Satellite to Radar: Sequence to Sequence Learning for precipitation nowcasting

MARK BRUDERER, University of Twente, The Netherlands

Fig. 1. A supercell thunderstorm at twilight in SW Oklahoma.[1]

ABSTRACT

The forecasting of rain is a complex problem with centuries of scientific work. The implications of weather for individuals and companies continue to be important. Machine Learning approaches have been shown to outperform state of the art physics based models of weather for short term predictions. Using multi-spectral satellite images as out input and radar reflectivity as the target. We investigate three different types of models: 3D U-Net, ConvLSTM and ConvLSTM with self attention. We found that ConvLSTM outperforms the other approaches for both classification and regression pixel rain intensities.

Additional Key Words and Phrases: Machine Learning, Sequence to Sequence, Radar, Satellite, Storms, Forecasting, 3D U-Net, ConvLSTM, Attention Mechanism

1 INTRODUCTION

Precipitation forecasting is essential to reduce the risk of life threatening situations. Different types of rainfall ranging from mist to heavy rain have a major impact for different societal sectors including agriculture, aviation, outdoor events, and the energy industry. By having timely and accurate predictions of rainfall which in turn indicate the potential for destructive storms we can prevent injuries, assist companies in predicting energy production and use resources efficiently.

[1]Photograph by Raychel Sanner, Unsplash Licence

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

TScIT 39, July 7, 2023, Enschede, The Netherlands
© 2023 Association for Computing Machinery.

At present meteorologists are able to successfully predict many instances of precipitation. Techniques that are used in practice range from manual analysis of current weather data (e.g radar or satellite images) to complex physics based simulations of our atmosphere with Numerical Weather Prediction (NWP) models. Various short term forecasting methods are based on *optical flow*. Optical flow functions in two steps, first cumulonimbus (storm) clouds are identified, and then their movement is tracked to predict the location of precipitation. Thus in this case, the forming and dissipation of clouds known as a *cell-lifecycle* [29] is not taken into account [26].

Machine Learning (ML) approaches have also been developed to predict precipitation. An improvement of machine learning models over NWP models is that they are much faster to produce predictions, thus ML models are more suitable for real-time or near-real-time predictions, such as required in disaster response and energy management. These short term predictions are referred to as *nowcasts*. According to the universal approximation theorem [11], deep neural networks have the property of being able to approximate any function provided they have the correct weights, thus it is suggested that machine learning models can incorporate sources of predictability beyond optical flow such as the *cell-lifecycle*. Other suggested sources of predictability are: the elevation of terrain, convergence lines and the current time among others [26].

Thus far most machine learning approaches for precipitation nowcasting have focused on predicting future frames of currently available radar data [7, 30, 31]. However taking this approach may eliminate the possibility of learning the *cell-lifecycle*, due to the fact that the model only sees precipitation itself but not the cloud that is causing the precipitation.

We propose to use multi-spectral satellite data to learn spatio-temporal mappings between sequences of satellite data and precipitation data in the near future. A well performing model could predict storm clouds when these clouds are still forming. An additional advantage is that contrary to radar data, satellite data is readily available over oceans and remote communities (See figure 8) which allows for the prediction of precipitation over these regions.

1.1 Problem Formulation

We consider precipitation nowcasting as a self-supervised problem. The prediction of labels can be accomplished through two approaches: predicting discrete classes that correspond to different rain intensity intervals, or conducting pixel-level regression to learn the precise values of precipitation.

1.1.1 Regression Formulation. Consider a dataset {X, Y} consisting of pairs of input-output sequences indexed by $i \in \mathbb{N}$, Let

$$X = \{x^{(i)} \in \mathbb{R}^{t\times h\times w\times c}\}\forall i$$

where $x^{(i)}$ is a tensor of dimension $t \times h \times w \times c$ representing the sequence of satellite images at position i, having t time-steps, h height, w width and c channels. The set of output sequences is denoted as

$$Y = \{y^{(i)} \in \mathbb{R}^{w\times h}\}\forall i$$

where $y^{(i)}$ represents the i^{th} $h \times w$ dimensional tensor with each pixel being a real number collected by the radar reflectivity reading. Here the width and height are the same as in the input sequence. The problem is formulated as finding a function f(x). This function must minimize a chosen distance function $\mathcal{D}$ as follows: Let $\hat{Y} = \{p(x^{(i)})\}\forall i$, representing the predicted outputs for each sequence of satellite images and identical in dimensions to $y^{(i)}$. find f(x) such that $\mathcal{D}(\hat{Y}, Y)$ is minimized.

1.1.2 Classification Formulation. Consider a dataset {X, Y} consisting of pairs of input-output sequences indexed by $i \in \mathbb{N}$, Let

$$X = \{x^{(i)} \in \mathbb{R}^{t\times h\times w\times c}\}\forall i$$

where $x^{(i)}$ is a tensor of dimension $t \times h \times w \times c$ representing the sequence of satellite images at position i, having t time-steps, h height, w width and c channels. The set of output sequences is denoted as

$$Y = \{y^{(i)} \in \mathbb{N}^{w\times h}\}\forall i$$

where $y^{(i)}$ represents the i^{th} h $\times w$ dimensional tensor containing discrete integers mapping to rainfall intensity classes. The problem is formulated as finding a probability mass function p(x). This function must minimize the Cross Entropy Loss expressed as follows:

$$E = \frac{1}{h+w}\sum_{i=1}^{h}\sum_{j=1}^{w} t_{ij} log(p_{ij})$$

Let $\hat{Y} = \{p(x^{(i)})\}\forall i$, representing the predicted outputs for each sequence of satellite images and identical in dimensions to $y^{(i)}$. find p(x) such that $E(\hat{Y}, Y)$ is minimized.

1.2 Research Question

RQ: *How can a deep learning model be trained to predict radar data with multi-spectral satellite data ?*

This research question will be answered by looking at the following sub research questions:

(1) *how must data be preprocessed and aggregated to create a model capable of predicting precipitation based on satellite data?*
(2) *How can the process of training be simplified to be able to experiment with different architectures ?*
(3) *what model architecture performs the best based on established metrics for classification and regression ?*

2 CONTRIBUTION

In this research we focus on finding possible approaches to forecast precipitation, given only satellite imagery. We create 3 different models that are then trained for this task. We train each of these models in a classification and regression variant. We then evaluate our models using a wide range of evaluation metrics.

3 RELATED WORKS

In this section we will discuss the existing work in precipitation nowcasting via machine learning. This section is structured by the type of input data used in the approaches, first we list radar based learning and then we discuss satellite based approaches.

3.1 Radar Based Nowcasting

Recurrent neural networks (RNNs) have been created to learn temporal relationships in data, therefore they are a natural candidate to the task of learning spatio-temporal patterns of weather. The LSTM architecture was developed by Hochreiter and Schmidhuber [23], to solve the problem of vanishing and exploding gradients in RNNs and is widely used. Taking LSTM as a base and adapting the weights to kernels, ConvLSTM [30] was introduced for the task of precipitation nowcasting. Multiple layers of ConvLSTM are used in this paper to obtain a sequence to sequence architecture. A further improvement of ConvLSTM is trajGRU which was proposed by Shi et al. [31] to be able to learn the *location-variant* structure for recurrent connections.

Pure convolutional neural networks have also been used to predict precipitation. As demonstrated by Bai et al. and Gering et al. [8, 12] convolutional neural architectures can outperform recurrent neural networks for a variety of sequence modelling tasks. This is the reason why many works on precipitation nowcasting have opted for pure convolutional networks [6, 7].

Due to machine learning models attempting to minimize loss, *blurry predictions* can be produced by models. This can be alleviated by using generative models which sample from the possible futures and do not seek to provide a best average fit. Generative Adversarial Networks have been successfully applied to the task of precipitation nowcasting [27].

3.2 Satellite Based Nowcasting

In their study Chen et al. built a [10] MLP to forecast radar data from satellite data. The researchers used a combination of low earth

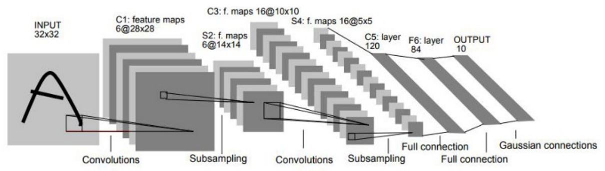

Fig. 2. Architecture of LeNet-5: used for digit recognition and introduced by Yan LeCun [20]

orbit satellite passive microwave and infrared channels from two different satellites. Their model is developed to predict up to 1.5 hours in the future by recursive predictions of the model.

A study that does not predict precipitation but uses lightning as a marker for extreme precipitation was performed by Brodehl et al. [9] this study uses a convolutional network to predict lightning events, and contributes the important observation that both the visual and infrared channels are important in differing ways to predict lightning.

The approach taken by the researchers of MetNet [33] is to combine a convolutional block for spatial downsampling, then a ConvLSTM block for temporal encoding and finally a Axial attention block [34]. MetNet is able to perform more accurate forecasts than NWP models for up to 8 hours. In this study the input data that is used is both satellite and radar data as well as the elevation, time of the year and latitude and longitude values.

4 BACKGROUND

In this section, we will describe the machine learning techniques utilized as well as relevant background information regarding the provided datasets and meterological use thereof.

4.1 Convolutional Neural Networks

Convolutional neural networks (CNNs) were initially introduced by Yann LeCun et al. in 1998 [20] and applied to the challenge of handwritten digit classification. However, the breakthrough for CNNs came later with the success achieved by Krizhevsky et al [13]. in the ImageNet paper of 2012. This work significantly advanced the state of the art of image classification.

CNNs are a class of deep learning models strongly capable of solving computer vision tasks. Unlike fully connected neural networks, which treat input data as a one dimensional vector, CNNs are designed to process higher dimensional data such as images. This distinction enables CNNs to exploit more spatial relationships and patterns in visual data as opposed to a flattened vector where these patters are not recoverable.

Additionally Convolutional neural networks reduce the amount of parameters that are needed for each layer. This reduction is caused by parameter sharing. In a traditional multi-layer perceptron weights exist for each connection while in CNNs a set of kernels are applied to the input. The kernels used in CNNs are small and are repeatedly applied across the input. This reduces the amount of parameters the network has.

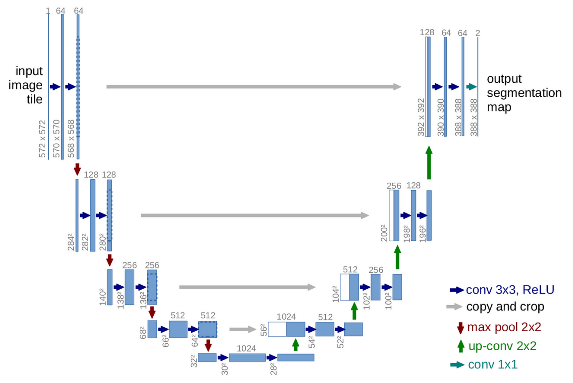

Fig. 3. U-Net architecture, encoder and decoder sections of the model form the U structure. Encoder Passes context information to the decoder through *skip connections*. [18]

4.2 U-Net

Semantic segmentation, the task of assigning a class label to each pixel in an image, is key to understand an image from a computer vision perspective. This is the starting point for U-Net which was developed by Ronneberger et al. [18] to segment images from microscopes in biomedical applications. U-Net is a fully convolutional architecture that provides accurate and detailed pixel-level predictions. U-Net is specifically designed to capture both local and global context information. The U-Net architecture gets its name from its U-shaped design (Figure 3), which consists of an encoder path and a decoder path. The encoder path resembles a traditional CNN and serves the purpose of capturing spatial information It is made out of multiple convolutional and pooling layers, where each convolutional layer extracts increasingly abstract features by convolving with learnable filters and applying the Rectified Linear Unit (ReLU) function [17]. The decoder path, on the other hand, aims to recover the spatial information lost during the pooling and convolutional operations of the encoder. It employs a series of transposed convolutional layers to gradually increase the spatial resolution. The skip connections between the corresponding encoder and decoder layers help preserve fine-grained details that would be otherwise be lost during the encoding process. U-Net is able to segment 2 dimensional images, however in some biomedical contexts 3D scans are used. To segment these kinds of inputs 3D U-Net was introduced by Çiçek et al. [21]. 3D U-Net is similar to 2D U-Net except instead of using 2D pooling and 2D convolution layers it uses it's 3 dimensional counterparts. Authors such as Brodehl. [9] have shown that the depth dimension in a 3D U-Net can be replaced by the time dimension which leads to capturing time based context information.

4.3 Convolutional LSTM

A traditional LSTM operates with vectors [23], this would mean that to process images in a LSTM these would need to be flattened to a vector, most spatial information would be lost. This is why Shi et al. introduced ConvLSTM [30]. Which replaces the learnable vectors in the LSTM with kernels, and the operations become convolutions. The equations of the ConvLSTM cell are below (equation 1-4). Each ConvLSTM cell produces a short term memory $\mathcal{H}_t$ and a long term memory C_t, the output after passing all of our sequence to the model is the short term memory $\mathcal{H}_t$. The formulas can be explained relying with the use of activation functions σ and *tanh*, sigmoid and hyperbolic tangent. The sigmoid function maps any x between 0 and 1, values between 0 and 1 are used here as the *fraction of information that is added or removed.* On the other hand the hyperbolic tangent function maps an input x between -1 and 1. Therefore when the sigmoid activation represents the fraction or percentage of information being added or removed from for example the long term memory while the *tanh* function is being used to normalize the information itself between 0 and 1. Take for example the equation for the forget gate (equation 2), noting that $*$ notates a convolution operation. The result of this equation is a matrix with values between 0 and 1 due to the sigmoid function. The f_t is used in equation 3, it is used to calculate the element wise multiplication notated $\odot$ with the previous long term memory. This allows only a fraction of the long term memory, to continue into the output of the current cell and further cells in the unrolled Convolutional LSTM. This is why it is called the forget gate.

$$i_t = \sigma\left(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \odot C_{t-1} + b_i\right) \tag{1}$$

$$f_t = \sigma\left(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \odot C_{-1} + b_f\right) \tag{2}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tanh\left(W_{xc} * X_t + W_{hc} * \mathcal{H}_{t-1} + b_c\right) \tag{3}$$

$$o_t = \sigma\left(W_{xo} * X_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \odot C_t + b_o\right) \tag{4}$$

$$\mathcal{H}_t = o_t \odot \tanh\left(C_t\right) \tag{5}$$

4.4 Attention

The concept of the attention mechanism in machine learning comes from Bahadanau et al. [14]. The authors were investigating how to assign different levels of importance to each input in a sequence to sequence model. Usually attention is computed through a shallow multi-layer perceptron (MLP) since it is expensive to add attention to a model. However this shallowness is not enough when it comes to computer vision models. Because we are dealing with inputs that are images the amount of connection becomes $(h \times w)^2$ due to the fact that every neuron in the input layer must be connected to the neuron in the output layer. To address this issue, Axial Attention was introduced by Ho et al. [22] as a means of alleviating this problem. Axial attention simplifies the computation of attention by exclusively considering the adjacent row and column, thus mitigating the exponential growth of parameters.

4.5 Radar Data

Radar instruments are employed to measure precipitation levels within a given area. These devices operate from ground level and emit microwave pulses while rotating. As these pulses encounter atmospheric particles, they disperse in various directions. A portion of the energy emitted by the radar is reflected back and recorded. By employing the relationship between speed, time, and distance, the radar can determine the particle's proximity to itself. The quantity of energy that returns to the radar following interaction with precipitation is termed *reflectivity* denoted by Z. To assess the rainfall rate in millimeters per hour, the Marshall Palmer Relationship, [16] is utilized to convert reflectivity factor. Meteorologists often prefer to use decibels relative to Z as a more convenient unit. This unit expresses reflectivity relative to a 1 millimeter drop within a cubic meter of volume (Z_0) [19].

$$dBZ = 10 \times log_{10}\left(\frac{Z}{Z_0}\right) \tag{6}$$

See table 6 for a table with dBZ values and their corresponding meterological interpretation. The range of a radar extends to a couple hundred kilometers from their origin which means that to observe a larger area a group of radars can be combined to form a composite radar image (see figure 11).

4.6 Satellite Data

Satellites play a crucial role in meteorology as they provide valuable observations of the Earth from above. Geostationary satellites orbit the earth at the same speed as the planet's rotation on it's own axis, which allow them to be fixed relative to a given longitude. This class of satellites orbit the earth at an altitude of $\approx 35786km$. Worldwide coverage can be achieved by deploying multiple satellites, for example EUMETSAT operates MSG satellites for Europe, NOAA operates GOES satellites for the Americas and the Himawari satellite is operated by the Japan Meteorological Agency, which covers the Asia-Pacific region. Focusing on Meteosat-10 satellite from which we obtain our data, this satellite produces a scan every 15 minutes. We obtain 11 satellite images corresponding to different spectral channels (see figure 12). Three main types of channels are available: visible, infrared and water vapour channels (See table 4). Visible channels are measured by the satellite when radiation from the sun reflects on the earth's surface, infrared channels on the other hand receive radiation emitted by the earth and clouds allowing for imagery even at nighttime. The wave vapour channels on the other hand, capture radiation emitted by water vapour in the upper troposphere.

5 METHODOLOGY

In this section, we will provide explanations of the methods and techniques employed in our experiments.

5.1 Engineering For Machine Learning

In order to streamline our experimental process and minimize effort, we used specific tools and techniques to make testing models more efficient. We adopted the `zenml` [5] framework to structure our training and preprocessing pipelines. This framework brings several advantages to our workflow. One notable benefit is the promotion of modularity in the design of our pipelines. Each pipeline consists of individual steps, which enhances the code's modularity. By defining a series of functions with clear inputs and outputs, we ensure that each step can be easily understood and modified as needed. Moreover, the `zenml` web application offers a convenient way to monitor the status of our pipelines. This feature provides transparency and improves comprehension by providing insights into the outputs generated at each step. For experiment tracking we used the `MLFlow` framework [2]. This makes it easy to track all metrics over all experiments. With `MLFlow` it is also possible to compare different parameters that were used during training to experimentally find the best combinations. In `MLFlow` we save each finished model as an artifact which can be directly served as a server endpoint to start providing end users access to our models. We used the pytorch lightning library. Using patterns such as the `DataModule` and LightningModule to accelerate the research process. This library abstracts the process of backpropagation and updating parameters. Furthermore pytorch ligthning [3] handles switching from training devices automatically for example cpu and cuda devices which decouples the training code from specific hardware. We made use of the `typed-settings` library [4] to allow cleanly structuring and validating settings for models. This ensures that to train a new version of a model in most cases only adjustments to the configuration file needs to be done. `typed-settings` supports passing training settings through toml configuration files, environment variables and command line options.

5.2 Data Preprocessing

For the preprocessing of data we created a pipeline which distinguishes between satellite images and radar images to process each following their own needs (See figure 6).

The pipeline begins by obtaining all necessary files, from a remote storage bucket. This is done by a Bucket Service class which uses the `boto3` [1] library to interface with the bucket and download files in the required date ranges.

In the case of satellite data, the obtained files are compressed in zip files. The pipeline handles the extraction of these files deleting any files which are not needed along the way to ease the storage requirements. Then we *reproject* the satellite images using the `satpy` package. This downsampling is done by a combination of cropping and interpolation via the nearest-neighbor algorithm.

By reprojecting we reduce the dimensions of each satellite image to `256 x 256` pixels from it's original dimensions of `3712 x 3712`. We also obtain only the geographical area of interest, specified by the coordinates for the lower corner `(50°0'0"N 0°0'0"E)` and the upper corner `(55°0'0"N 10°0'0"E)` of the region, this gives us the area centered on the netherlands with other bordering countries (see figure 4). Additionally we change the map projection to *Mercator* from the original *Geostationary*. This is done to have both input and target grids in the same map projections. After reprojecting we perform a *statistics* step where we aggregate the dataset by finding the minimum and maximum values for each channel see table 4 for the list of all channels. The statistics are necessary for the next step which is normalization. During the normalization step we perform the *Min-Max* Normalization (equation 1).

$$x_{normalized} = \frac{x - min(\bar{x})}{max(\bar{x}) - min(\bar{x})} \tag{7}$$

After finalizing the normalization step we sample an image from the dataset which is visualized to check for errors in the pipeline.

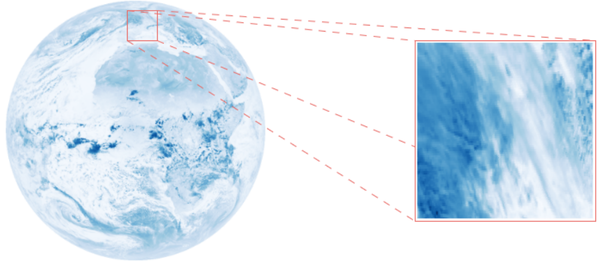

Fig. 4. Reprojection of Satellite Data: Converting from geostationary projection to mercator projection and reduce to area of interest.

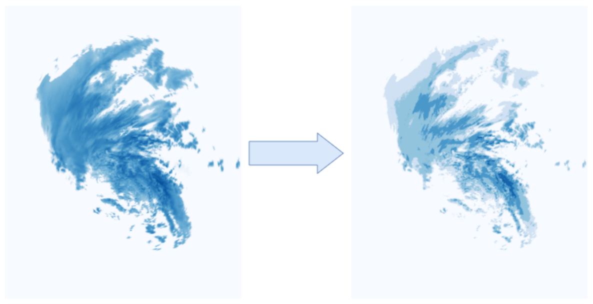

Fig. 5. Preprocessed Radar Reflectivity Data For 2023-03-10 at 11:55 UTC. Left with normalized data and right with pixels put into 8 different rainfall intensity classes.

The radar pipeline begins with the downloaded h5 files each is converted to decibels relative to Z (dBZ) (equation 8) from a *grayscale* unit ranging from 0 to 255. Using decibels relative to Z adds to the interpretability and makes it easier to weight loss functions and metrics according to the level of precipitation. The drawback of this is that to plot the image or perform image transformations many existing libraries make the assumption of either *grayscale* or *rgb* images, therefore to make use of these resources we must sometimes convert back to the *grayscale* unit.

$$dBZ(x) = x \cdot 0.5 - 32 \tag{8}$$

The preprocessing pipeline then splits into two, one step will normalize values between 0 and 1 using *Min-Max* normalization and the other step will use levels of *dBZ* to create discrete ranges of precipitation to use as classes during training (See table of classes, 3). Finally the current images in the pipeline are resized using the nearest neighbor algorithm, to avoid changing the class values with bilinear interpolation. Next identically to the satellite pipeline we sample and visualize a radar image for verification purposes.

5.3 Proposed Model Architectures

Three model architectures are tested: 3D U-Net, ConvLSTM and ConvLSTM with Attention Mechanism, all of these are suitable for classification and regression with a few alterations of the last layers.

The 3D U-Net based model is given a input patch of dimension $8 \times 11 \times 256 \times 256$. The U-Net model produces a segmented image for

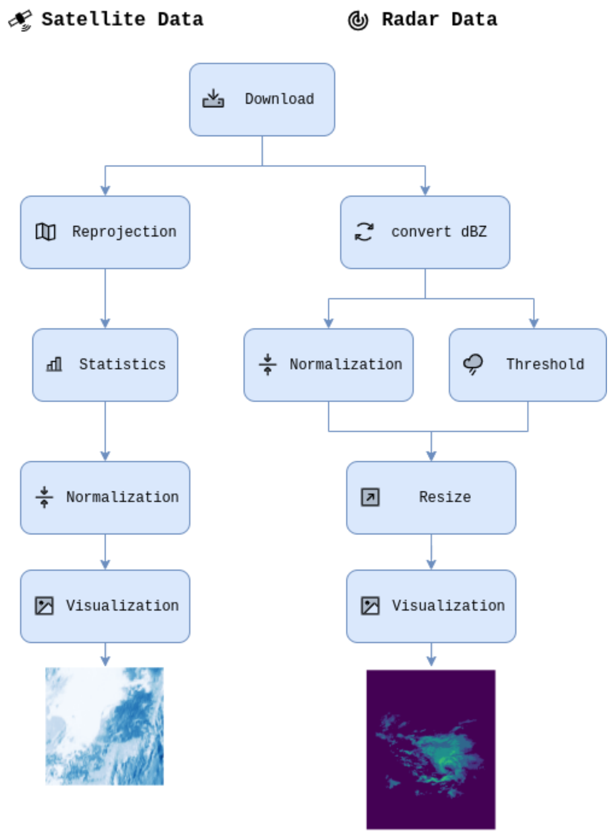

Fig. 6. Data Preprocessing Pipeline: Satellite Data and Radar data preprocessed separately

each slice of the depth dimension. Therefore we added an additional 3D convolution after the normal U-Net architecture which reduces the output of the network from the shape $8 \times 8 \times 256 \times 256$ to the desired $1 \times 8 \times 256 \times 256$ or in the case of regression to $1 \times 256 \times 256$

For ConvLSTM models, we start with 3 layers of Convolutional LSTMs with 64 filters that use a kernel size of 3. Then the short term memory of the last layer is passed to three 2D Convolutional Layers. The first Convolutional layer reduces the amount of channels to 32, the second to 16 and the last one to either 8 or 1 for classification and regression respectively. In the intermediate layers a ReLU [17] activation function is performed.

The architecture of ConvLSTM with Attention Mechanism consists of 3 layers of Convolutional LSTMs, with 64 filters with kernel size of 3. Then the short term memory of the last layer is passed to a Axial Positional Embedding Layer and a single Axial Attention layer with 4 attention heads.

5.4 Experimental Setup

We trained each model for 50 epochs. The batch size was set to 1 because of memory constraints. We used the Adam algorithm for optimization from Kingma et al [15]. All model were trained on a single NVIDIA GeForce RTX 2070 SUPER GPU. We used different losses for classification and regression. First we used Multi-Class CrossEntropyLoss for classification this first applies a log softmax activation function which normalizes the outputs produced by our models. Then it calculates the cross entropy loss between the normalized input and the target. For regression we used the MSE loss. To address class imbalance in the data we added weights for the CrossEntropyLoss. These weights were obtained by finding the frequencies of the classes in the dataset as follows:

$$weight(c) = 1 - \frac{1}{h \times w \times n} \times \sum_{i=1}^{n} \sum_{j=1}^{h} \sum_{k=1}^{w} [Y_{ijk} = c] \tag{9}$$

Referring back to the problem formulation we decided to conduct our experiments using $t = 5$, allowing for 1 hour and 15 minutes of temporal data, $h = 256$, $w = 256$, and $c = 12$.

We created 2 different types of datasets for use in the experiments. The first type is a sequence dataset, this dataset does not repeat any satellite files. It increments the starting point at each sample by the length of the satellite sequence. On the other hand to make use of all the available data we created a sliding sequence dataset which increments the starting point of the sequences by 1 meaning that the sequence moves by 1 satellite image forward. We have also handled aligning the satellite and corresponding radar target based on their timestamps.

5.5 Performance Evaluation

To evaluate the performance of our models we implemented several metrics. Each of these metrics can be used to measure the quality of a trained model. Utilizing a range of metrics allows us to better analyze the performance of a particular machine learning model. Metrics that are used to measure the performance of classification problems are different from the metrics that are used for regression. Note that we give the formulas that are used to calculate the metric for each image, in the reported results these metrics have been averaged over the entire test dataset.

5.5.1 Regression Evaluation Metrics. Regression metrics are often error functions, these functions calculate the distance between prediction and target values. When predicting more than one value, we can extend the definition of an error function by calculating the mean or sum of all distances, the distances themselves are calculated with a certain distance metric for 2 values that are in the same position in the prediction and target. This is an important distinction when considering that we predict a $h \times w$ image, to calculate the distance between the prediction and target, we will calculate the distance between each pixel then divide it by the number of pixels in the image.

The simplest metric is the *Mean Absolute Error* (equation 3). It takes the absolute value of the distance between to values since the sign of the error does not affect it's importance. The mean average error is not differentiable at $x = 0$.

$$MAE = \frac{1}{h \times w} \sum_{i=0}^{h} \sum_{j=0}^{w} |\hat{y_{ij}} - y_{ij}| \tag{10}$$

A common regression metric is *Mean Squared Error* (equation 4). MSE squares the distance between two values, so it is always positive and becomes larger exponentially faster than MAE as the distance between the two values increases.

$$MSE = \frac{1}{h \times w} \sum_{i=0}^{h} \sum_{j=0}^{w} (\hat{y_{ij}} - y_{ij})^2 \tag{11}$$

MSE punishes outliers more severely and is harder to interpret than MAE because the unit of the error is the squared original unit for example in our case (dBZ) becomes (dBZ^2). The root mean squared error solves the problem of MSE producing uninterpretable units by taking the root of the MSE (equation 5).

$$RMSE = \sqrt{MSE} \tag{12}$$

Some metrics have been designed for the task of precipitation nowcasting itself, what is particular about this case is that we place more importance on predicting the outliers than the overall data. This is because extreme values of rain are the most important to predict correctly as these cause the most damage to society. Shi et al. [31] created *BMAE* (equation 6) and *BMSE* (equation 7) because of this. These metrics weight errors higher as the target pixel value increases.

$$BMAE = \frac{1}{h \times w} \sum_{i=0}^{h} \sum_{j=0}^{w} weight(y_{ij}) \cdot |\hat{y_{ij}} - y_{ij}| \tag{13}$$

$$BMSE = \frac{1}{h \times w} \sum_{i=0}^{h} \sum_{j=0}^{w} weight(y_{ij}) \cdot (\hat{y_{ij}} - y_{ij})^2 \tag{14}$$

5.5.2 *Classification Evaluation metrics.*

Many classification metrics are based on the *Confusion Matrix*. The simplest case is a binary classification here a model predicts between two classes suppose we call them *positive* and *negative*. There are two possibilities for this output it can be either *true* or *false*. Thus there exist 4 disjoint sets (TP, TN, FP and FN) all subsets of $\hat{Y}$ the set of all predictions. Calculating values based on the cardinality of these sets helps us understand the performance of a classification model. Accuracy is a intuitive metric, it is defined as the fraction of correct classifications over the total amount of classifications.

$$Accuracy = \frac{|TP| + |TN|}{|TP| + |FP| + |TN| + |FN|} \tag{15}$$

The precision brings the focus on predicting a false positive. Precision can be a useful measure when we try to measure if a model is giving too many incorrect positive predictions.

$$Precision = \frac{|TP|}{|TP| + |FP|} \tag{16}$$

The recall gives the score focusing on when the model predicts a false negative. Recall can be useful when we want to know if the model is predicting too many incorrect negative predictions.

$$Recall = \frac{|TP|}{|TP| + |FN|} \tag{17}$$

The F1 score captures the trade-off between precision and recall, offering a single metric to measure the model's effectiveness.

$$F1Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{18}$$

The Jaccard index, also known as intersection over union (IoU) created by Jaccard [24], Can be used to evaluate a model's performance particularly for image segmentation, it quantifies the overlap between the predicted positive instances and the actual positive instances. The Jaccard index is particularly useful when evaluating models on imbalanced datasets, where it can provide a robust measure of performance by focusing on the correct prediction of positive instances while discounting true negatives.

$$JaccardIndex = \frac{|TP|}{|TP| + |FP| + |FN|} \tag{19}$$

These metrics can be extended from the binary classification case that has two classes to a general definition for n classes by redefining TP, FP, FN and TN. Another important remark is that for images, since each pixel is classified, the metrics are first calculated at the image level, then averaged over the number of samples.

6 RESULTS

In this section we will present the results of the experiments.

6.1 Data

After preprocessing we obtain `3331` satellite files and `10348` radar files see table (6). The storage size of which is equal to `387.6` Gigabytes. We split the data with a `0.8` split for testing `0.1` for validation and testing. A sample of a preprocessed satellite file can be seen in Figure 12. From this satellite image we can see that the area has been cropped from the original. The area visible corresponds to the European Netherlands in the center, on the left we have south east England and on the right we have Germany and Denmark. On the south of the image we can observe clouds in all visual and infrared channels. On the water vapour channels we can also see that there is higher degree of water vapour in the south and middle of the image, compared to the other parts of the image. A preprocessed radar image can be seen in Figure 13. A binned radar image can be seen in Figure 14 where each pixel is represented by one of the eight classes (table 3).

6.2 Trained Classification Models

Trained classification models are listed in table 1. We focus on the jaccard index to evaluate the best performing model due to class imbalance. Based on the jaccard index of **0.1249** the best performing model is the ConvLSTM without using the attention mechanism. This same model also managed to predict 53% of the radar images in the test set exactly. The model most likely obtains a perfect score when there is no rain in the image and the expected output therefore is a empty image. In second place we have the ConvLSTM with attention, and in the last place we have the 3D U-Net. We have also visually analyzed the predictions made by the models which can be seen in figure 15 and figure 16. The U-Net based model predicts low amounts of rain but generally predicts the position and movement of clouds better. Meanwhile ConvLSTM based models are more eager to predict higher rainfall classes and therefore look similar to the target, however when investigating the outputs for the entire test set it seems that the ConvLSTM based model does not learn the spatio-temporal patterns as well as the U-Net. This can be observed in Figure 17, where the LSTM based model does not segment areas covered by clouds, but seemingly randomly predicting rainfall over the whole Netherlands. Compare this with figure 18 where we can clearly see that U-Net based model knows that the incoming cloud from the south west, will produce precipitation.

6.3 Trained Regression Models

Trained classification models are listed in table 2. Similar to the classification models the ConvLSTM has the lowest error for the regression experiments. From an analysis of the metrics it can be seen that the U-Net based model performs worse compared to the ConvLSTM variant. The balanced MSE and balanced MAE metrics which weight pixels with higher amounts of rain we can see that the ratio of MSE to BMSE is 26.08 for U-Net and only 6 for ConvLSTM. Interestingly U-Net suffers from a bias in predicting low amounts of precipitation in both classification and regression.

7 DISCUSSION

Despite training for 50 epochs Models could be under-fit, this can be verified by training for more than 50 epochs. For U-Net the experimental setup is slightly different since we increased the available satellite timesteps from 5 to 8 to support the 3D Encoding, which gives the U-Net model an advantage with a larger context than other models, this is however not such a large problem since the context can be increased for the other networks and they are currently performing better, based on the metrics. One difficulty of this problem is finding good metrics that quantify the skill of the model accurately. The jaccard index works well. However from the visual inspection of predictions we would assume that U-Net is performing better. Based on these two conflicting measures it can be put into doubt if the ConvLSTM actually performs better than the U-Net model. The F1 score should also to provide a good evaluation for imbalanced prediction. However the F1 score is very high for all models this leads us to believe that the aggregation of the F1 score was not preformed correctly.

8 FUTURE WORK

A core issue with this problem is data imbalance. There is data imbalance at two levels first of all at the radar image level, frequencies for pixels containing rain are much lower than the opposite. Secondly at the dataset level, the training, validation and test datasets contain different amounts of rain. Future work could focus on created a curated dataset, that studies the amounts of rain in each partition and checks for sufficient rain events in the training dataset. Another avenue for future work could be working with alternative loss functions such as multi-class dice loss as proposed by Sudre [32] for highly imbalanced segmentations or focal loss as proposed by Lin [25]. Ensembling instead of training an end-to-end model could also be researched. For example by training a model to segment current available satellite images to radar images and then train a different model with both the satellite images and segmented radar reflectivity images. The problem also presents difficulties because the resolution of the satellite images is worse than the required output (3 vs 1 kilometer resolution). This could be improved by using the high resolution visual channel of the satellite which has a 1km spatial resolution.

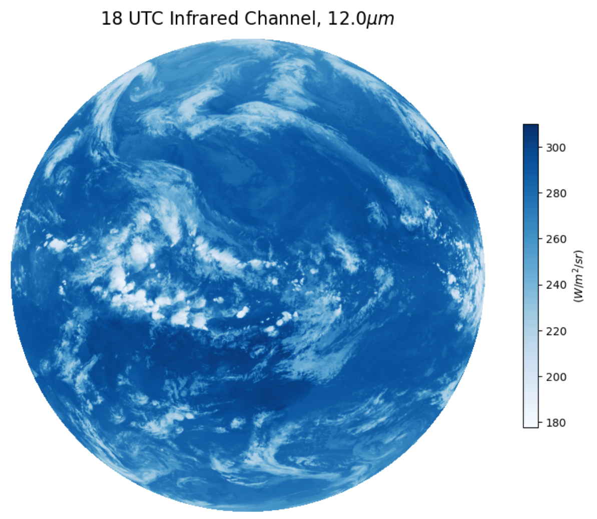

Fig. 7. Satellite Image: Infrared Channel 18UTC 12.0μm

9 CONCLUSIONS

In this paper we investigate possible methods to create a model capable of predicting precipitation based on satellite images. We propose and train 3 different model architectures. The trained models are benchmarked and compared based on several metrics. All models are framed in two different tasks: classification and regression. In the classification task we train a model to predict from 8 classes mapping to rain intensity categories. In pixel level regression we do not modify the radar reflectivity data and we train a pixel-level regression model. We find that all trained models are capable of predicting some precipitation but they require improvements to perform well in practice. This can be attributed to relatively low amounts of data being used, difference in resolution of the two types of data, data imbalances and the complexity of the problem space. The model that achieves the best score over the test dataset is the ConvLSTM model. The attention mechanism as used currently seems to make predictions worse for both classification and regression. More testing needs to be done to state certainly if ConvLSTM surpasses the performance of 3D U-Net since form a visual standpoint U-Net appears to give more meteorologically viable results.

ACKNOWLEDGMENTS

I would like to thank my supervisor from Elena Mocanu, for providing me with resources and feedback through our weekly meetings. I would like to also thank my company supervisor Christian Versloot for supporting me with the ML infrastructure, data gathering and providing suggestions regarding different machine learning approaches. Furthermore I would like to thank I would like to thank Dina Lazorkeno for her support and advice.

Table 1. Metrics on Test Set for variants of classification models trained on 50 epochs.

Models	Accuracy	Precision	Recall	F1Score	Exact Match	Jaccard Index
3D U-Net	0.9199	0.9199	0.9199	0.9199	0.0000	0.1150
ConvLSTM	**0.9999**	**0.9999**	**0.9999**	**0.9999**	**0.5385**	**0.1249**
ConvLSTM + Attention	0.9300	0.9300	0.9300	0.9300	0.0000	0.1160

Table 2. Metrics on Test Set for variants of regression models trained on 50 epochs.

Models	MAE	MSE	RMSE	BMAE	BMSE
3D U-Net	0.366	0.351	0.565	5.148	9.155
ConvLSTM	**0.032**	**0.001**	**0.034**	**0.055**	**0.006**
ConvLSTM + Attention	0.181	0.059	0.242	0.265	0.159

REFERENCES

[1] [n. d.]. Boto3. https://boto3.amazonaws.com/v1/documentation/api/latest/index.html

[2] [n. d.]. mlflow. https://mlflow.org/ Software available from mlflow.org.

[3] [n. d.]. pytorch lighting. https://www.pytorchlightning.ai Software available from pytorchligthing.ai.

[4] [n. d.]. typed settings. https://typed-settings.readthedocs.io/en/latest/ Software available from typed-settings.

[5] [n. d.]. zenml. https://www.zenml.io/home Software available from zenml.io.

[6] Shreya Agrawal, Luke Barrington, Carla Bromberg, John Burge, Cenk Gazen, and Jason Hickey. 2019. Machine Learning for Precipitation Nowcasting from Radar Images. arXiv:1912.12132 [cs.CV]

[7] G. Ayzel, T. Scheffer, and M. Heistermann. 2020. RainNet v1.0: a convolutional neural network for radar-based precipitation nowcasting. *Geoscientific Model Development* 13, 6 (2020), 2631–2644. https://doi.org/10.5194/gmd-13-2631-2020

[8] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. 2018. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. arXiv:1803.01271 [cs.LG]

[9] Sebastian Brodehl, Richard Müller, Elmar Schömer, Peter Spichtinger, and Michael Wand. 2022. End-to-End Prediction of Lightning Events from Geostationary Satellite Images. *Remote Sensing* 14, 15 (2022). https://doi.org/10.3390/rs14153760

[10] Haonan Chen, V. Chandrasekar, Robert Cifelli, and Pingping Xie. 2019. A Machine Learning System for Precipitation Estimation Using Satellite and Ground Radar Network Observations. *IEEE Transactions on Geoscience and Remote Sensing* PP (10 2019), 1–13. https://doi.org/10.1109/TGRS.2019.2942280

[11] George Cybenko. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems* 2, 4 (12 1989), 303–314. https://doi.org/10.1007/bf02551274

[12] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional Sequence to Sequence Learning. arXiv:1705.03122 [cs.CL]

[13] family=Krizhevsky given i=A, given=Alex, family=Sutskever given i=I, given=Ilya, and family=Hinton given i=GE, given=Geoffrey E. [n. d.]. ImageNet classification with deep convolutional neural networks. 60, 6 ([n. d.]), 84–90. https://doi.org/10.1145/3065386

[14] family=Bahdanau given i=D, given=Dzmitry, family=Cho given i=K, given=Kyunghyun, and family=Bengio given i=Y, given=Yoshua. [n. d.]. Neural Machine Translation by Jointly Learning to Align and Translate. Cornell University. https://arxiv.org/pdf/1409.0473

[15] family=Kingma given i=DP, given=Diederik P. and family=Ba given i=J, given=Jimmy. [n. d.]. Adam: A Method for Stochastic Optimization. ([n. d.]). https://doi.org/10.48550/arxiv.1412.6980

[16] family=Marshall given i=JL, given=Jennifer L. and family=Palmer given i=WMK, given=W. Mc K. [n. d.]. THE DISTRIBUTION OF RAINDROPS WITH SIZE. 5, 4 ([n. d.]), 165–166. https://doi.org/10.1175/1520-0469(1948)005

[17] family=Fukushima given i=K, given=Kunihiko. [n. d.]. Cognitron: A self-organizing multilayered neural network. 20, 3-4 ([n. d.]), 121–136. https://doi.org/10.1007/bf00342633

[18] family=Ronneberger given i=O, given=Olaf, family=Fischer given i=P, given=Philipp, and family=Brox given i=T, given=Thomas. [n. d.]. *U-Net: Convolutional Networks for Biomedical Image Segmentation.* Springer Science+Business Media. 234–241 pages. https://doi.org/10.1007/978-3-319-24574-4_28

[19] family=Rogers given i=RR, given=Roddy Rhodes. [n. d.]. *A short course in cloud physics.* http://ci.nii.ac.jp/ncid/BA29653451

[20] family=LeCun given i=Y, given=Yann, family=Bottou given i=L, given=Léon, family=Bengio given i=Y, given=Yoshua, and family=Haffner given i=P, given=Patrick. [n. d.]. Gradient-based learning applied to document recognition. 86, 11 ([n. d.]), 2278–2324. https://doi.org/10.1109/5.726791

[21] family=Çiçek given i=Ö, given=Özgün, family=Abdulkadir given i=A, given=Ahmed, family=Lienkamp given i=SS, given=Soeren S., family=Brox given i=T, given=Thomas, and family=Ronneberger given i=O, given=Olaf. [n. d.]. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. ([n. d.]). https://doi.org/10.48550/arxiv.1606.06650

[22] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. 2019. Axial Attention in Multidimensional Transformers. *CoRR* abs/1912.12180 (2019). arXiv:1912.12180 http://arxiv.org/abs/1912.12180

[23] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural computation* 9 (12 1997), 1735–80. https://doi.org/10.1162/neco.1997.9.8.1735

[24] Paul Jaccard. 1912. THE DISTRIBUTION OF THE FLORA IN THE ALPINE ZONE.1. *New Phytologist* 11, 2 (1912), 37–50. https://doi.org/10.1111/j.1469-8137.1912.tb05611.x arXiv:https://nph.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1469-8137.1912.tb05611.x

[25] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. 2017. Focal Loss for Dense Object Detection. *CoRR* abs/1708.02002 (2017). arXiv:1708.02002 http://arxiv.org/abs/1708.02002

[26] Rachel Prudden, Samantha Adams, Dmitry Kangin, Niall Robinson, Suman Ravuri, Shakir Mohamed, and Alberto Arribas. 2020. A review of radar-based nowcasting of precipitation and applicable machine learning techniques. arXiv:2005.04988 [physics.ao-ph]

[27] Suman Ravuri, Karel Lenc, Matthew Willson, Dmitry Kangin, Remi Lam, Piotr Mirowski, Megan Fitzsimons, Maria Athanassiadou, Sheleem Kashem, Sam Madge, Rachel Prudden, Amol Mandhane, Aidan Clark, Andrew Brock, Karen Simonyan, Raia Hadsell, Niall Robinson, Ellen Clancy, Alberto Arribas, and Shakir Mohamed. 2021. Skilful precipitation nowcasting using deep generative models of radar. *Nature* 597, 7878 (sep 2021), 672–677. https://doi.org/10.1038/s41586-021-03854-z

[28] Elena Saltikoff, Katja Friedrich, Joshua Soderholm, Katharina Lengfeld, Brian Nelson, Andreas Becker, Rainer Hollmann, Bernard Urban, Maik Heistermann, and Caterina Tassone. 2019. An Overview of Using Weather Radar for Climatological Studies: Successes, Challenges, and Potential. *Bulletin of the American Meteorological Society* 100, 9 (2019), 1739 – 1752. https://doi.org/10.1175/BAMS-D-18-0166.1

[29] NOAA's National Weather Service. [n. d.]. NWS JetStream - Life Cycle of a Thunderstorm. https://www.weather.gov/jetstream/life

[30] Xingjian SHI, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun WOO. 2015. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 28. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2015/file/07563a3fe3bbe7e3ba84431ad9d055af-Paper.pdf

[31] Xingjian Shi, Zhihan Gao, Leonard Lausen, Hao Wang, Dit-Yan Yeung, Wai kin Wong, and Wang chun Woo. 2017. Deep Learning for Precipitation Nowcasting: A Benchmark and A New Model. arXiv:1706.03458 [cs.CV]

[32] Carole H. Sudre, Wenqi Li, Tom Vercauteren, Sébastien Ourselin, and M. Jorge Cardoso. 2017. Generalised Dice overlap as a deep learning loss function for highly unbalanced segmentations. *CoRR* abs/1707.03237 (2017). arXiv:1707.03237 http://arxiv.org/abs/1707.03237

[33] Casper Kaae Sønderby, Lasse Espeholt, Jonathan Heek, Mostafa Dehghani, Avital Oliver, Tim Salimans, Shreya Agrawal, Jason Hickey, and Nal Kalchbrenner. 2020. MetNet: A Neural Weather Model for Precipitation Forecasting.

arXiv:2003.12140 [cs.LG]
[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. arXiv:1706.03762 [cs.CL]

A APPENDIX

Table 3. Available Classes for Classification Models. Based on intervals of 10 dBZ in a interval between 0 and 60

Class	dBZ range
0	$(-\infty, 0]$
1	$(0, 10]$
2	$(10, 20]$
3	$(20, 30]$
4	$(30, 40]$
5	$(40, 50]$
6	$(50, 60]$
7	$(60, +\infty)$

Table 4. Available Satellite Channels.

Channel	Type	λ
VIS006	Visual	0.6 mm
VIS008	Visual	0.8 mm
IR_016	Infrared	1.6 mm
IR_039	Infrared	3.9 mm
IR_087	Infrared	8.7 mm
IR_097	Infrared	9.7 mm
IR_108	Infrared	10.8 mm
IR_120	Infrared	12.0 mm
IR_134	Infrared	13.4 mm
WV_062	Water Vapor	6.2 mm
WV_073	Water Vapor	7.3 mm

Table 5. Reflectivity in dBZ versus Rainrate

LZ(dBZ)	R(mm/h)	R(in/h)	Intensity
5	(mm/h)	<0.01	Hardly noticeable
10	0.15	<0.01	Light mist
15	0.3	0.01	Mist
20	0.6	0.02	Very light
25	1.3	0.05	Light
30	2.7	0.10	Light to moderate
35	5.6	0.22	Moderate rain
40	11.53	0.45	Moderate rain
45	23.7	0.92	Moderate to heavy
50	48.6	1.90	Heavy
55	100	4	Very heavy/small hail
60	205	8	Extreme/moderate hail
65	421	16.6	Extreme/large hail

Table 6. Available Preprocessed Files, Split into train, validation and test data-sets. Start Data and End Date are the time of the first and last file in each array.

Start Date	End Date	Partition	Satellite Files	Radar Files
03-01 23:42	03-29 20:27	Training	2664	8275
03-29 20:27	04-02 07:57	Validation	333	1034
04-02 07:57	04-05 22:42	Testing	333	1034
03-01 23:42	**04-05 22:42**	**Total**	**3331**	**10348**

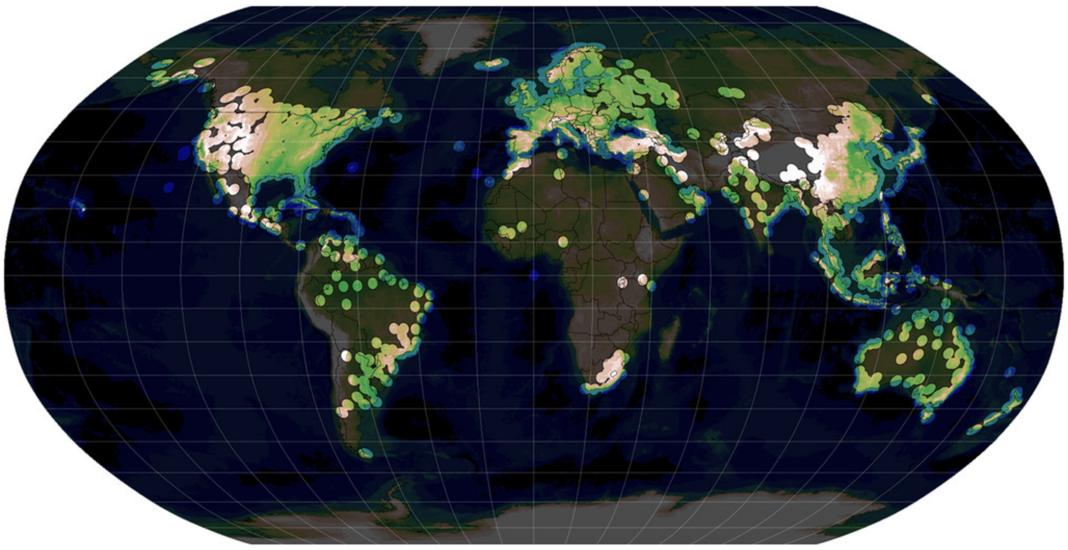

Fig. 8. Worldwide availability of radar data. Notably Oceans; the African and South American continents lack good coverage. [28]

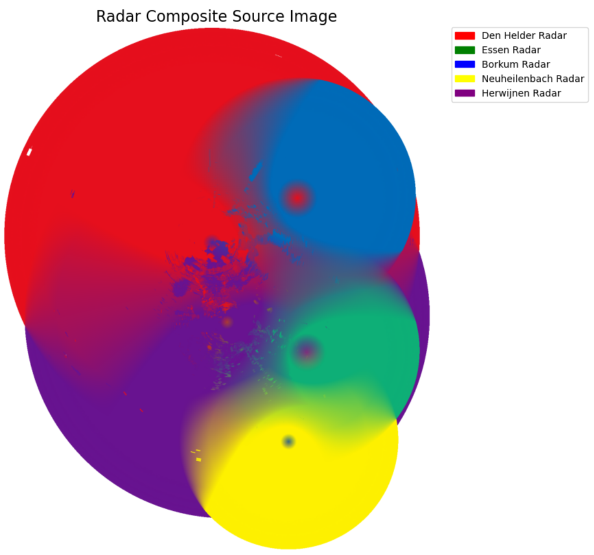

Fig. 11. Used Composite Radar Image Source. Colors mapping to 5 different radars: Den Helder, Essen, Borkum, Neuheilenbach and Herwijnen.

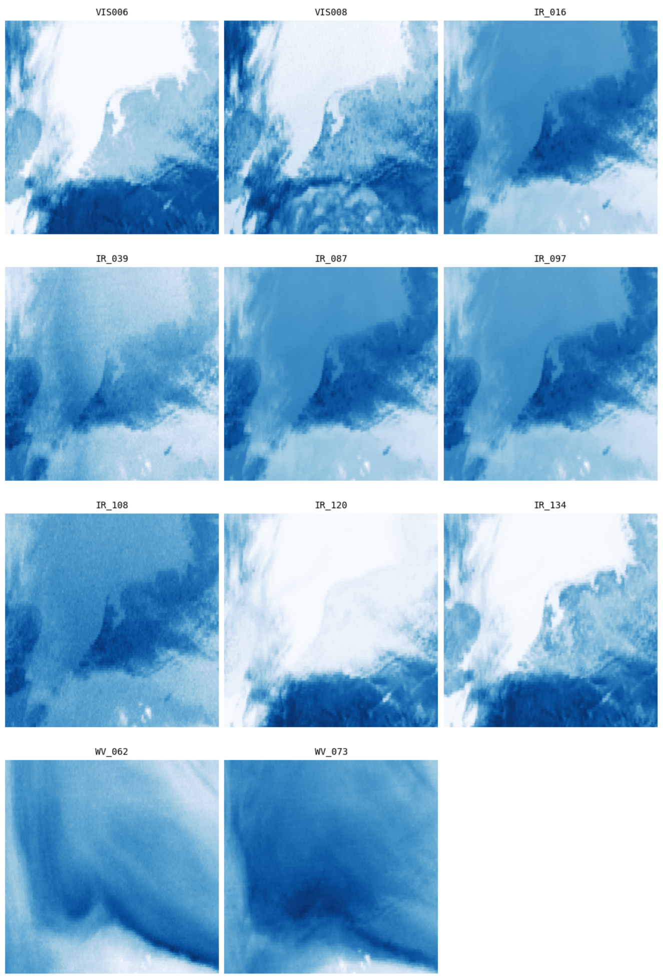

Fig. 12. Preprocessed Satellite Image. Each image corresponds to a different spectral band: 4

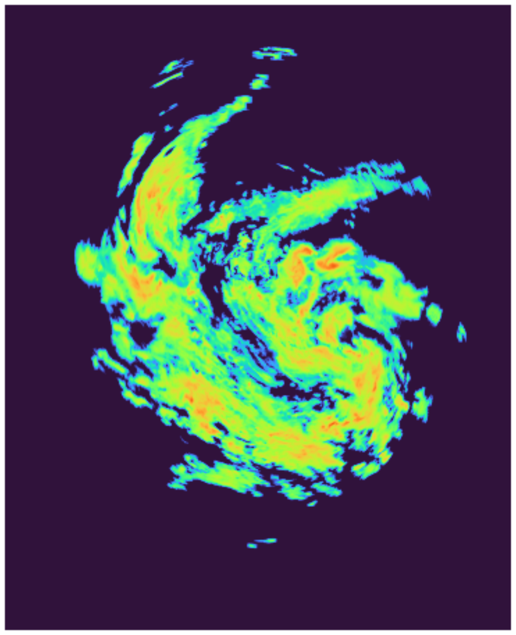

Fig. 13. Preprocessed Radar Image without continuous values.

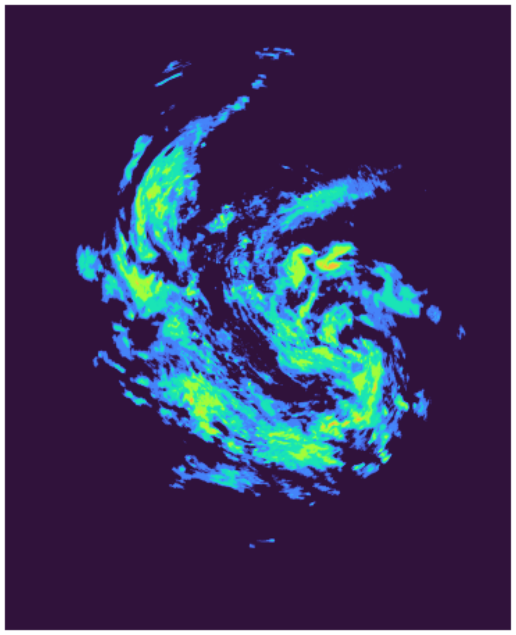

Fig. 14. Preprocessed Radar Image With Classes.

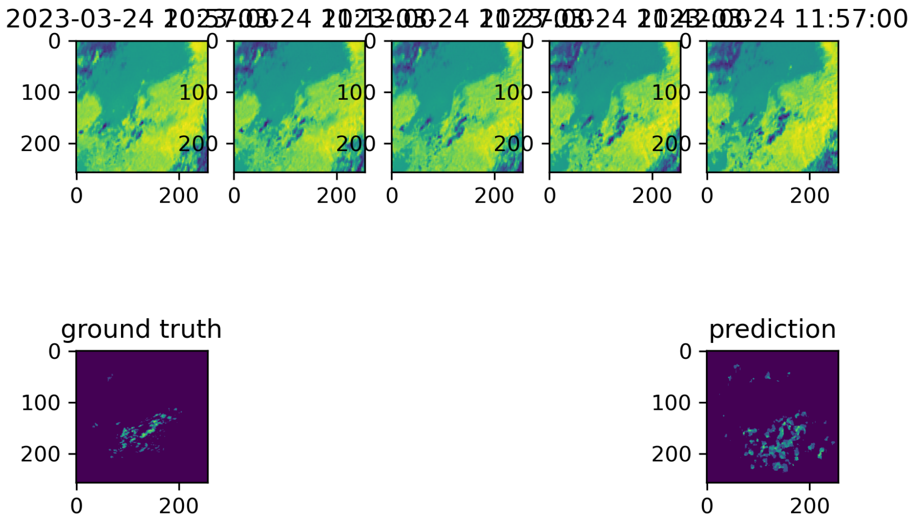

Fig. 15. Testset prediction with ConvLSTM model for 2023-03-24 12:02:00 UTC.

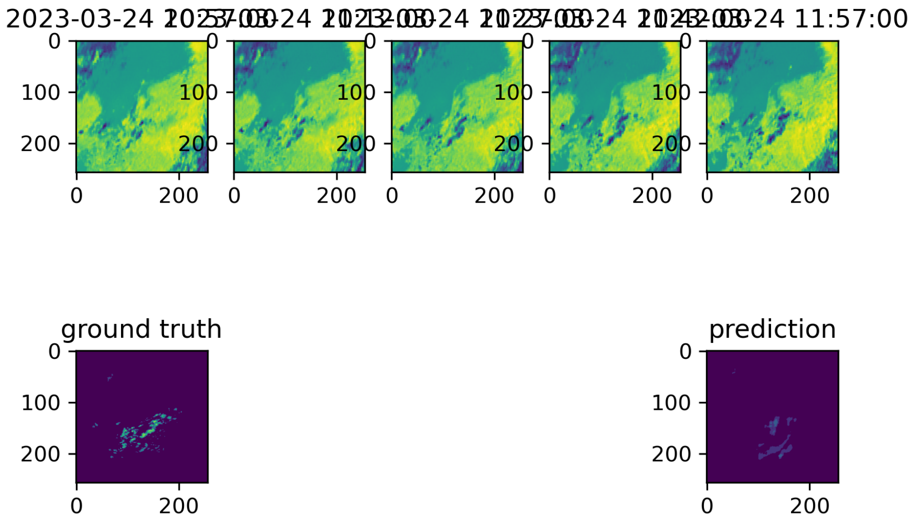

Fig. 16. Testset prediction with U-Net model for 2023-03-24 12:02:00 UTC.

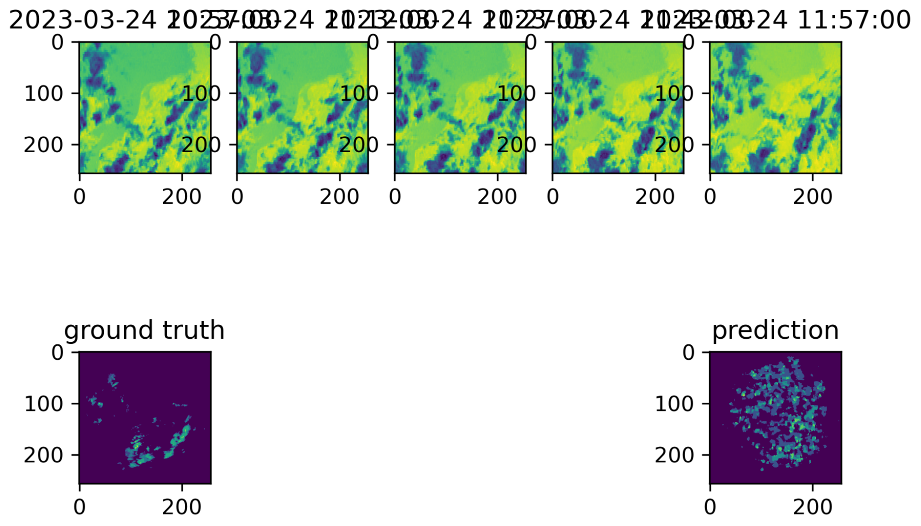

Fig. 17. Testset Prediction with ConvLSTM model. Rainfall covers areas where the sky is clear, seemingly at random.

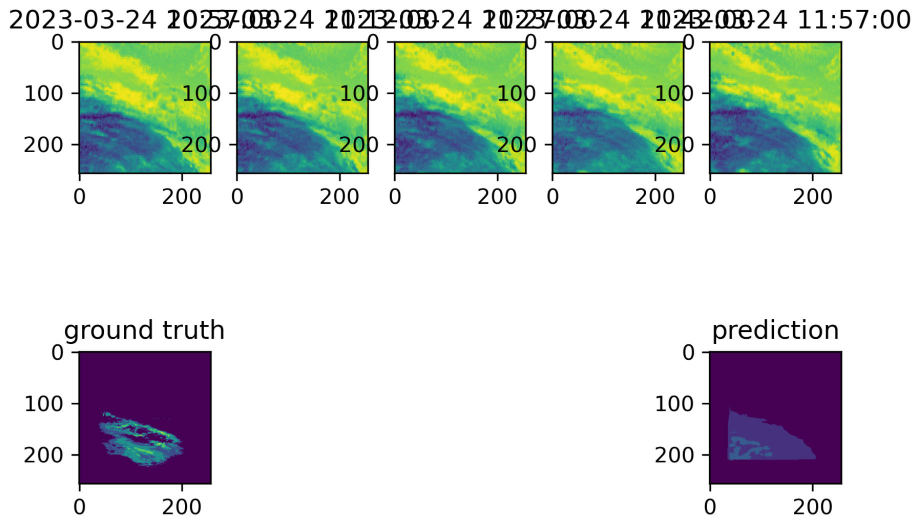

Fig. 18. Testset Prediction with 3D U-Net model. Prediction covers the same area as target but misses details. And predicted intensity is lower.