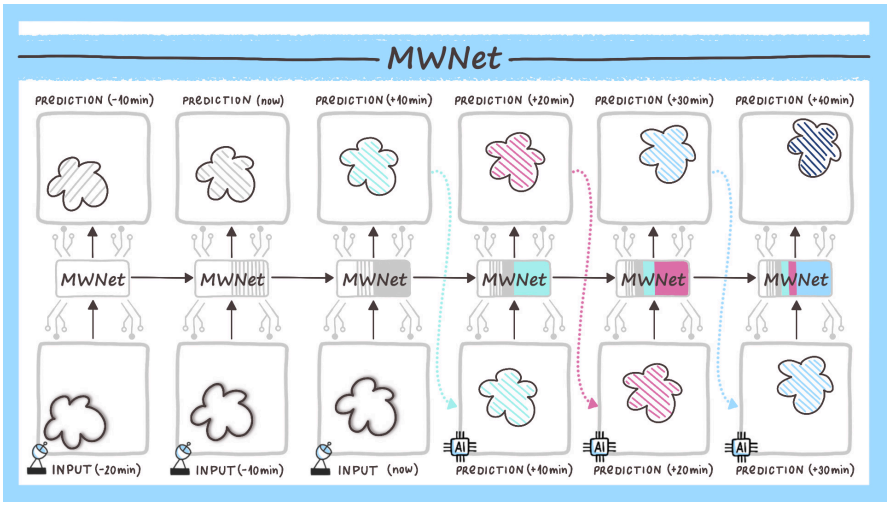


Figure 4.1: Illustration of sequence to sequence prediction with a recurrent cell. Image courtesy of Meteopress.

where θ_F are trainable parameters of $\mathcal{F}$'s inner recurrent cell F. This cell is trained to predict the nearest future state as

$$F(\mathbf{\Psi}^{(T)}, \theta_F, \mathcal{R}_F) = \widehat{\mathbf{\Psi}}^{(T+\Delta)}, \tag{4.3}$$

where $\mathcal{R}_F$ is the cell's memory, which is updated after each use of the function. To predict τ_O future states the cell F is used recurrently as illustrated in the Figure 4.1, processing the sequence chronologically. The prediction is discarded during the input states $(\mathbf{\Psi}^{(T-(\tau_I-1)\Delta)}, \ldots, \mathbf{\Psi}^{(T-\Delta)})$, learning just the inner representation of the seen precipitation situation $\mathcal{R}_F$. The state $\widehat{\mathbf{\Psi}}^{(T+\Delta)}$ for the first lead time is predicted according to the Equation 4.3, and the following $\tau_O - 1$ predictions are computed as

$$F(\widehat{\mathbf{\Psi}}^{(T+(i-1)\Delta)}, \theta_F, \mathcal{R}_F) = \widehat{\mathbf{\Psi}}^{(T+i\Delta)}, \tag{4.4}$$

for $i \in \{2, \ldots, \tau_O\}$.

$\mathcal{F}$ is a supervised ML model, meaning that its parameters θ_F are learned on a dataset of N training samples $\mathcal{D}_{Tr} = \{\mathbf{X}^{(i)}\}_{i=\{1:N\}}$. Each sample $\mathbf{X}^{(i)} = (\mathbf{\Psi}^{(T-(\tau_I-1)\Delta)}, \ldots, \mathbf{\Psi}^{(T)}, \ldots, \mathbf{\Psi}^{(T+\tau_O\Delta)})$ is a sequence of $\tau_I + \tau_O$ atmospheric measurements, from which the first τ_I measurements $\mathbf{X}_I^{(i)}$ are used as an input to the model and the following τ_O as ground truth $\mathbf{X}_O^{(i)}$. Model parameters θ_F are learned through optimization of the loss function

$$\begin{aligned} \mathcal{L}(\theta_F, \mathcal{D}_{Tr}) &= \frac{1}{N} \sum_{i \in \{1:N\}} \mathcal{L}_{\text{sample}}(\mathcal{F}(\mathbf{X}_I^{(i)}, \theta_F), \mathbf{X}_O^{(i)}) \\ &= \frac{1}{N} \sum_{i \in \{1:N\}} \left(\frac{1}{\tau_O} \sum_{j \in \{1:\tau_O\}} \mathcal{L}_{\text{step}}(\widehat{\mathbf{\Psi}}^{(T+j\Delta)}, \mathbf{\Psi}^{(T+j\Delta)}) \right). \end{aligned} \tag{4.5}$$

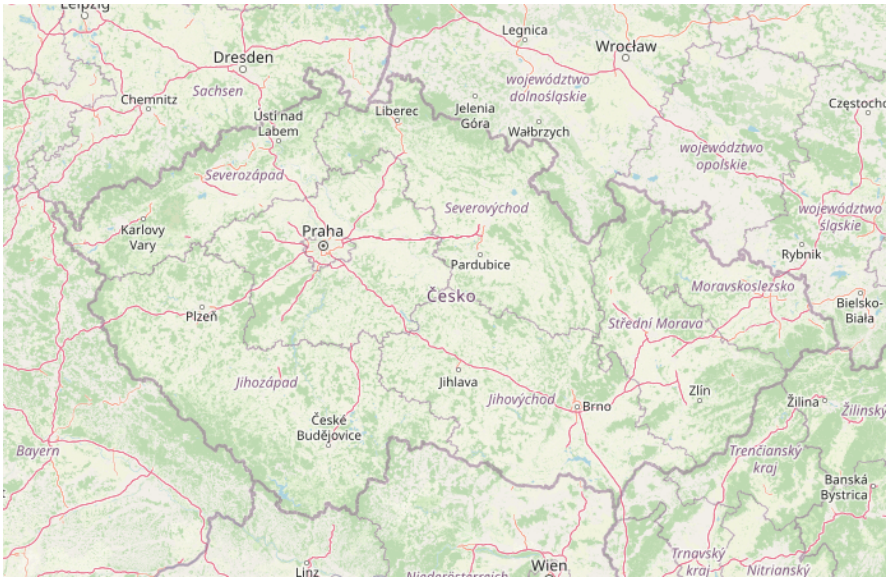


Figure 4.2: Domain of the radar echo images in the dataset, visualized on OpenStreetMatp [5].

4.2 Radar Echo Dataset

The dataset is crucial when approaching any task from the ML perspective. The selected model, which may be capable of fully describing the studied system dynamics, will be useless if the training samples are not representative of the tackled task. Thus, improvements to any ML system are two-fold. On one side is increasing the quality of the samples and dataset as a whole. On the other are changes to the model architecture and its learning process defining how is the information from the samples extracted.

In this work, we look into improvements of our ML precipitation nowcasting model, and its training, considering a fixed dataset of radar echo image sequences (Section 1.4). Thus, only $C = 1$ input channel is used for each measurement $\mathbf{\Psi}^{(T)}$. The source radar echo data comes from the composite images created by and distributed through the OPERA program of EUMETNET [28]. At the time of the dataset creation, Meteopress's archives contained 249176 images from the time window from 2015-10-23 19:30 UTC to 2020-07-21 23:50 UTC with a time step of 10 min. The remaining 403 images from this time range are missing or corrupted. Specifically, the five years from 2015 to 2019 are missing respectively $(98, 174, 66, 34, 31)$ images (Table 4.2).

We have chosen to crop the prediction domain from the composite data to the area of the Czech Republic with small surroundings (Figure 4.2). Our reasoning behind this decision is to develop the models in an area with good radar

coverage and quality measurements, keep the domain small enough for reasonable training times and memory requirements, and finally, a local preference. The WGS 84 coordinates[3] of the domain in degrees are

- North-West corner – 51.397001 N, 11.672296 E,
- South-East corner – 48.223874 N, 19.274628 E.

The images portray measurements above the Earth's surface in the Pseudo-Mercator geographic projection, known from the web mapping applications (projection code EPSG:3857 [29]). We are aware of the potential pitfalls associated with Mercator projection that keeps local shapes but not distances. A more distant pixel from the equator represents a smaller area on the ground than a nearer pixel. In this case, resolution of the data is ~ 0.94 km/px at the lowest latitude of the domain and ~ 0.88 km/px at the highest latitude. In the scope of this work, we do not identify the resolution difference 0.06 km as a problem. However, an eye needs to be kept on the selected geographic projection for larger-scale applications.

4.2.1 Precipitation Intensity

The precipitation intensity is represented in the source data with 8-bit values using the *dBZ* units. The scale is linearly mapping values $[0, 60]$ dBZ to integers in $[0, 255]$, except the 0 dBZ measurement rendered as *no precipitation*. In fact, any measurement from range $(-\infty, 0]$ dBZ is represented as the 0 value. An example of the used mapping may be seen in Table 4.1.

Table 4.1: Precipitation intensity in 8-bit representation, dBZ radar echo measurements and MLdBZ values.

8-bit value	**dBZ**	**MLdBZ**
0	no precipitation	0
1	~ 0.235	~ 0.0039
2	~ 0.471	~ 0.0078
...	...	...
17	4	~ 0.0667
...	...	...
255	60	1

One of the rules of thumb in ML is scaling or standardizing input and output values to small numbers to help stabilize the training process [3]. We have chosen to simply linearly scale the 8-bit integers to float numbers in the range $[0, 1]$, internally called *MLdBZ*.

[3] coordinates used in GPS, EPSG:4326 [29]

For a domain of the size of the Czech Republic, it is not raining every day. Thus, not every radar echo image contains information valuable for the ML model training, and *rainy* images need to be selected.

Definition 1. *A precipitation field* $\mathbf{\Psi}^{(T)}$ *is flagged as* ***rainy*** *if*

- $> 7\%$ *of its area contains non-zero values,*
- *or* $> 1\%$ *of its area has values* > 24 dBZ*.*

After removing clearly noisy samples, we have identified 102873 rainy radar echo measurements in the studied time range.

4.2.2 Dataset Splitting

An ML model is only useful if it is able to generalize – to perform predictions on input samples never seen during the training process. Unfortunately, due to their large expressive capacity, models can sometimes memorize the training samples instead of learning a solution to the task. Despite its great performance on the training dataset, such models become useless in applications. A generally accepted solution for checking this behavior and objectively comparing various models is to split the dataset into three subsets – *train, validation* and *test*, using them isolated in various steps of the model development. [3]

The splitting of the dataset cannot be done arbitrarily. Ideally, the three datasets should be sampled from the same "real-world" distribution. Thus, a not overfitted model should achieve the same quantitative performance over the datasets, and the test scores should be representative of the production ones. Moreover, it is necessary to watch out for any information leakage among the datasets to utilize this technique effectively. When two samples are very similar, they need to be included in the same set. Otherwise, a sample memorized during training may falsely generate performance on other datasets.

With time-series data, such as radar echo sequences, two consecutive samples capture the same atmospheric situation, just slightly shifted. Thus, the samples cannot be split randomly, as memorizing one sample will help predict the following one. The inclusion of these in different datasets would create a false performance. We see two feasible approaches.

Firstly, the dataset may be split by years based on the seasonality of the weather – few years are used in training, the next one for validation, and the next for testing. However, the visualization of mean radar reflectivity and the number of days containing precipitation per month in Figure 4.3 does not show any seasonality throughout the years considered in the dataset. The

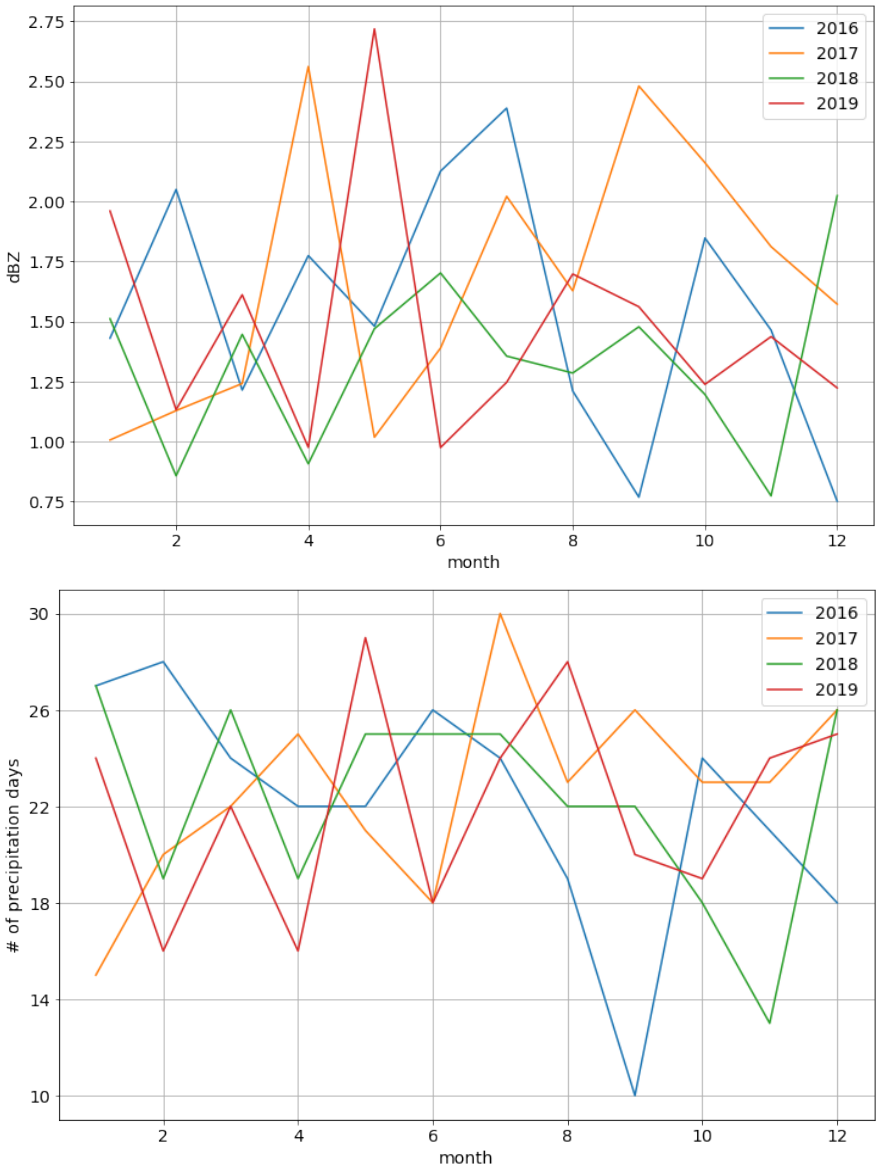


Figure 4.3: Mean radar reflectivity expressed in dBZ (top) and number of days containing precipitation (bottom) per month. Every year is different.

means of reflectivity per year (Table 4.2) tell a similar story of every year being different. These three comparisons, which exclude data from 2015 and 2020 as these years are incomplete, hint that precipitation distribution depends on other variables than the time of the year. Consequently, we think that the assumption of datasets being sampled from the same distribution would be violated if they were split by years.

Table 4.2: Year statistics of radar echo data.

Year	**Mean reflectivity [dBZ]**	**# missing samples**	**# independent situations**
2015	–	98	15
2016	1.542	174	60
2017	1.668	66	53
2018	1.334	34	61
2019	1.481	31	57
2020	–	0	29

The other chosen approach is to identify whole precipitation situations, independent among themselves, and split them randomly into the sets.

Definition 2. *Two rainy precipitation fields* $\mathbf{\Psi}^{(t_a)}$ *and* $\mathbf{\Psi}^{(t_b)}$ *are considered* ***dependent*** *if there is less than 24 hours between them,* $|t_b - t_a| < 24$ hours.

Definition 3. *An* ***independent precipitation situation*** $\mathcal{S}_{T_a}^{T_b}$ *is the shortest chronologically sorted sequence of all precipitation fields between two timestamps* $\mathcal{S}_{T_a}^{T_b} = (\mathbf{\Psi}^{(t)})_{t\in[T_a,T_b]}$*, containing all the precipitation fields dependent with* $\mathbf{\Psi}^{(t)}$ *for every field* $\mathbf{\Psi}^{(t)} \in \mathcal{S}_{T_a}^{T_b}$*.*

Following the Definition 3, two consecutive independent precipitation situations are separated by at least 24 hours without any rainy radar echo measurements. The time delta of 24 hours was chosen empirically to ensure that memorizing a sample from one situation will not help with the prediction from a different one.

We have identified 275 independent precipitation situations in the considered time range with a median length of 78 hours and a mean length of 112 hours. The number of situations per year is displayed in Table 4.2. These situations were split randomly into the train, validation, and test set with ratios and counts summarized in Table 4.3.

Each independent precipitation situation $\mathcal{S}_{T_a}^{T_b}$ consists of a set of training sequences $\{\mathbf{X}(t)\}_{t\in[T_a,T_b]}$, which are added to the corresponding dataset $\mathcal{D}$. The notation $\mathbf{X}(t)$ stands here for a sequence centered around the measurement $\mathbf{\Psi}^{(t)}$ as $\mathbf{X}(t) = (\mathbf{\Psi}^{(t-(\tau_I-1)\Delta)}, \ldots, \mathbf{\Psi}^{(t)}, \ldots, \mathbf{\Psi}^{(t+\tau_O\Delta)})$.

Table 4.3: Dataset statistics.

Dataset	**Situation split percentage**	**# independent situations**	**Hours of precipitation**
Train	72%	198	22724
Validation	$\sim$ 12.7%	35	3678
Test	$\sim$ 15.3%	42	4570

Chapter 5

PhyCell Adjustments for Precipitation Nowcasting

PhyDNet (Chapter 3) is a deep RNN designed to predict future frames of general video sequences. PhyDNet's authors [4] do not assume much about the underlying physical dynamics governing the captured system and let PhyCell learn some of the broad family of PDEs described in the Equation 3.4.

However, in the case of precipitation nowcasting (Section 1.3), there is both some knowledge of precipitation physics and requirements for forecasts to be operationally usable. In this chapter, we propose changes to the PhyDNet's architecture, aiming at utilizing its strengths to the full potential in the context of nowcasting.

PhyDNet is used as the recurrent cell F in the Equation 4.3, where cell's memory $\mathcal{R}_F = (\mathbf{h}_p, \mathbf{h}_r)$ contains hidden states of both physical and residual branches.

5.1 Intensity Classification Loss

One of the primary motivations for implementing precipitation nowcasting systems is the advantage of higher-resolution forecasts during storm events. Most short-time severe weather risks, such as hail, flash floods, strong winds, or lightning, are connected to convective systems and thus high-intensity precipitation. The ability to nowcast storms accurately, even if only dozens of minutes into the future, benefits both the general public and operational meteorologists monitoring these situations. The automatization of nowcasting brings unprecedented forecast localization to the end-users while providing another valuable information source for meteorologists issuing severe weather

alerts.

However, when the training of a neural network is formulated as a regression problem, the model may not be motivated to predict pixels with high intensities and other high-frequency features in general. Due to the chaoticness of weather, the uncertainty of prediction naturally rises with increasing lead time, and it needs to be handled. Given the limited information on the input, it may not be possible to decide whether a storm will move to some point or one 10 km souther. Traditional regression loss functions, such as MSE, are penalizing an incorrect selection of the storm location twice – once for predicting it at a wrong place and the second time for not predicting it at the correct one. To avoid these errors, regression-based models generally learn to express the uncertainty with smoothed-out predictions, entirely omitting high-frequency features in the predictions.

While smoothed out predictions achieve optimal prediction error, they do not provide valuable information during storm events. To emphasize the prediction of high intensities, we propose to create a new output of the model, containing prediction of "probabilities" of severe rainfall over 40 dBZ. According to Equation 4.3, output of one recurrent cell's prediction step is $\widehat{\mathbf{\Psi}}^{(T+\Delta)}$. In the case of the PhyDNet, it is the output of the deep convolutional decoder D that is combining and processing predictions of the two branches (Section 3.1). The new output is computed via a single convolutional layer θ_{prob} with a kernel size 3 as

$$\widehat{\mathbf{\Psi}}_{\text{prob}}^{(T+\Delta)} = \theta_{\text{prob}} \circledast \widehat{\mathbf{\Psi}}^{(T+\Delta)}. \tag{5.1}$$

Considering these two outputs, the error of prediction on one training sample $\mathbf{X}^{(i)}$ (Equation 4.5) decomposes to

$$\begin{aligned}\mathcal{L}_{\text{sample}}(\mathcal{F}(\mathbf{X}_I^{(i)}, \theta_F), \mathbf{X}_O^{(i)}) = \frac{1}{\tau_O} \sum_{j \in \{1:\tau_O\}} & \mathcal{L}_{\text{img}}(\widehat{\mathbf{\Psi}}^{(T+j\Delta)}, \mathbf{\Psi}^{(T+j\Delta)}) + \\ & \mathcal{L}_{\text{icl}}(\widehat{\mathbf{\Psi}}_{\text{prob}}^{(T+j\Delta)}, \mathbf{\Psi}_{\text{prob}}^{(T+j\Delta)}).\end{aligned} \tag{5.2}$$

In this equation, the ground truth $\mathbf{\Psi}_{\text{prob}}^{(T+j\Delta)}$ is a binary image obtained via thresholding of the ground truth $\mathbf{\Psi}^{(T+j\Delta)}$ – each pixel is assigned the value *one* if its intensity is greater than 40 dBZ and *zero* otherwise. The comparison of this binary ground truth and the predicted "probabilities" $\mathcal{L}_{\text{icl}}$ is called *ICLoss* (Intensity Classification Loss) and performed via a cross-entropy loss as implemented in PyTorch[4]. The loss is weighted towards the *one* class with a scaling factor 5 to reduce the imbalance of classes a bit.

[4] https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html

Even though the prediction in a classification task like this can be generally interpreted as a probability of the *one* class, we use quotation marks here. The reason is that while the prediction of severe rainfall probabilities is an interesting product from a meteorological point of view, the primary aim of $\mathcal{L}_{\text{icl}}$ here, is to improve the prediction of high intensities in the original output.

5.2 Non-linearity in the PhyCell

A general aim of NNs is to be able to model a wide variety of functions on compact subsets of $\mathbb{R}^n$. As the perceptron function is linear, this universal function approximation trait is achieved by using a finite number of neurons in single or multiple layers with non-linear activation functions. [3]

Similar is true for CNNs such as the ConvLSTM (Section 3.3) used in the residual branch of the PhyDNet. In the default setting from [4], it is configured with three stacked cells, which are respectively operating on inputs with $(128, 128, 64)$ channels. While this setting theoretically gives it the ability to learn various functions, it is not designed to perform the point-wise multiplication of two images effectively. PhyCell, with its linear prediction step is not designed for multiplication as well (Equation 3.4, repeated here as Equation 5.3 for clarity). However, the multiplication of different physical quantities is a very common operation.

$$\Phi(\mathbf{h}_p^{(t)}) \coloneqq \sum_{i,j<k} c_{i,j} \mathcal{D}_{i,j}(\mathbf{h}_p^{(t)}) = \sum_{i,j<k} c_{i,j} \frac{\partial^{i+j} \mathbf{h}_p}{\partial x^i \partial y^j}(t, \mathbf{x}), \tag{5.3}$$

In this equation, scalars $c_{i,j}$ relevant to particular differential operators are learned during training and shared across all positions $\mathbf{x}$ of the domain. Considering the precipitation nowcasting task, the change of precipitation intensity at all times and all places of the domain would only be linearly dependent on the gradient of the hidden state. Referencing the non-linear advection term in Navier-Stokes equations (Section 1.5.2) for modeling fluids, we identify this linearity as a significant limitation for the correct precipitation prediction.

There is a second weakness in the default settings of PhyCell for the presented use case as well – the parameter limiting the order of the partial derivatives computed is set as $k = 7$. To the best of our knowledge, derivatives of this order and higher are used, for example, in numerical methods trying to achieve the target accuracy of computation, but rarely in equations robustly describing physical phenomenons. For instance, the prognostic equations of NWP (Section 1.2) do not contain higher than second-order derivatives. We believe that this setting reduces the potential of explainability of the predictions, which is a trait highly valued by meteorologists. Moreover, it creates a large space in the PhyCell for loss optimization, possibly reducing the robustness of

PhyCell's predictions and interfering with the task of ConvLSTM. Thus, we use $k = 3$ in our later experiments (limiting to second-order derivatives).

In the following subsections, we propose two different approaches to enable non-linearity in the physical prediction of PhyDNet.

5.2.1 Quadratic Non-linearity Approach

The first approach, which we later started to call *quadratic*, is built on a traditional DL paradigm of letting the deep model learn, what is important for loss optimization. $\Phi(\mathbf{h}_p^{(t)})$ from Equation 5.3 can be described as a first-degree polynomial of spatial partial derivatives. Considering a vector of all k^2 partial derivatives $\mathbf{d}(\mathbf{h}_p^{(t)}) = \left(\mathcal{D}_{0,0}(\mathbf{h}_p^{(t)}), \dots, \mathcal{D}_{k-1,k-1}(\mathbf{h}_p^{(t)})\right)$ up to some hyperparameter k, and vector of learned scalars $\mathbf{c} = (c_{0,0}, \dots, c_{k-1,k-1})$, the prediction equation may be rewritten as[5]

$$\Phi = \mathbf{c} \cdot \mathbf{d}. \tag{5.4}$$

We propose to compute all of the possible second-degree terms through matrix multiplication, learn corresponding scalars and add them to this equation. The matrix of second-degree terms $\mathbf{d}^{(2)}$ is obtained as

$$\mathbf{d}^{(2)} = UT(\mathbf{d}^\intercal \times \mathbf{d}), \tag{5.5}$$

where UT is a function selecting only the $\frac{k^2(k^2+1)}{2}$ upper triangular terms

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ 0 & 1 & \cdots & 1 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \end{pmatrix}$$

to remove duplicates and flattens them by rows to a row vector. A vector of corresponding scalars $\mathbf{c}^{(2)}$ is learned by 1×1 convolution as in the Section 3.2.1 and the prediction equation is extended to

$$\Phi = \mathbf{c} \cdot \mathbf{d} + \mathbf{c}^{(2)} \cdot \mathbf{d}^{(2)}, \tag{5.6}$$

which can be expressed in the expanded form as

$$\begin{aligned} \Phi(\mathbf{h}_p^{(t)}) = \sum_{i,j<k} c_{i,j} \frac{\partial^{i+j} \mathbf{h}_p}{\partial x^i \partial y^j}(t, \mathbf{x}) + \\ \sum_{m,n<k; i \le m; j \le n} c^{(2)}_{i,j,m,n} \frac{\partial^{i+j} \mathbf{h}_p}{\partial x^i \partial y^j}(t, \mathbf{x}) \cdot \frac{\partial^{m+n} \mathbf{h}_p}{\partial x^m \partial y^n}(t, \mathbf{x}). \end{aligned} \tag{5.7}$$

[5] The hidden state $\mathbf{h}_p^{(t)}$, which is input to the Φ, is omitted here for clarity.

5.2.2 Advection-diffusion Equation

The other approach relies on hand-engineering of prior knowledge, using the *advection-diffusion* PDE (Section 1.5.1) to model the precipitation in the PhyCell. The prediction step from Equation 5.3 theoretically changes to

$$\Phi(\mathbf{h}_p^{(t)}) = \underbrace{-c_0 \frac{\partial \mathrm{u}_x \mathbf{h}_p}{\partial x}(t, \mathbf{x}) - c_1 \frac{\partial \mathrm{u}_y \mathbf{h}_p}{\partial y}(t, \mathbf{x})}_{\text{advection}} + \underbrace{c_2 \frac{\partial^2 \mathbf{h}_p}{\partial x^2}(t, \mathbf{x}) + c_3 \frac{\partial^2 \mathbf{h}_p}{\partial y^2}(t, \mathbf{x})}_{\text{diffusion}}, \tag{5.8}$$

where $\mathbf{u} = (u_x, u_y)$ is a vector field by which the precipitation is advected. The original idea is inspired by the work on Hidden Fluid Mechanics by Raissi et al. [25]. They assume that the flow of fluid is observed through a passive scalar that is moved by the flow described by Equation 5.8 but not affecting it.

However, in the case of precipitation nowcasting, we do not aim to infer global advection field, interpretable as wind, that would move precipitation as a passive scalar. We are rather interested in modeling of precipitation local developments. Thus, based on the advection term of Navier-Stokes equations (Section 1.5.2), $\mathbf{u}^{(t)}$ is inferred from the system state $\mathbf{h}_p^{(t)}$, guided just by the use of $\mathbf{u}^{(t)}$ in Equation 5.8. This approach introduces non-linearity to the PhyCell as $\mathbf{u}^{(t)}$ is a function of $\mathbf{h}_p^{(t)}$.

The advection vectors $\mathbf{u}^{(t)}$ at time t are computed with a single convolutional layer θ_U with kernel size 5 as

$$\mathbf{u}^{(t)} = U(\mathbf{h}_p^{(t)}) = \theta_U \circledast \mathbf{h}_p^{(t)}. \tag{5.9}$$

Following the original implementation of PhyCell, partial derivatives are computed with learned differential operators $\mathcal{D}_{i,j}$. Thus, four terms of the implemented PDE are[6]

$$\mathbf{d}(\mathbf{h}_p) = \left(\mathcal{D}_{1,0}\left(U(\mathbf{h}_p)_x \mathbf{h}_p\right), \mathcal{D}_{0,1}\left(U(\mathbf{h}_p)_y \mathbf{h}_p\right), \mathcal{D}_{2,0}\left(\mathbf{h}_p\right), \mathcal{D}_{0,2}\left(\mathbf{h}_p\right)\right), \tag{5.10}$$

which are linearly combined using coefficients $\mathbf{c} = (c_0, \ldots, c_3)$, learned through 1×1 convolution.

Following problems with convergence during training, we have added Group Normalization[7] (GN) to standardize the equation terms as one group. While we are struggling with interpretation of GN in terms of physical simulation, it should be noted that the original implementation of PhyCell[8] uses GN on

[6] Omitting the time index $^{(t)}$ for clarity.

[7] https://pytorch.org/docs/stable/generated/torch.nn.GroupNorm.html

[8] https://github.com/vincent-leguen/PhyDNet

the partial derivatives as well, splitting the 49 terms into 7 groups. Finally, the prediction step is implemented as

$$\Phi(\mathbf{h}_p^{(t)}) = \mathbf{c} \cdot GN(\mathbf{d}(\mathbf{h}_p^{(t)})). \tag{5.11}$$

5.3 Implementation Details

Whole project is implemented in Python 3.8[9]. The neural network, training and evaluation code is written with the libraries PyTorch 1.10 [30] (BSD license) and PyTorch Lightning 1.5[10] (Apache 2.0 license). The implementation of PhyDNet model is derived from the original implementation[11] published with [4] under MIT license. The rest of the submitted code is our work.

The dataset is saved on disk in the form of individual radar echo precipitation fields $\mathbf{\Psi}^{(T)}$ that are loaded to form sequences during the training. Each precipitation field $\mathbf{\Psi}^{(T)}$ is saved as an 8-bit grayscale `.png` file.

All of the training and experiments were computed locally on the computer of Meteopress. The machine is running a Debian[12] operating system on a 12-core AMD Ryzen 9 5900X processor, using two dedicated NVIDIA RTX 3090 GPUs (Graphics Processing Unit), each with 24 GB of RAM (Random Access Memory).

5.4 Settings of the Trained Models

All of the models were trained primarily with $\mathcal{L}_{1,2}$ loss, which we define for one prediction step (Equation 4.5) as the sum of Mean Absolute Error (MAE) and Mean Squared Error (MSE)

$$\begin{aligned}\mathcal{L}_{1,2}(\widehat{\mathbf{\Psi}}^{(T+j\Delta)}, \mathbf{\Psi}^{(T+j\Delta)}) = \\ \frac{1}{H \cdot W} \sum |\widehat{\mathbf{\Psi}}^{(T+j\Delta)} - \mathbf{\Psi}^{(T+j\Delta)}| + \left(\widehat{\mathbf{\Psi}}^{(T+j\Delta)} - \mathbf{\Psi}^{(T+j\Delta)}\right)^2,\end{aligned} \tag{5.12}$$

where the sum is over spatial dimensions. The uniform input length is 10 images and the models are trained to predict the future 6. The resulting models of trainings were selected after the validation loss stabilized, based on the combination of loss and CSI.

See the enclosed CD (Appendix B) for exact definition of model trainings and Appendix C for sample predictions by all models. The PhyDNet models

[9] https://docs.python.org/3.8/
[10] https://www.pytorchlightning.ai/
[11] https://github.com/vincent-leguen/PhyDNet
[12] https://www.debian.org/

use ConvLSTM module with the same configuration, PhyCell models lack the ConvLSTM.

- `PhyCell Baseline` – loss $\mathcal{L}_{1,2} + \mathcal{L}_{\text{icl}}$, differential operators limited by $k = 7$, 49 linear terms in the PDE.
- `PhyCell Quadratic` – loss $\mathcal{L}_{1,2} + \mathcal{L}_{\text{icl}}$, differential operators limited by $k = 3$, 9 linear terms and 45 non-linear in the PDE.
- `PhyCell AdvDiff` – loss $\mathcal{L}_{1,2} + \mathcal{L}_{\text{icl}}$, *advection-diffusion* PDE with 4 terms.

- `PhyDNet Baseline` – loss $\mathcal{L}_{1,2}$, PhyCell differential operators limited by $k = 7$, 49 linear terms in the PDE.
- `PhyDNet ICLoss` – loss $\mathcal{L}_{1,2} + \mathcal{L}_{\text{icl}}$, PhyCell differential operators limited by $k = 7$, 49 linear terms in the PDE.
- `PhyDNet AdvDiff` – loss $\mathcal{L}_{1,2} + \mathcal{L}_{\text{icl}}$, PhyCell with *advection-diffusion* PDE with 4 terms.

Chapter 6

Experiments

This chapter summarizes concluded experiments, results, and findings acquired during PhyDNet adjusting for precipitation nowcasting.

6.1 Intensity Classification Loss

The effects of *ICLoss* (Section 5.1) were studied during the development phase on the validation set and `PhyDNet Baseline` model. Its effects are well seen at Figure 6.1, where the baseline model concentrates on precipitation with larger area but smaller intensity, not predicting the small-area storms at all. On the other hand, `PhyDNet ICLoss` does better job in identification of the storms and does not smooth them out, while the "probability" output correctly marks at least the lower part of the storms.

The quantitative comparison of these two models on the test set (Figure 6.2) displays the same trends as were observed on the validation one. As summarized in Table 6.1 in terms of relative changes to performance, there is almost no difference in achieved MAE, MSE or SSIM. However, the trade-offs of training with *ICLoss* can be seen in other metrics. The decrease in the low-threshold CSI alongside better high-threshold CSI and the slight performance improvement over time, which may seen in the plots, support the impression that predictions are less smoothed out if *ICLoss* is used. The improvement in Kolmogorov-Smirnov distance suggests that focus on intensities over 40 dBZ makes empirical CDFs of the predictions more similar to the ground truth.

Following these observations, we have decided to use *ICLoss* in our later experiments. However, closer inspection of its effects on the sample predictions from the test set shows limitations of the proposed *ICLoss* implementation. Due to setting of the threshold to 40 dBZ, the model tends to quickly lower the predicted intensities to this value. This may be clearly seen in the Figure 6.3,

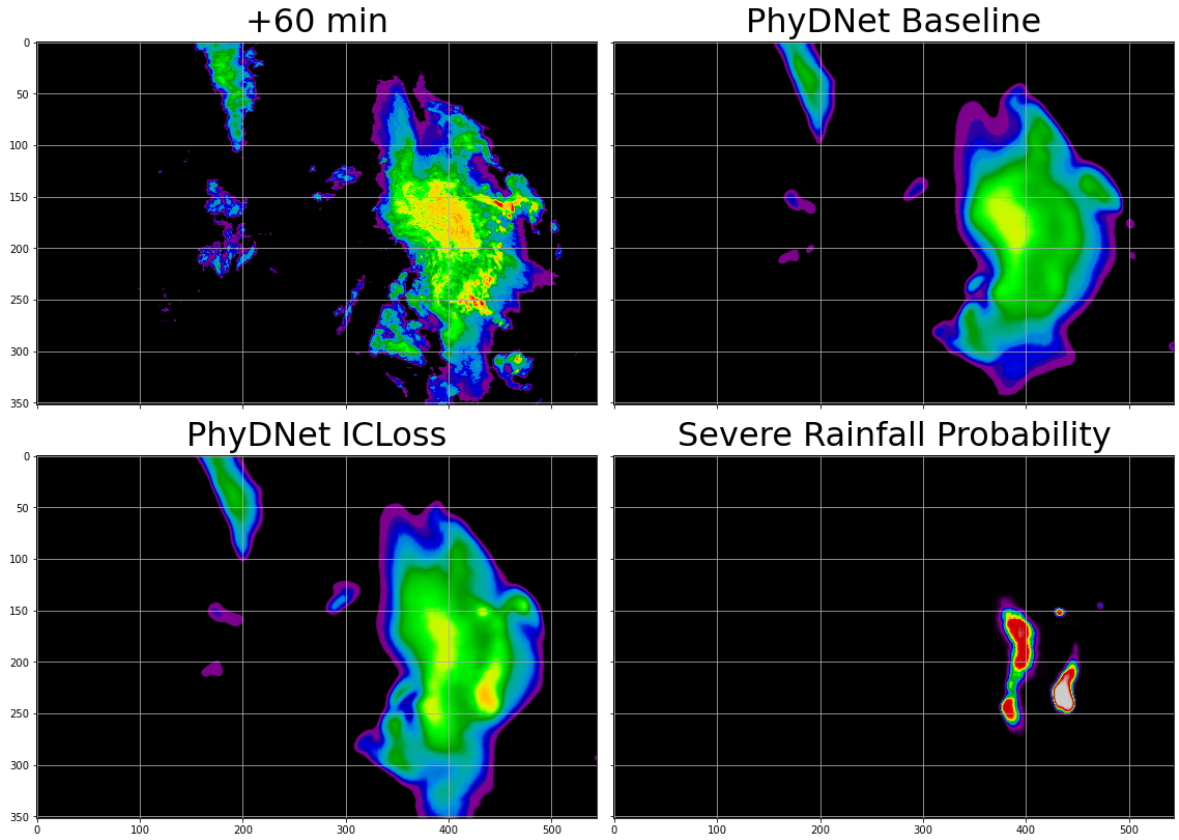


Figure 6.1: Effect of training with *ICLoss* on prediction for 60 min (validation set). By rows: ground truth, baseline prediction, baseline with *ICLoss* prediction and "probability" of severe rainfall.

Table 6.1: Relative change in the metrics of PhyDNet ICLoss compared to PhyDNet Baseline (red denotes performance loss).

CSI 8 dBZ	−2.30 %
CSI 40 dBZ	+6.46 %
MAE	+0.25 %
MSE	−0.38 %
Kolmogorov-Smirnov	−2.13 %
SSIM	−0.22 %

but it happens in the first example as well. Moreover, due to small capacity of the convolutional module producing the "probabilities" (Section 5.1), these outputs lack gradient and very closely resemble the predicted intensities to be truly interpreted as probabilities of severe rainfall.

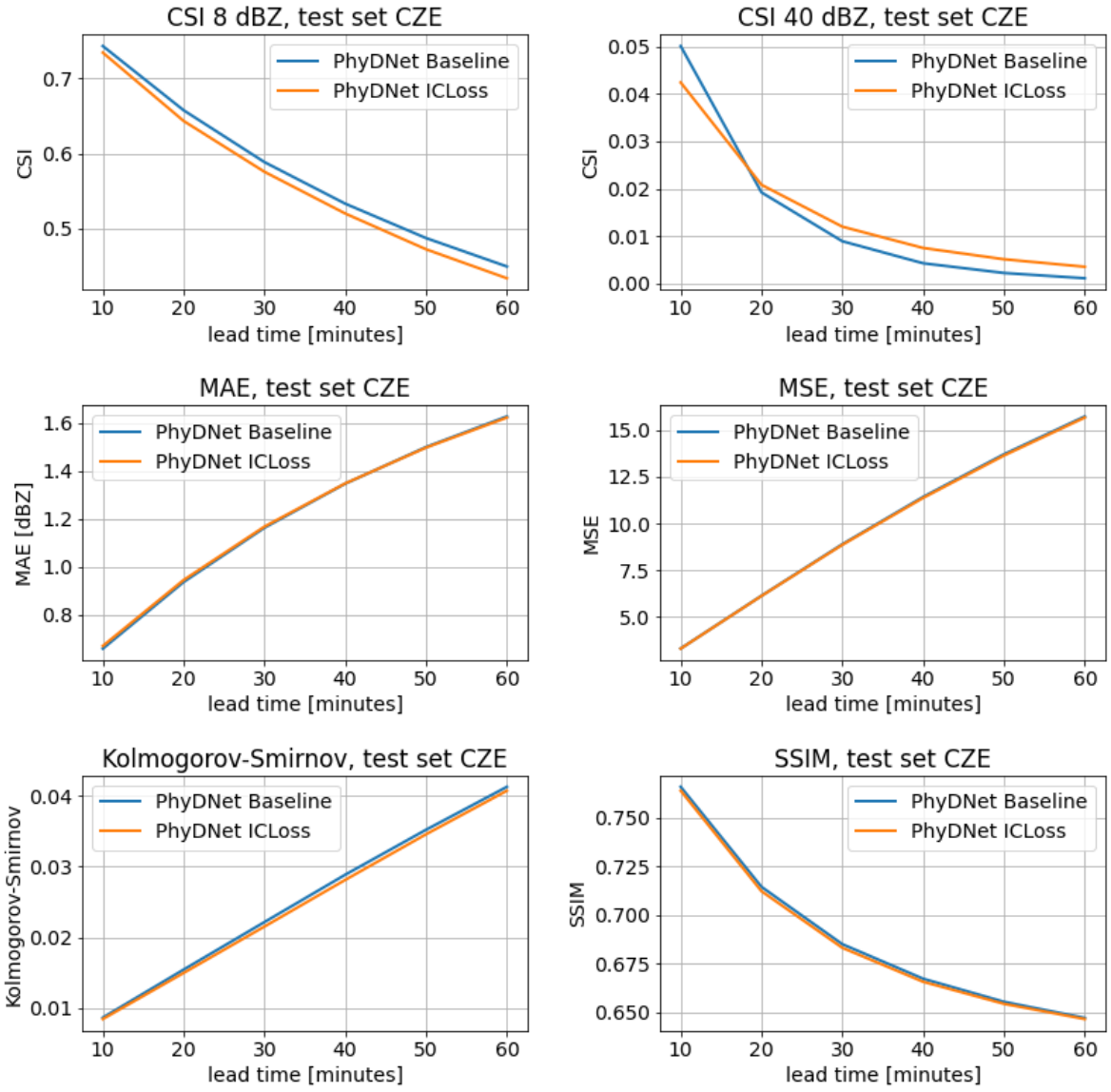


Figure 6.2: Effect of training with *ICLoss* on metrics achieved on the test set for 60 min predictions.

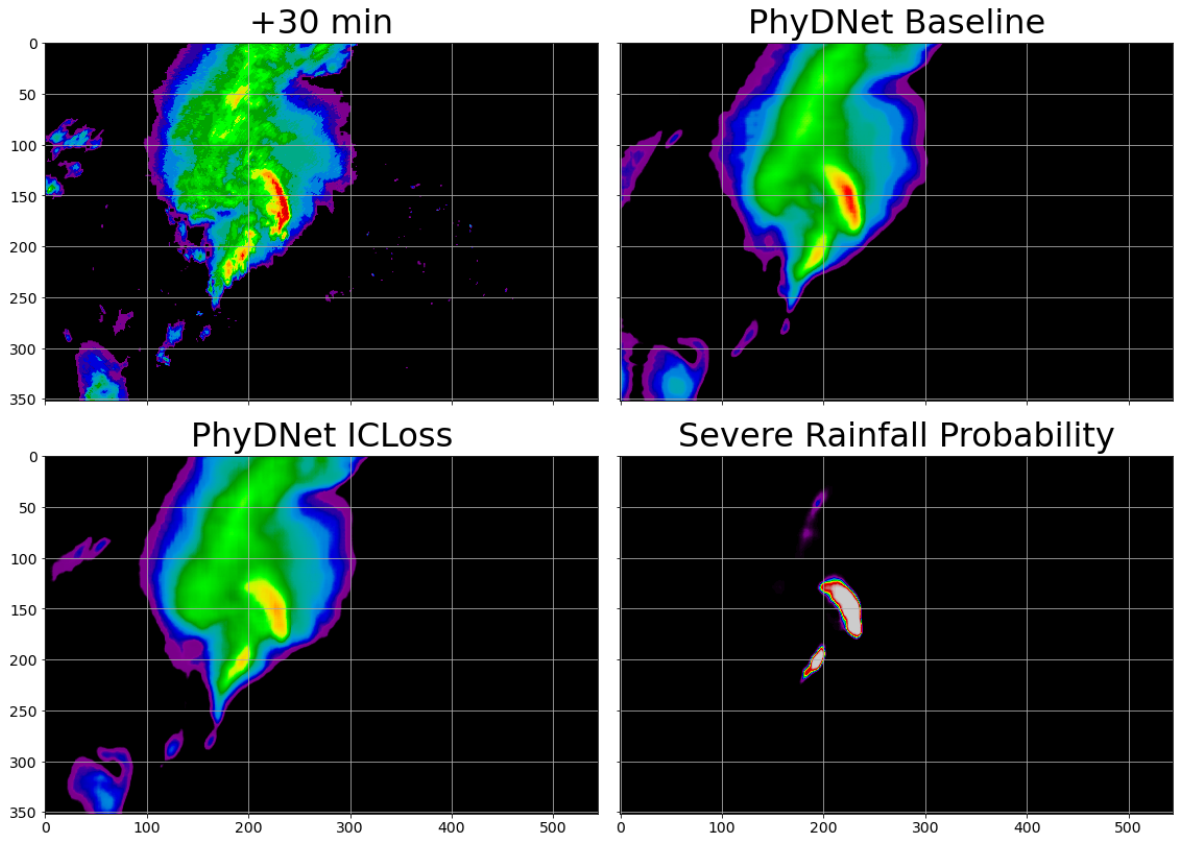


Figure 6.3: Effect of training with *ICLoss* on prediction for 30 min (test set). By rows: ground truth, baseline prediction, baseline with *ICLoss* prediction and "probability" of severe rainfall.

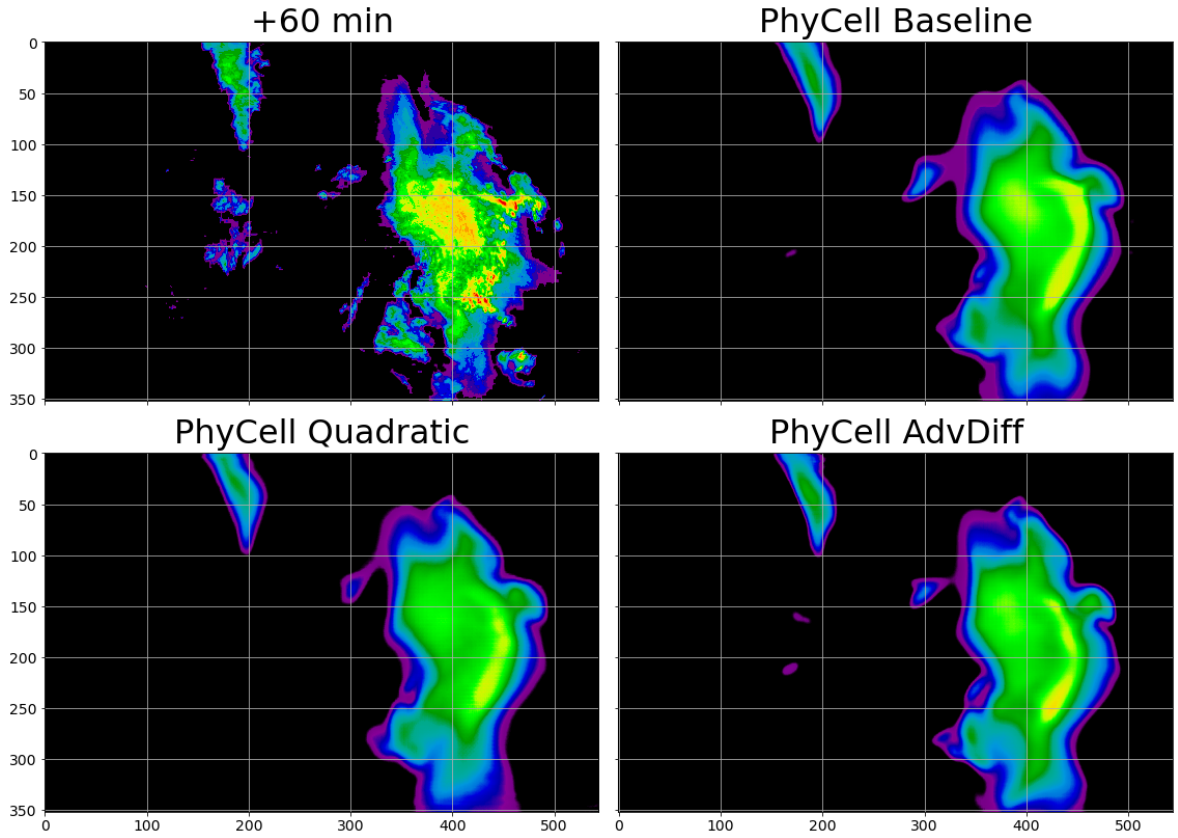


Figure 6.4: Sample prediction by pure PhyCell with different designs of Φ for 60 min (validation set). The top left image is ground truth.

6.2 Non-linearity in the PhyCell

The predictions possibly learned by PhyCell with various designs of the prediction step Φ (Equation 5.3), are studied using PhyDNet models without the deep ConvLSTM residual branch. Firstly, the designs differ by the number of terms in Φ.

- `PhyCell Baseline` computes linear combination of 49 spatial partial derivatives $\mathcal{D}_{i,j}(\mathbf{h}_p^{(t)})$ for $i, j < 7$.
- `PhyCell Quadratic` combines 9 of the first-degree terms $\mathcal{D}_{i,j}(\mathbf{h}_p^{(t)})$ for $i, j < 3$, with all 45 of their possible second-degree combinations for a total of 54 terms.
- `PhyCell AdvDiff` utilizes only differential operators $(\mathcal{D}_{0,1}, \mathcal{D}_{1,0}, \mathcal{D}_{0,2}, \mathcal{D}_{2,0})$, having 4 terms in Φ, out of which two are non-linear.

Consequently, `PhyCell AdvDiff` has significantly less capacity to encode precipitation dynamics than the other two. The utilization of these terms visu-

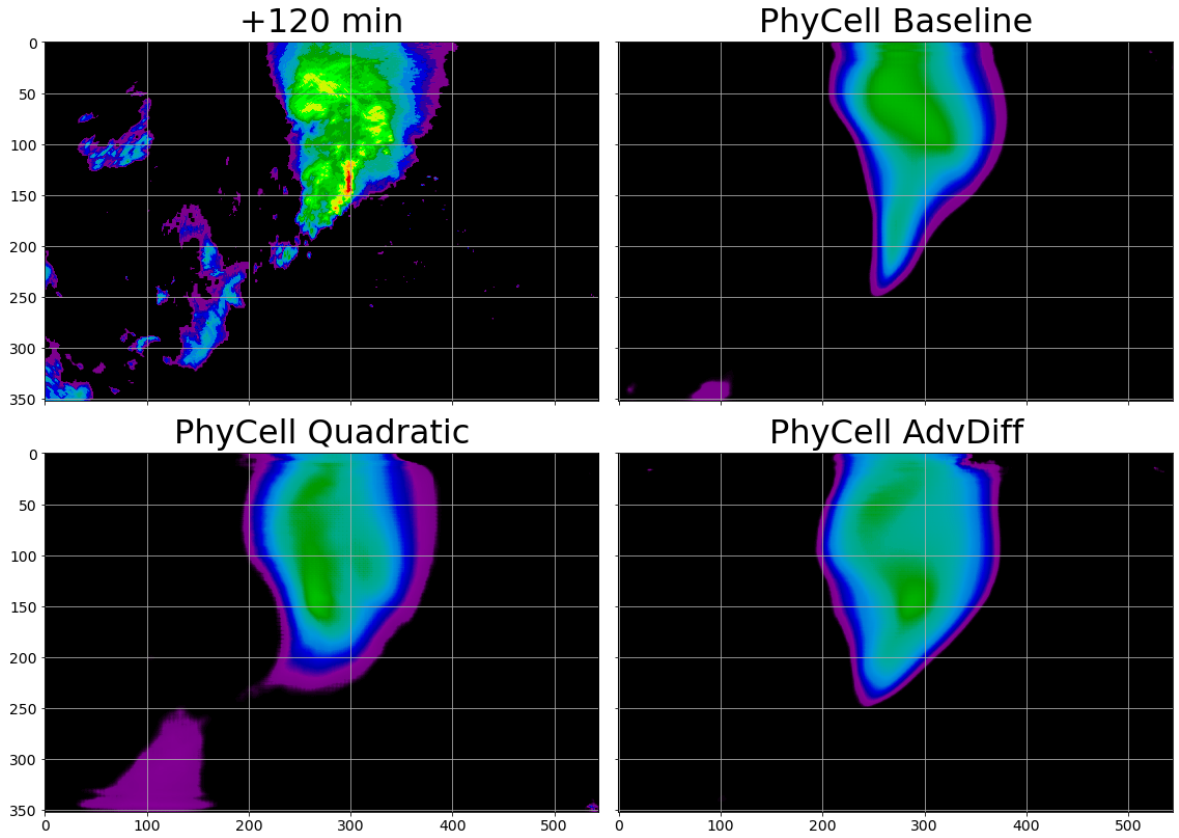


Figure 6.5: Sample prediction by pure PhyCell with different designs of Φ for 120 min (test set). The top left image is ground truth.

alized through c coefficients of Φ in Figure 6.6 shows that `PhyCell Quadratic` prioritizes some terms more than other, when compared to `PhyCell Baseline`. The Top 10 utilized terms by `PhyCell Quadratic` expressed in terms of used differential operators are

$$(\mathcal{D}_{0,0}, \mathcal{D}_{1,0}, \mathcal{D}_{0,1}, \mathcal{D}_{1,2}, \mathcal{D}_{2,0}, \mathcal{D}_{0,2}, \mathcal{D}_{1,1}, (\mathcal{D}_{0,0} \cdot \mathcal{D}_{1,0}), (\mathcal{D}_{0,0} \cdot \mathcal{D}_{1,1}), (\mathcal{D}_{0,0} \cdot \mathcal{D}_{1,2}))\,.$$

The fact that these are either of first degree or multiplied with an undifferentiated hidden state ($\mathcal{D}_{0,0}$) hints that this implementation of non-linearity in Φ is not effective. Moreover, both sample predictions (Figure 6.4) and quantitative evaluations (Figure 6.7) do not show any interesting results.

As summarized in Table 6.2, the proposed designs of PhyCell have worse quantitative performance when compared to the baseline. However, the aim of PhyCell is to give physically sound predictions on which the residual ConvLSTM module can build (Section 3.1) rather than to achieve the best possible quantitative performance alone. Thus, the results of `PhyCell AdvDiff`, given its much smaller capacity of Φ, indicate that it learns precipitation dynamics much more effectively in terms of model size. It may be seen from the

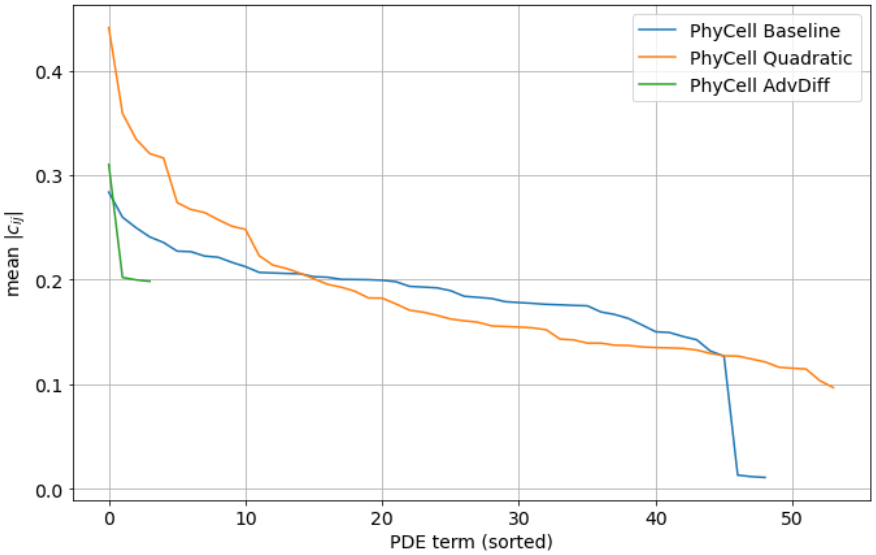


Figure 6.6: Mean absolute value of $c_{i,j}$ linear combination coefficients of PhyCell predictions step.

Table 6.2: Relative change in the metrics when compared to PhyCell Baseline (red denotes performance loss).

	PhyCell Quadratic	**PhyCell AdvDiff**
CSI 8 dBZ	−3.72 %	−1.81 %
CSI 40 dBZ	−9.41 %	+2.70 %
MAE	+2.34 %	+1.00 %
MSE	+4.29 %	+2.46 %
Kolmogorov-Smirnov	+16.26 %	+9.73 %
SSIM	−0.70 %	+0.12 %

sample predictions that `PhyCell Baseline` tends to smooth out the outputs to optimize for the loss, which is not the case for `PhyCell AdvDiff` (see the borders of the predicted precipitation in Figure 6.4). The sample in Figure 6.5, containing predictions for twice the length of the training horizon, shows that `PhyCell AdvDiff` is the only model to correctly predict the position of the high-intensity precipitation this far into the future, even though without the correct intensities. This hypothesis of less smoothed predictions that are better at predicting the location of the phenomena is supported by the gain in the high-threshold CSI alongside decay in the low-threshold one.

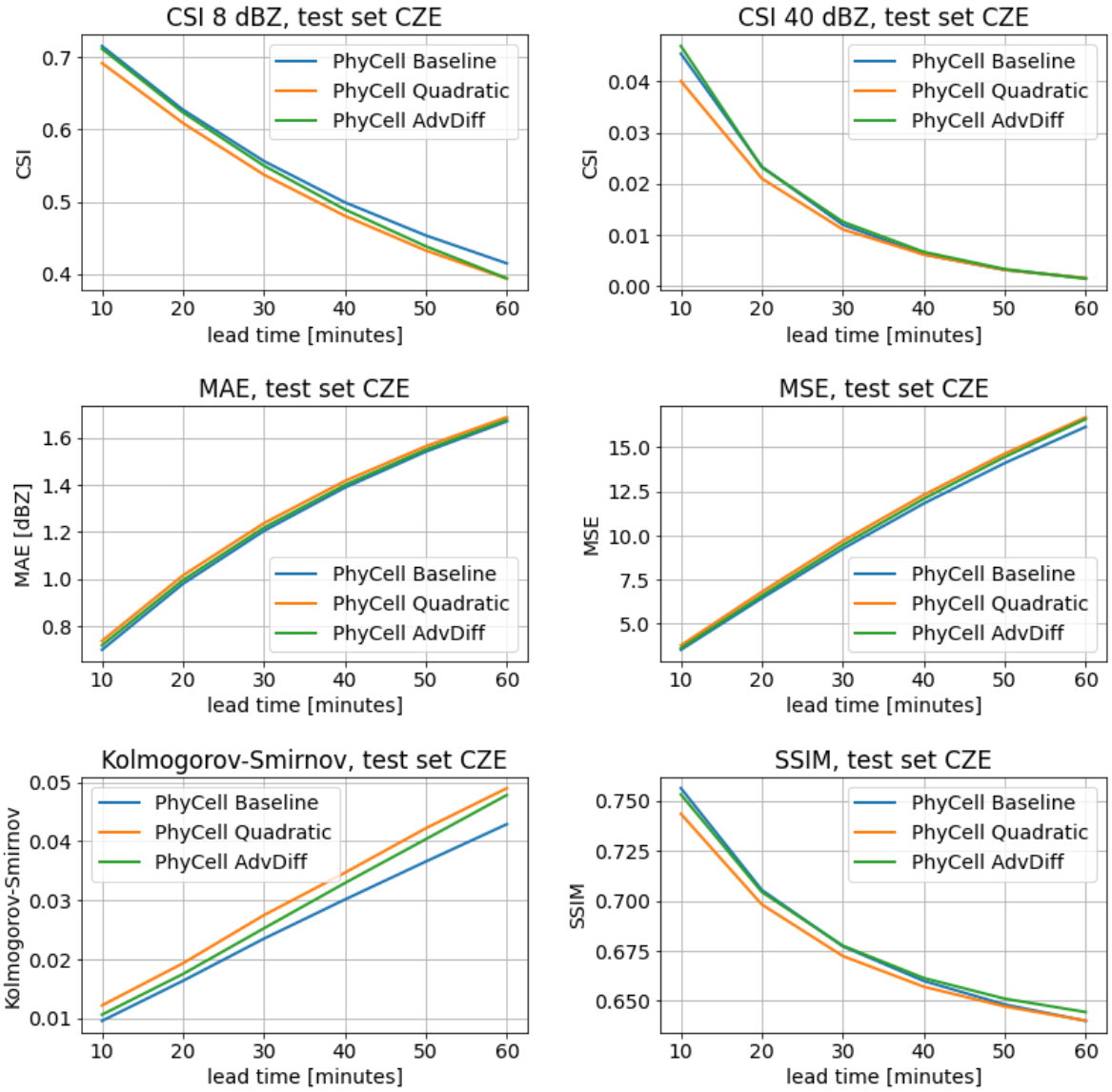


Figure 6.7: Quantitative performance of pure PhyCell with different designs of Φ on the test set for 60 min predictions.

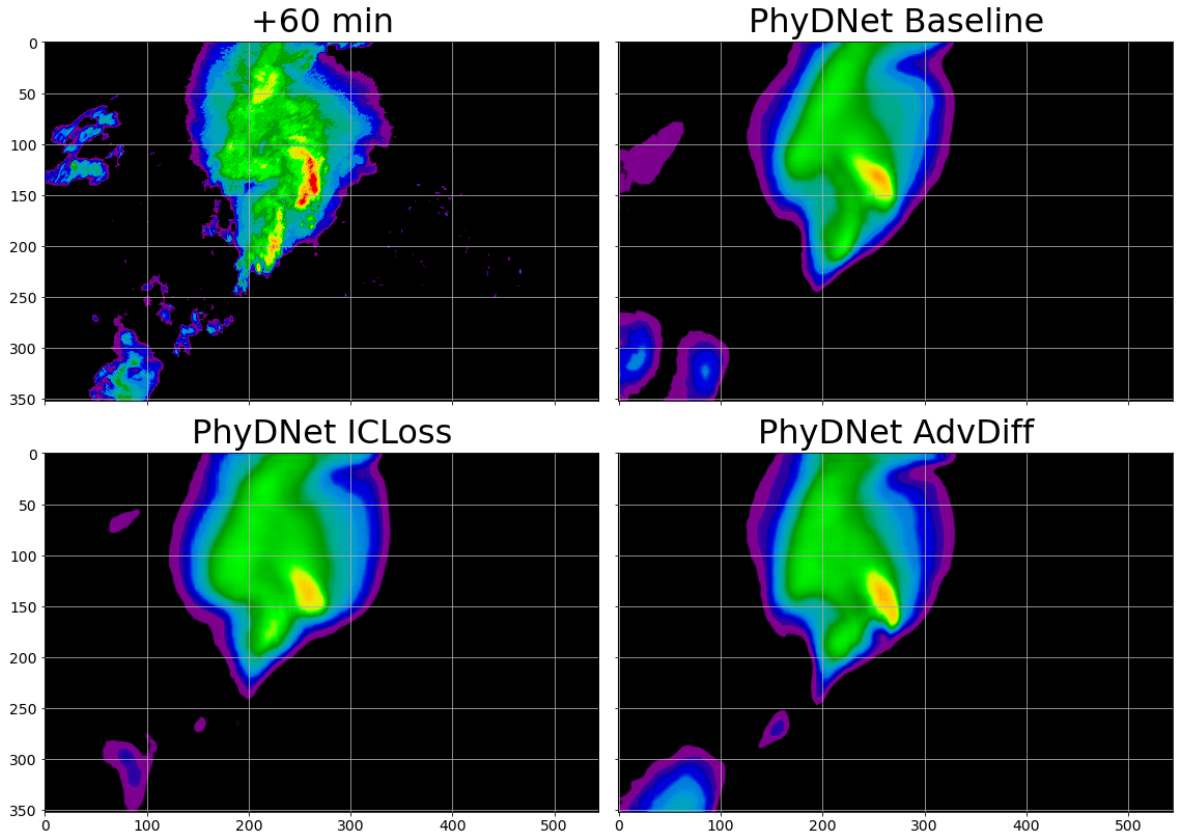


Figure 6.8: Sample prediction of convective precipitation by PhyDNet versions for 60 min (test set). The top left image is ground truth.

6.3 Evaluation of PhyDNet AdvDiff

In this section, `PhyDNet AdvDiff` is compared to the `PhyDNet Baseline` to evaluate the overall effect of the proposed changes on the prediction performance. `PhyDNet ICLoss` model is included, to distinguish between the changes introduced by *ICLoss* and *advection-diffusion* equation in the PhyCell. As may be seen in Figure 6.10 and is summarized in Table 6.3, all three models achieve very similar mean errors, and their specifics are projected into trade-offs in other metrics. However, unlike in the previous section, sample predictions on the test set subjectively do not show any features that would clearly differentiate them (an example prediction of convective precipitation in Figure 6.8 and of stratiform precipitation in Figure 6.9).

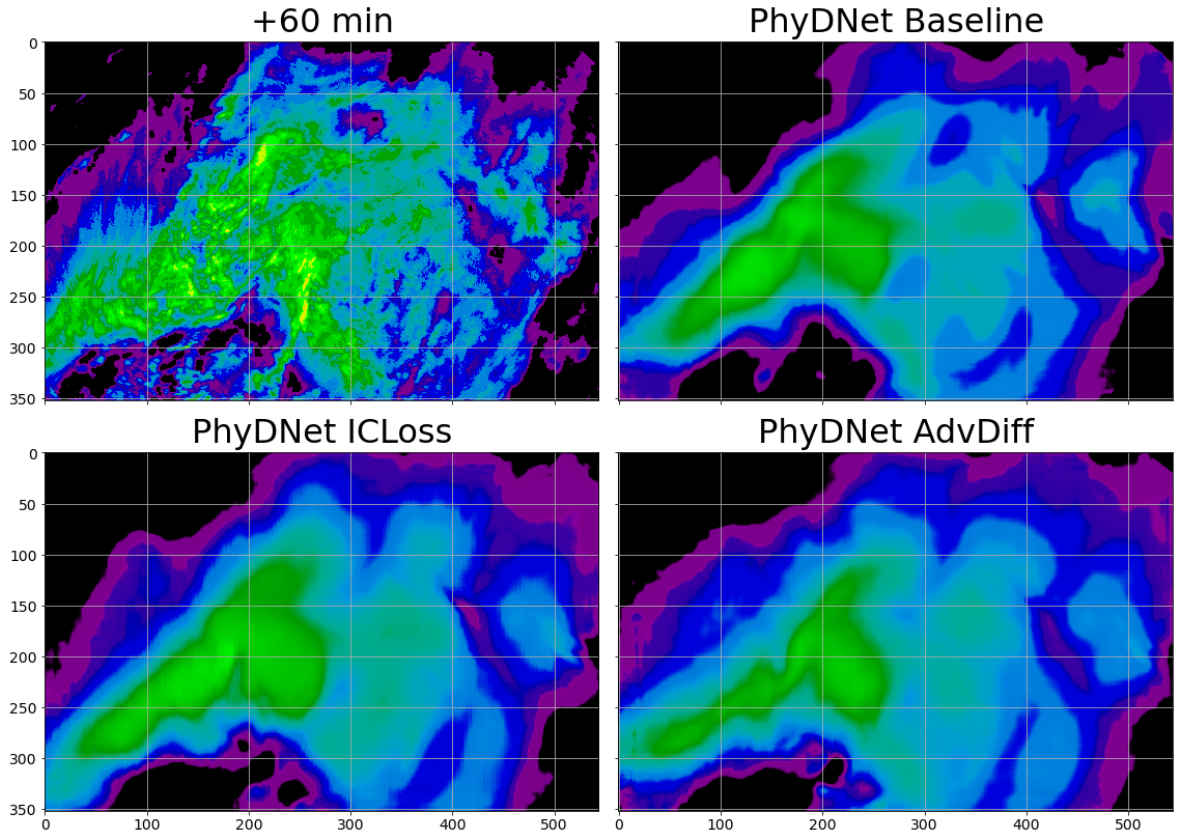


Figure 6.9: Sample prediction of stratiform precipitation by PhyDNet versions for 60 min (test set). The top left image is ground truth.

Table 6.3: Relative change in the metrics of PhyDNet AdvDiff and PhyDNet ICLoss compared to PhyDNet Baseline (red denotes performance loss).

	PhyDNet ICLoss	**PhyDNet AdvDiff**
CSI 8 dBZ	−2.30 %	−3.41 %
CSI 40 dBZ	+6.46 %	−6.14 %
MAE	+0.25 %	+0.14 %
MSE	−0.38 %	−0.40 %
Kolmogorov-Smirnov	−2.13 %	+9.82 %
SSIM	−0.22 %	−0.11 %

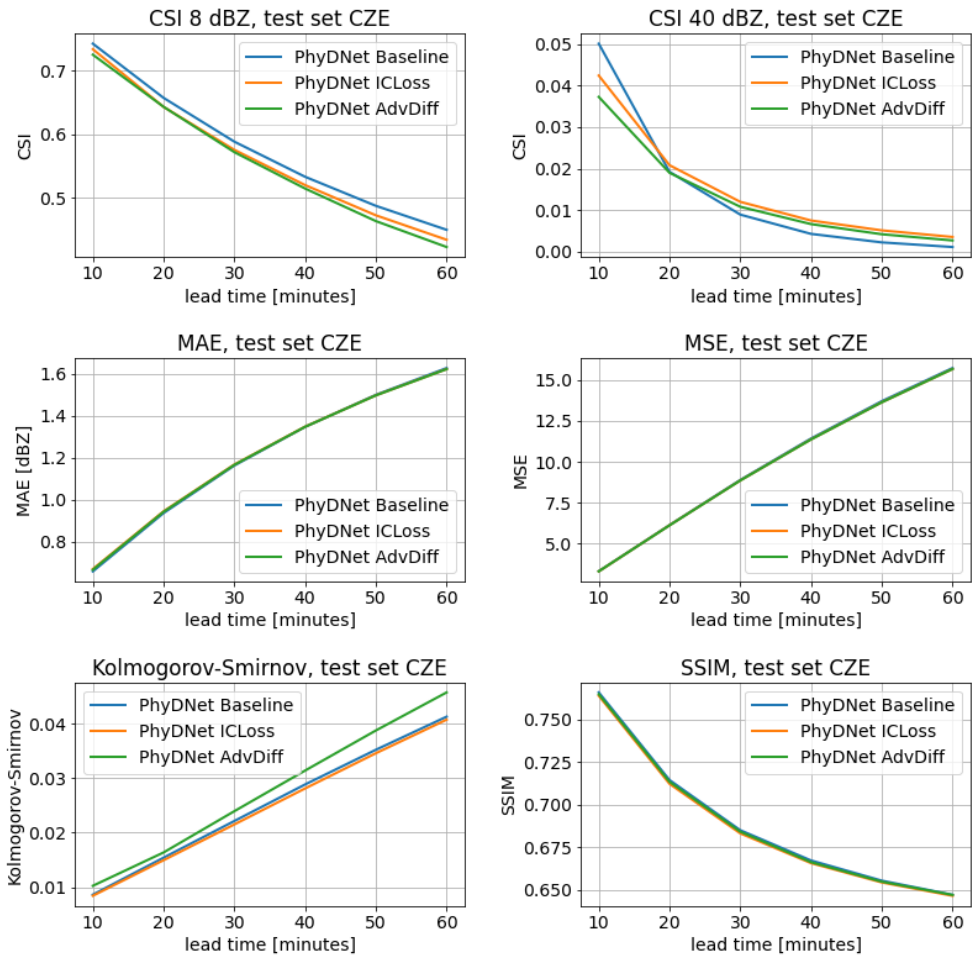


Figure 6.10: Quantitative performance of the proposed PhyDNet AdvDiff on the test set for 60 min predictions.

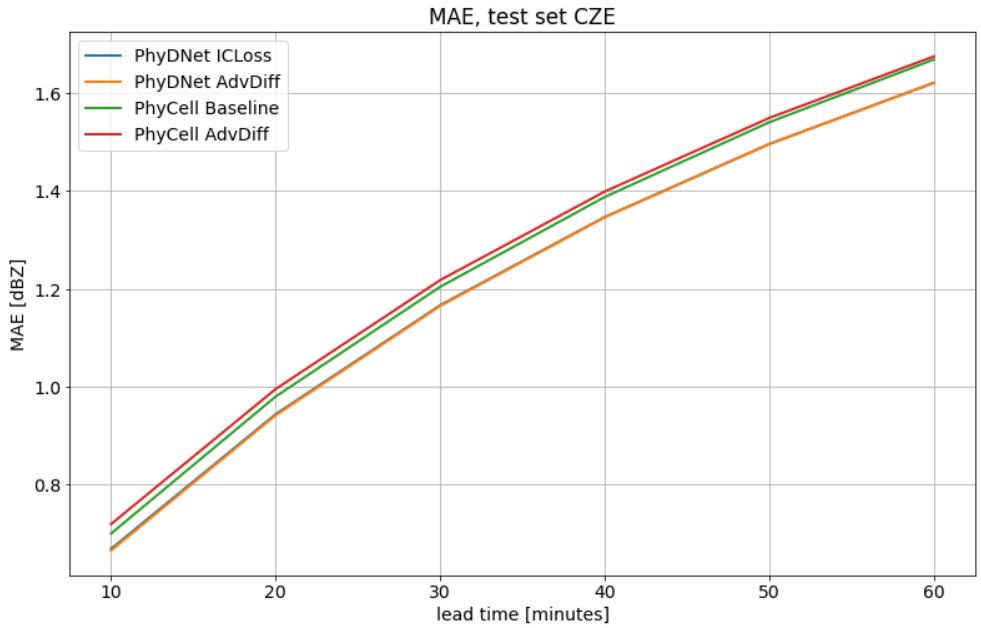


Figure 6.11: MAE on the test set.

A difference among the predictions may be observed if predictions are decomposed into physical and residual branches, which are separately reconstructed through decoder D and visualized (Figure 6.12). ConvLSTM of the `PhyDNet AdvDiff` learns predictions containing a variety of objects and intensities. In contrast, the ConvLSTM of `PhyDNet Baseline` predicts only objects with high intensities and the predictions of `PhyDNet ICLoss` ConvLSTM lack structure altogether. Thus subjectively, `PhyDNet AdvDiff` utilizes the residual part the most. To partially quantify this hypothesis, Figure 6.11 presents values of MAE for PhyCell and PhyDNet in one plot. While there is a difference in PhyCell errors, there is almost none in the case of PhyDNet – in different models, ConvLSTM contributed different amounts to the overall performance.

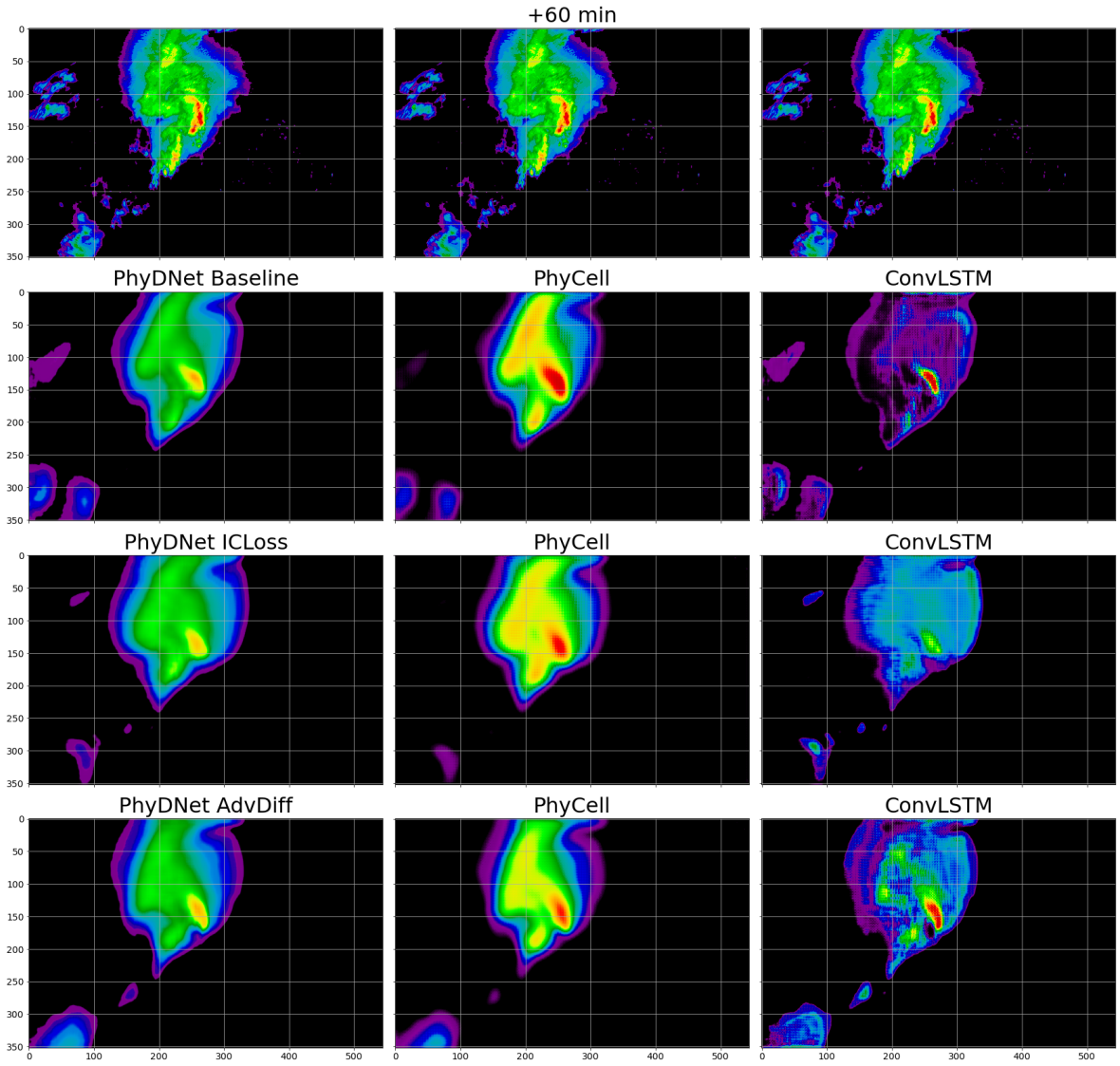


Figure 6.12: Decomposition of PhyDNet branches on a prediction for 60 min (test set). The top row displays ground truth three times.

6.4 Advection Field Inferred by PhyCell AdvDiff

This section contains visualizations (Figure 6.13) of the advection field $\mathbf{u}^{(t)}$, inferred from the observed data through their use in *advection-diffusion* equation in PhyCell (Subsection 5.2.2). $\mathbf{u}^{(t)}$ is computed from a single hidden state $\mathbf{h}_p^{(t)}$, which should theoretically contain complete information about the current precipitation situation. In the case of `PhyCell AdvDiff`, it may be observed that direction of the vectors matters. However, instead of the general motion vectors (interpretable as wind), these rather resemble the direction of temporarily and spatially local development.

On the other hand, the PhyCell of `PhyDNet AdvDiff` seems to ignore the direction of $\mathbf{u}^{(t)}$ and uses it just to multiply the $\mathbf{h}_p^{(t)}$ with the observed precipitation intensities. In this case, $\mathbf{u}^{(t)}$ points uniformly South-East, independently of the actual movement directions. Nevertheless, it has to be pointed out that the eastward direction of precipitation movement prevails in the considered Central-European geographical location.

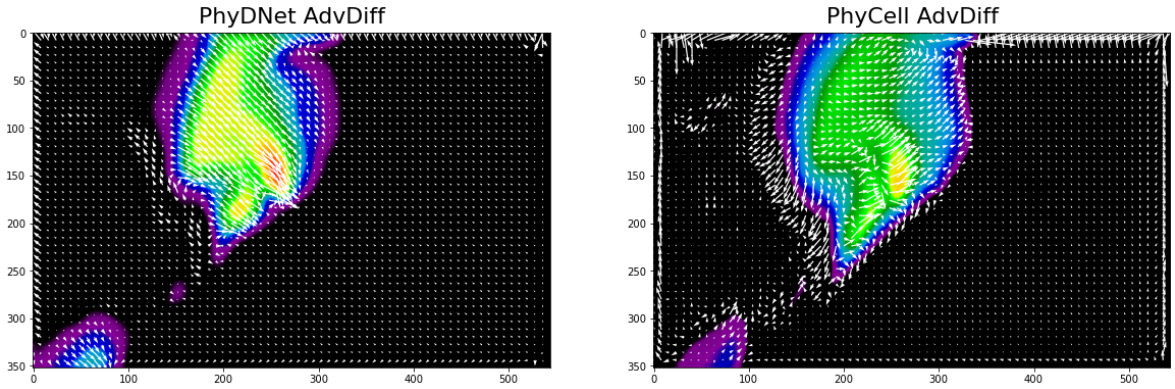
6.5 Summary of PhyCell Experiments

There is a difference in the learned dynamics of *advection-diffusion* PhyCell when trained alone and as a part of PhyDNet, alongside the different amount of contribution to the prediction by ConvLSTM (Figure 6.11). These results lead us to speculate that only a limited amount of dynamics governing the precipitation are learnable under the current problem setting. Results in Section 6.2 indicate that PhyCell can utilize the provided physics prior to predicting precipitation. However, combining PhyCell with a high-capacity ConvLSTM and training it for regression using a mean error loss function neglects the physics learned by the regularized PhyCell, in favor of smoothed predictions with unchanged performance. Moreover, the varying contribution of ConvLSTM hints that it can possibly learn more complex dynamics, given a change in the training formulation. This observation points to the loss function selection as the main limitation of the current approach.

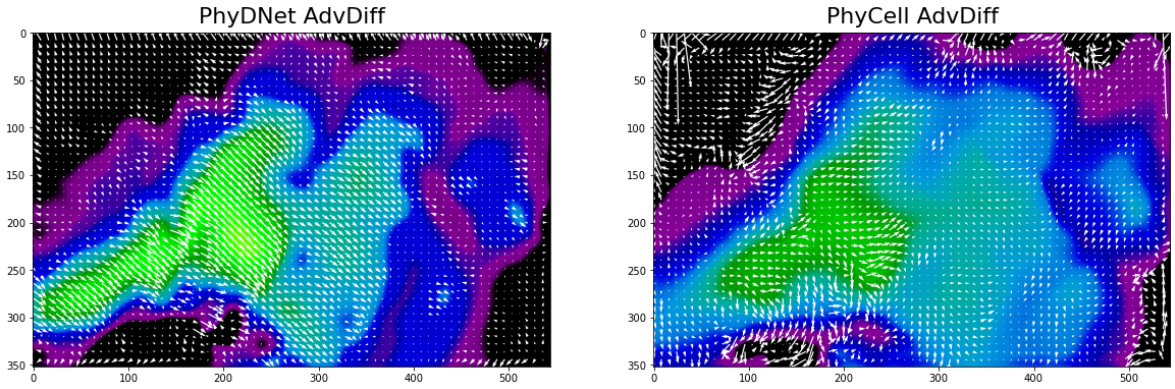
However, there are caveats to this theory that need to be addressed. Firstly, trying to infer a global advection field as well, we experimented with $\mathbf{u}^{(t)}$ computation from multiple previous states and with enforcing more physics through a continuity equation

$$\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} = 0, \tag{6.1}$$

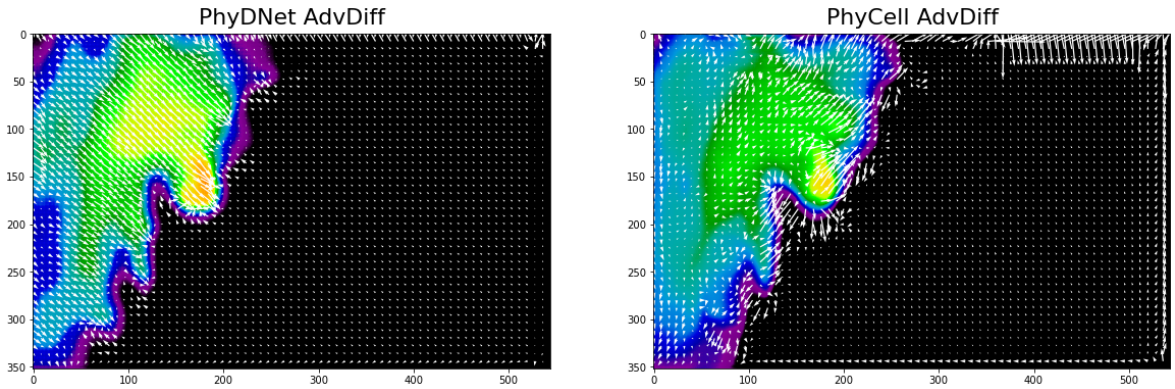
both resulting in divergence during the training. Two possible reasons are that the convolutional module U (Equation 5.9) does not have enough capacity to



(a) convective precipitation moving East-North-East



(b) stratiform precipitation moving East



(c) convective precipitation moving North-North-East

Figure 6.13: Advection field **u** plotted over a PhyCell partial prediction for 60 min (test set).

capture this complicated vector field or that there is not enough input data for its inference (e.g., it rains only in some parts of the domain). Secondly, the training of disentanglement in PhyDNet was not studied sufficiently. It may be possible that pre-training of some modules, introducing non-linear disentanglement, or emphasizing predictions of the PhyCell over the ones of ConvLSTM could lead to different results.

Conclusion

Short-term high-resolution precipitation forecasts, called precipitation nowcasting, can increase everyday comfort and safety while providing information beneficial for protecting human lives and property during storms. In this thesis, we were working on addressing the limitations of DL models for precipitation nowcasting, such as smoothing out the important high-frequency features and low explainability of prediction dynamics.

The thesis is divided into six chapters. Chapter 1 started the thesis by introducing the reader to a theory about precipitation and describing approaches to weather measurements and forecasting. A summarization of important differential equations and deep learning concepts follows. Related work relevant to this thesis is presented in Chapter 2.

A detailed description of PhyDNet is provided in Chapter 3. The concept of a convolutional RNN that disentangles physical and unknown dynamics serves as a building stone for the practical part of this thesis. In Chapter 4 we formally define the tackled precipitation nowcasting problem and the DL approach to it. A description of the radar echo sequence dataset on which the presented models are trained closes this chapter.

We present researched changes to PhyDNet and explain our motivation for them in Chapter 5. To shift the training of the model toward high-intensity precipitation, we explore the use of the presented *ICLoss*. By engineering *advection-diffusion* PDE into PhyCell, we aimed to introduce possibly missing dynamics to the model and shed some light on the dynamics learned. This chapter is ended with a list of details of all the models trained.

Results of the experiments and their analysis are presented in Chapter 6. We found out that training with *ICLoss* results in better prediction of precipi-

tation over the selected threshold but, at the same time, predicts no rainfall more intense than the threshold as well. Thus, this approach improves performance during severe rainfall events but remains ignorant of the important high-intensity features.

The introduction of *advection-diffusion* equation to PhyCell, resulted in a regularized model with predictions more resembling actual dynamics in the atmosphere. However, changes to PhyCell have not improved either the overall quantitative performance of PhyDNet or the problem with smoothed-out predictions. This result indicates that even if a part of PhyDNet is regularized with a physics prior and possibly learns the corresponding dynamics more effectively, the final predictions remain decided by the optimization of the loss function during training.

The goals of this thesis were achieved. Contributing to bridging the gap between physics-based and DL weather forecasting, we have explored human knowledge of precipitation and how it may be used in a physics-constrained DL model. We have presented ideas possibly addressing the identified problems of DL nowcasting models. All of the trained models were tested, and the results were analyzed.

Limitations and Outline of Future Work

Our implementation of the presented ideas has its limitations. The convolutional modules added to the network to compute a new output for the *ICLoss* and velocity vectors used in the *advection-diffusion* equation are possibly too small to utilize these designs to their full potential. We have not experimented sufficiently with the inclusion of *advection-diffusion* PhyCell into the whole PhyDNet. These need to be addressed to verify the presented results.

We suggest that the future work is focused on the following steps.

- To better leverage the physics prior, its inclusion not only through regularization but in the loss term should be explored. Alternatively, this could be improved by controlling the contribution ratio between PhyDNet branches.
- We are aware that equations governing the development and dynamics of precipitation are more complex than the advection-diffusion. More precise equations could possibly be inferred from the data.
- The prediction of high-intensity features could be achieved by training under a Generative Adversarial Network framework.
- Modeling more complex dynamics in the PhyCell, and enlarging the capacity of the residual network (ConvLSTM) could improve the quantitative performance.

Bibliography

[1] Donald Ahrens, C. *Essentials of Meteorology: An Invitation to the Atmosphere*. Brooks/Cole/Thomson Learning, 2001, ISBN 9780534372002.

[2] Bauer, P.; Thorpe, A.; et al. The quiet revolution of numerical weather prediction. *Nature*, volume 525, no. 7567, Sept. 2015: pp. 47–55.

[3] Zhang, A.; Lipton, Z. C.; et al. Dive into Deep Learning. June 2021, arXiv:2106.11342.

[4] Le Guen, V.; Thome, N. Disentangling Physical Dynamics from Unknown Factors for Unsupervised Video Prediction. Mar. 2020, arXiv:2003.01460.

[5] OpenStreetMap. https://www.openstreetmap.org/copyright, accessed: 2022-4-16.

[6] Peštová, Z. Na Hodonínsku se vyskytlo tornádo. https://www.meteopress.cz/report/na-hodoninsku-se-dnes-vyskytlo-tornado/, June 2021, accessed: 2022-5-2.

[7] Choma, M. *Interpolation and Extrapolation of Subsequent Weather Radar Images*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

[8] Choma, M. MWNet v1.0 — AI precipitation nowcasting for the public. https://medium.com/pocasi/mwnet-v1-0-ai-precipitation-nowcasting-for-the-public-a9192b6dc652, Sept. 2021, accessed: 2022-5-1.

[9] Kalnay, E. *Atmospheric Modeling, Data Assimilation and Predictability*. Cambridge University Press, 2003, ISBN 9780521791793.

[10] Auger, L.; Dupont, O.; et al. AROME–NWC: a new nowcasting tool based on an operational mesoscale forecasting system. *Quart. J. Roy. Meteor. Soc.*, volume 141, no. 690, July 2015: pp. 1603–1611.

[11] World Meteorological Organization (WMO). *Guidelines for Nowcasting Techniques*. WMO, 2017, ISBN 9789263111982.

[12] Trench, W. F. *Elementary Differential Equations with Boundary Value Problems*. Brooks/Cole-Thomson Learning, 2001, ISBN 9780534263287.

[13] Strang, G. Applied Mathematics and Scientific Computing. https://math.mit.edu/classes/18.086/2006/am54.pdf, 2007.

[14] Thuerey, N.; Holl, P.; et al. Physics-based Deep Learning. Sept. 2021, arXiv:2109.05237.

[15] Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.*, volume 9, no. 8, Nov. 1997: pp. 1735–1780.

[16] Shi, X.; Chen, Z.; et al. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. June 2015, arXiv:1506.04214.

[17] Wu, Y.; He, K. Group Normalization. Mar. 2018, arXiv:1803.08494.

[18] Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Feb. 2015, arXiv:1502.03167.

[19] Prudden, R.; Adams, S.; et al. A review of radar-based nowcasting of precipitation and applicable machine learning techniques. May 2020, arXiv:2005.04988.

[20] Ayzel, G.; Heistermann, M.; et al. Optical flow models as an open benchmark for radar-based precipitation nowcasting (rainymotion v0.1). *Geosci. Model Dev.*, volume 12, no. 4, Apr. 2019: pp. 1387–1402.

[21] Pulkkinen, S.; Nerini, D.; et al. Pysteps: an open-source Python library for probabilistic precipitation nowcasting (v1.0). *Geosci. Model Dev.*, volume 12, no. 10, Oct. 2019: pp. 4185–4219.

[22] Wang, Y.; Wu, H.; et al. PredRNN: A Recurrent Neural Network for Spatiotemporal Predictive Learning. Mar. 2021, arXiv:2103.09504.

[23] Ravuri, S.; Lenc, K.; et al. Skillful Precipitation Nowcasting using Deep Generative Models of Radar. Apr. 2021, arXiv:2104.00954.

[24] Raissi, M.; Perdikaris, P.; et al. Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. Nov. 2017, arXiv:1711.10561.

[25] Raissi, M.; Yazdani, A.; et al. Hidden Fluid Mechanics: A Navier-Stokes Informed Deep Learning Framework for Assimilating Flow Visualization Data. Aug. 2018, arXiv:1808.04327.

[26] Le Guen, V.; Yin, Y.; et al. Augmenting Physical Models with Deep Networks for Complex Dynamics Forecasting. Oct. 2020, arXiv:2010.04456.

[27] Kalman, R. E. A new approach to linear filtering and prediction problems. *J. Basic Eng.*, volume 82, no. 1, Mar. 1960: pp. 35–45.

[28] Eumetnet. OPERA. https://www.eumetnet.eu/activities/observations-programme/current-activities/opera/, Aug. 2016, accessed: 2022-5-4.

[29] EPSG Geodetic Parameter Dataset. https://epsg.org/home.html, accessed: 2022-4-16.

[30] Paszke, A.; Gross, S.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, edited by H. Wallach; H. Larochelle; A. Beygelzimer; F. d' Alché-Buc; E. Fox; R. Garnett, Curran Associates, Inc., 2019, pp. 8024–8035.

Appendix A

Acronyms

CNN Convolutional Neural Network

DALR Dry Adiabatic Lapse Rate

DL Deep Learning

GPU Graphics Processing Unit

GN Group Normalization

LSTM Long Short-Term Memory

MAE Mean Absolute Error

ML Machine Learning

MSE Mean Squared Error

NN Neural Network

NWP Numerical Weather Prediction

ODE Ordinary Differential Equation

PDE Partial Differential Equation

RAM Random Access Memory

RNN Recurrent Neural Network

SALR Saturated Adiabatic Lapse Rate

Appendix B

Contents of enclosed CD

- `README.md` ..... the file with CD contents description
- `example.ipynb` ..... Jupyter notebook with examples
- `train.py` ..... training script
- `configs` ..... the directory with training configurations
- `core` ..... the directory containing training source files
  - `data_loader` ..... the directory containing data loading logic
  - `model` ..... the directory containing model implementations
    - `layers` ..... the directory with PhyDNet modules
      - `phydnet_constrain_moments.py` ..... moments of convolutional kernels
      - `phydnet_layers.py` ..... default ConvLSTM and PhyCell
      - `phydnet_phycells.py` ..... custom PhyCell
    - `loss.py` ..... loss function implementation
    - `metric_detail.py` ..... metric implementations
    - `phydnet.py` ..... PhyDNet implementation
  - `utils` ..... the directory containing utility functions
  - `env.yml` ..... conda environment specification
- `data` ..... the directory with sample radar echo data
- `text` ..... the thesis text directory
  - `src` ..... the directory with thesis LaTeX source files
  - `thesis.pdf` ..... the thesis text in PDF format

Appendix C

Sample Predictions of the Trained Models

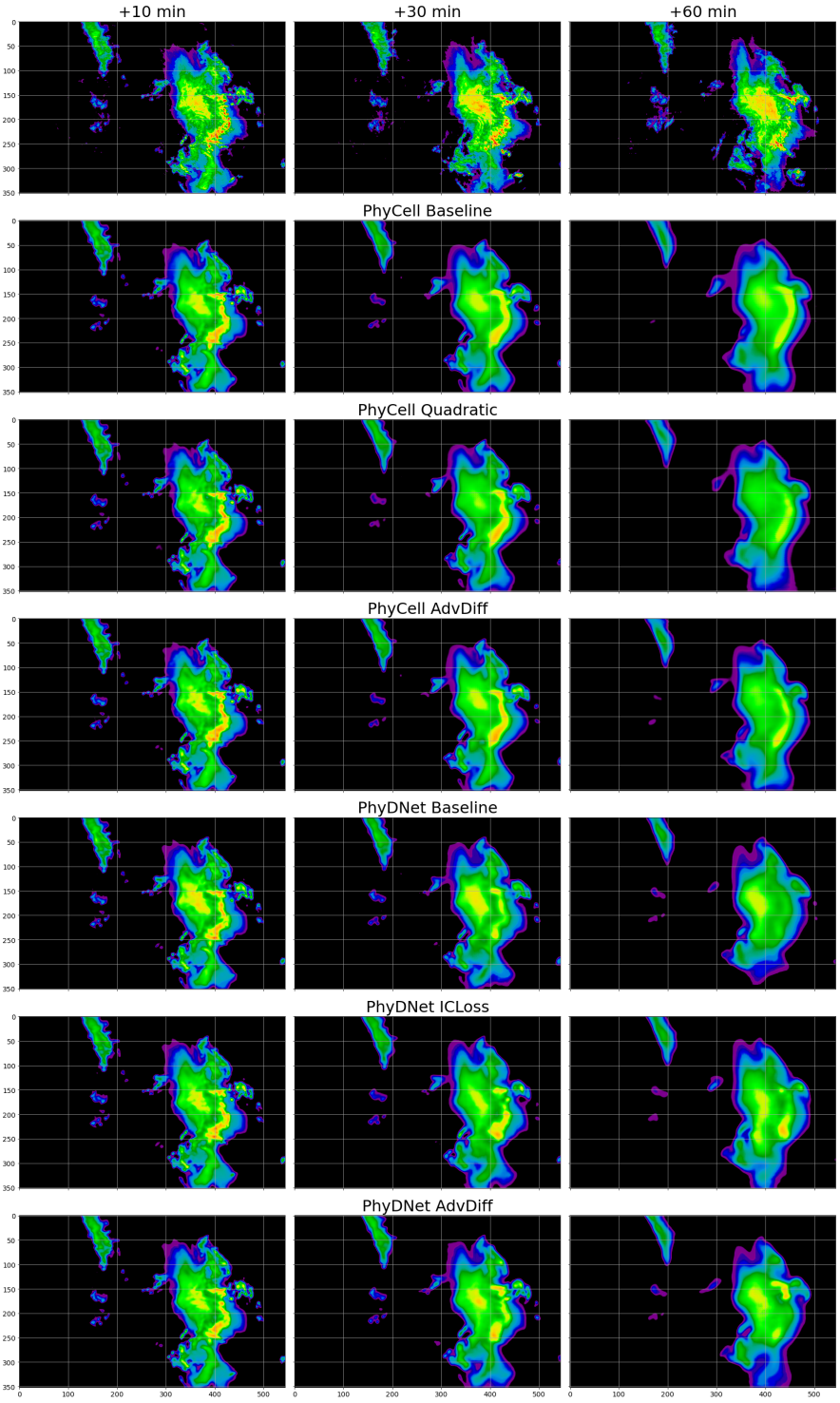


Figure C.1: Sample prediction by every model for three lead times. The top row displays ground truth. Convective precipitation, validation set.

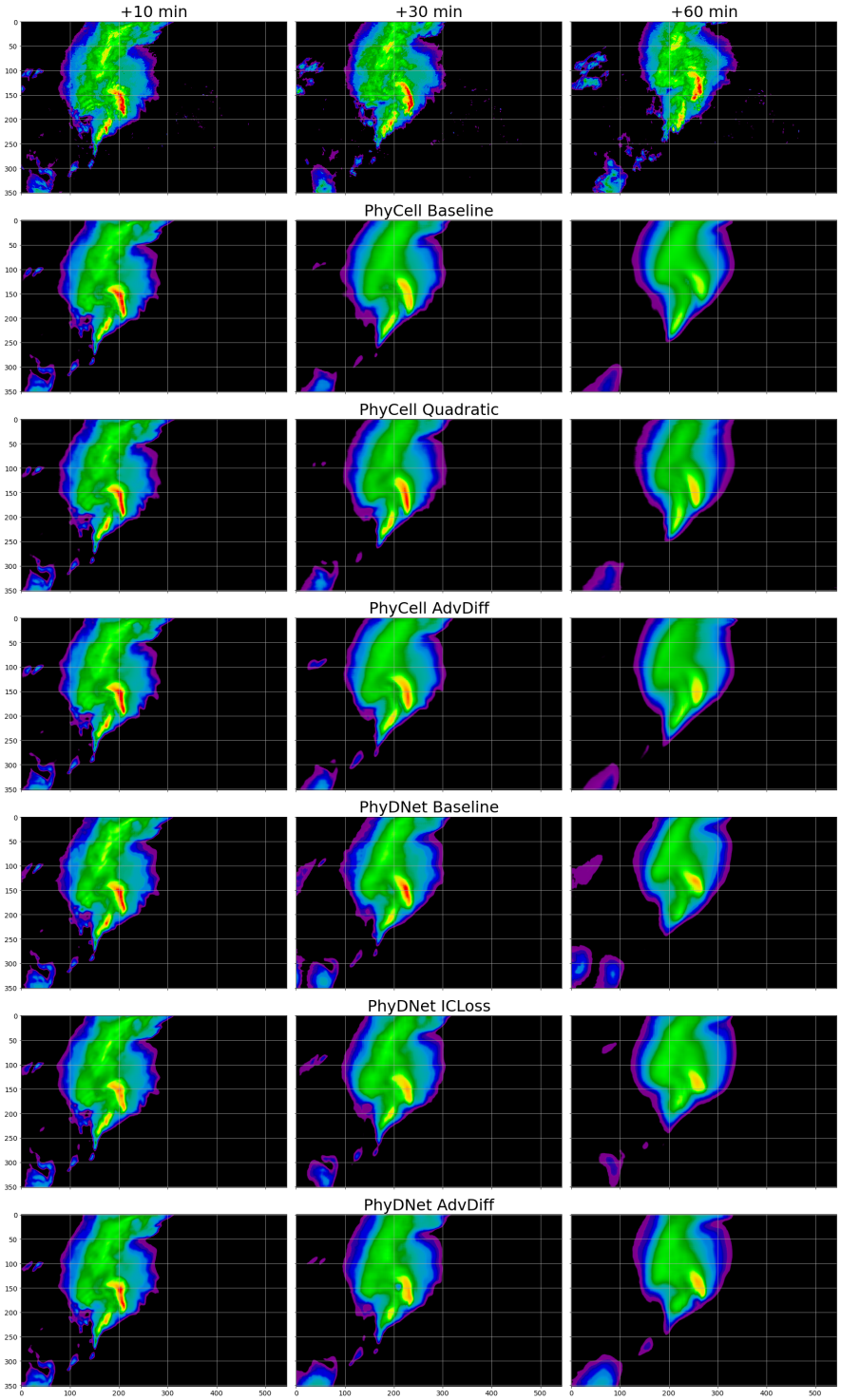


Figure C.2: Sample prediction by every model for three lead times. The top row displays ground truth. Convective precipitation, test set.

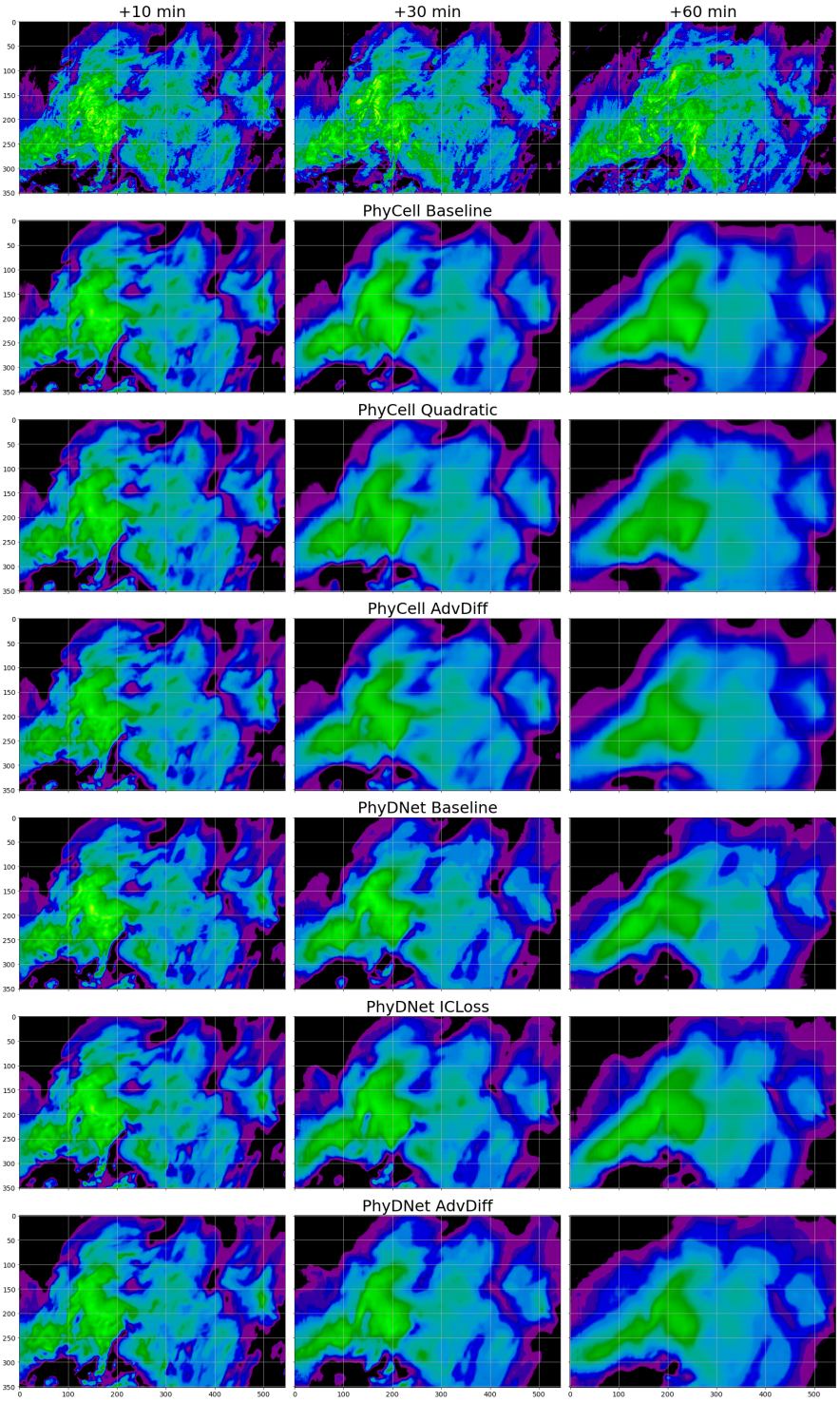


Figure C.3: Sample prediction by every model for three lead times. The top row displays ground truth. Stratiform precipitation, test set.