Encoder-Forecaster-Decoder: a Modular Deep Learning Framework for Cloudage Forecasting

Master's Thesis

M.H. de Bijl

6th August 2020

Committee

University of Twente

Dr. M. Poel
Dr.Ir. J. Ettema

Supervision

Alten Nederland

H.M. Logmans
G. Klarenbeek

abstract

Solar Team Twente is always looking to increase their performance in the world solar challenge. An essential part of their race strategy is the weather model, which is tasked to forecast the surface solar irradiance (SSI). The current weather model lacks detail in both spatial and temporal resolution. SSI forecasts based on geostationary satellite observations of cloudage do have the desired resolution. Since cloudage is highly correlated with SSI, forecasting cloudage is an accurate method for forecasting SSI.

Based on state-of-the-art deep learning models, the modular encoder-forecaster-decoder framework is proposed to forecast cloudage. The models that make up the elements of this framework are determined by experimentation to be the SegNet model for the encoder and decoder, and the TrajGRU cell as the forecaster. The forecasting performance of the novel model, together with the TV-L_1 optical flow method, the ConvLSTM model, and the TrajGRU model is measured for different forecast horizons on a data set containing cloud masks. In the end, it was determined that the TrajGRU model performs the best. The proposed framework lacks the ability to forecast due to inherent flaws in its design.

Contents

1 Introduction

The world solar challenge is a solar-powered car race through Australia covering more than 3000 kilometres. To win this race, you have to develop a fast car, but perhaps more importantly you have to practice a perfect race strategy. The strategy determines how fast you should drive at each moment to race optimally, given the constraints of the battery power and solar panel area, the parameters of the car, and the weather conditions. Solar Team Twente has been participating in this biennial race since 2005 and has come close to first place in several editions. However, they have never been able to seize victory. They believe their car is top-notch, however, there is room for improvement in the second part of the strategy equation. A more accurate and higher resolution weather forecast model means a more accurate and optimal strategy. This desire for an improved weather forecast model has lead them to their software partner Alten, which proposed this as a possible research project to me, which finally lead to this thesis.

There are several possible methodologies on which an improved weather forecast model can be based. However, given the requirements of the Solar Team, using satellite imagery is the most justifiable. Satellite imagery is an effective data source because it can accurately ascertain surface solar irradiance (W/m^2), which is the most significant factor in the strategy model of the solar team. The surface solar irradiance is determined from satellite imagery by taking the cloudage and position of the sun into account. Because cloudage has the most significant impact on the variability of surface solar irradiance, forecasting cloudage is an accurate method to forecast surface solar irradiance.

Forecasting cloudage from satellite imagery can be classified as a spatiotemporal sequence forecasting (STSF) problem. Much research exists in this field; however, very few have been applied to cloudage forecasting specifically. Therefore, this research will test different STSF models on cloudage forecasting. More importantly, a novel STSF framework is proposed, which employs dedicated encoder, forecaster, and decoder elements to solve the cloudage forecasting problem.

1.1 Research goal

The main goal of the research is to use the proposed framework to develop a novel model that can accurately forecast cloudage. This includes finding the optimal elements of which such a model could exist. Furthermore, the performance of the novel model, together with existing STSF models, on cloudage forecasting is analysed. This is all done to be able to make a recommendation towards the Solar Team regarding an improved forecast model for their race strategy.

1.2 Research questions

A research question is defined to aid in achieving the research goal.

> How does an encoder-forecaster-decoder modular spatiotemporal sequence forecasting deep learning model compare against state-of-the-art spatiotemporal sequence forecasting models with regards to cloudage forecasting?

As it is the goal to develop a novel model that could aid the Solar Team in regards of their race strategy, a model based on the proposed framework is compared against other available models regarding their performance on cloudage forecasting.

To answer this question, three sub-questions are defined that are used as guidelines to find the specific implementation of the novel model.

1. What are suitable models to be used as the encoder, decoder, and forecaster elements?
2. How many images as the input sequence are optimal?
3. What are the possible forecasting horizons?

1.3 Contribution

The contribution of this research is two-fold. First, the performance of existing STSF models with regards to cloudage forecasting is analysed. Because each STSF problem has its specific characteristics, the performance of STSF models on other problems cannot be directly translated cloudage forecasting.

Secondly, a modular framework is proposed which could be used as the basis for many STSF models. This framework could, in theory, be used for different applications by using a similar approach in regards to determining the exact implementation of the elements of the framework.

1.4 Outline

The structure of this thesis is as follows. First, some background is provided in section 2. Second, in section 3 relevant computer vision and deep learning techniques are described. Third, the methodologies to design, optimise, and evaluate the novel network are described in section 4. Fourth, the results of the methodology are described in section 5, and discussed in section 6. Last, the work is concluded in section 7 and future work is presented in section 8.

2 Background

This section gives a background on the problem statement and the proposed approach. First, the requirements of the solar team are discussed. Secondly, from the possible solar irradiance forecasting methodologies, it is explained how satellite imagery fits the requirements the best. Third, it is elaborated why cloud masks are the best possible data product from satellite imagery. Fourth, a short background on cloudage forecast methodologies, as well as why an STSF deep learning approach has been chosen is given. Fifth, the basic building blocks of deep learning STSF models are elaborated. And in conclusion, the scope of this research is outlined.

2.1 Solar team requirements

During the world solar challenge, the race strategy is managed by the strategist, who rides in the decision-making unit (DMU) van behind the solar car. The strategy determines at what speed the solar car should drive. The strategist uses a strategy model, that is based on the parameters of the solar car, and a weather forecast model. During the race, the strategy model is constantly updated based on the performance of the car, as well as on new weather reports. The weather forecast model forecasts the surface solar irradiance (SSI) because it is an important factor in the available energy for the solar car.

The current weather forecast model has characteristics and limitations that do not fit its purpose very well. Both the temporal and spatial resolution is too low. The model has forecasts in steps of three hours, with a forecast horizon of 120 hours ahead. A new forecast is available every 12 hours. The forecast contains values every ten km^2. An example of a strategic decision that cannot be made on the current forecast model is to determine when an upcoming cloud crosses the path of the solar car, such that the strategy model can determine whether to speed up to avoid the cloud, or to slow down to a more efficient speed to save power under the cloud.

Ideally, a new forecast model would not replace the current forecast model but complement it. This means that it should preferably have a forecast horizon of 12 hours, but at least 3 hours. It should frequently have new forecasts. Moreover, it should have a higher spatial resolution. To be able to fit into the current strategy model, the novel model should forecast solar irradiance.

2.2 Forecasting solar irradiance

Solar irradiance is the power per unit area (W/m^2), received from the sun in the form of electromagnetic radiation for a specified range of wavelengths. The solar irradiance that arrives at the surface of the earth is called surface solar irradiance (SSI). SSI depends on the height of the sun above the horizon and atmospheric conditions.

The forecasting of solar irradiance can be divided into two main methodologies: statistical time series methods and physical methods. Statistical time

series methods use historical observations to fit or train some type of model. This is very effective if the location of the historical observations matches the location of the solar power system of interest. However, due to the influence of geographical features around the area of interest, these models are not applicable everywhere [1].

Models that capture the physical properties that influence solar irradiance are more adaptable regarding the location of interest. The physical phenomenon that influences solar irradiance are mostly properties of the current state of the atmosphere. However, due to the chaotic nature of the atmosphere, it is hard to capture it in a model. Three sub methods can be described that try to deal with this chaos: numerical weather prediction (NWP), sky imagery, and satellite imagery. A summary of their properties can be found in table 1.

NWP models use current observations of the atmosphere in combination with mathematical equations to make forecasts. NWP models usually have a spatial coverage of the entire earth and a spatial resolution of two to fifty kilometres. Forecasts are usually calculated four times a day, with a temporal resolution of one to three hours. This is due to the computationally intensive dynamic equations underlying these models. A forecast for solar irradiance is inferred from the forecasted state of the atmosphere.

Sky imaging methods use camera installation aimed at the sky to observe clouds. Because cloudage has the most significant impact on solar irradiance, forecasting cloudage is an accurate method to forecast ground irradiance. The relation between cloudage and surface solar irradiance is further discussed in the next section.

Sky imaging methods forecast cloudage by observing the cloud position from two or more subsequent sky images and determining the future trajectory of the clouds using computer vision techniques. However, this technique has the limitation that it can only forecast for a small area around the camera installation. By using a mobile camera installation, this approach becomes more flexible. However, it then becomes more difficult for a model to take the different geographical features into account because they can change depending on the location of the camera. Another disadvantage of sky imaging methods is the brightness of the sun. The most important region in the sky cannot be observed well due to this limitation.

Satellite imagery methods use a similar method as sky imaging methods, where a sequence of satellite observations are used to forecast SSI.

When comparing the characteristics of the methodologies in table 1, as well as the (dis)advantages of the methods, the methodology that matches the requirements of the Solar Team the best is satellite imagery. Therefore, solar irradiance forecasting using satellite imagery will be the focus of this research. A background regarding this approach is discussed in the next section.

Technique	Sampling rate	Spatial resolution	Horizon
Persistence	-	-	Minutes
Sky imagery	30 seconds	10 to 100 meters	10s of minutes
Satellite imagery	15 minutes	1 km	5 hours
NWP	1-3 hours	2-50 km	10 days

Table 1: Characteristics of the three physical methodologies [2], and the persistence method, further discussed in section 3.1.

2.3 Satellite imagery

In this section, some necessary background is given regarding satellite imagery. There are two types of weather satellites in use: sun-synchronous and geostationary orbiting. Sun-synchronous satellites orbit the earth around the poles at always the same angle towards the sun. From their point of view, the earth rotates beneath them, resulting in a different view each time the satellite makes an orbit. Geostationary satellites orbit the earth above the equator at such an altitude that their orbit has the same duration as the rotation of the earth, locking their view of the earth at always the same position. This characteristic makes them a good data source for nowcasting, and will, therefore, be used in the research as well.

Weather satellite carry instruments which observe electromagnetic waves reflected by the atmosphere. The raw observations from these instruments are called level 0 products. The raw observations are processed, and quality control is applied, resulting in the so-called level 1 products. The level 1 products are the building blocks for more advanced level 2 and level 3 products, which use algorithms to combine the observations to obtain useful meteorological values, such as cloud properties, ocean temperatures, or locations of active fires.

One of the available products is SSI. Because the satellite's sensors cannot directly observe the SSI, it has the be inferred from the observations. Empirical methods (e.g. [3]) relate satellite observations of cloudage linearly to ground-based measurements of the SSI. The cloudage is also a product of the satellite observations. It is obtained by combining observations of reflectance and temperature of the atmosphere. Physical methods (e.g. [4]) model the relation between the satellite observations and the SSI by employing radiative transfer modelling.

While using the SSI product from satellite observations to forecast SSI seems convenient, it adds additional complexity which can be prevented. While the variability of SSI on an hourly time scale is mostly dependent on cloudage, on a daily time scale, it is mostly dependant on the day and night cycle. This can cause issues when the two dependencies start to overlap. By splitting the forecasting of SSI into two parts, namely forecasting cloudage, and determining SSI from the forecasted cloudage, a more accurate result can be obtained [5]. Another advantage of this approach is that the algorithms that determine the relation between cloudage and SSI are constantly evolving [6], and novel

methods can be incorporated in the SSI forecasting model. Therefore, in this research, the focus will be on forecasting cloudage. Determining SSI from the cloudage is explicitly not part of this research.

In the next two paragraphs, a more in-depth background is given on how cloudage is obtained from satellite observations, as well as how SSI is determined from cloudage.

2.3.1 From satellite observations to cloud masks

By algorithmically combining observations of wavelengths in the visible and infrared spectrum, clouds can be distinguished from the earth's surface. The detected clouds are represented in a cloud mask, which is a matrix with values that indicate the amount of cloudage. The indices of the matrix correspond with coordinates on earth.

The optical thickness of the clouds has the most significant impact on the amount of solar irradiance that arrives at ground level [7]. Optical thickness is the ratio of incident to transmitted radiation. An illustration of this effect can be seen in figure 1. The optical thickness represented in the cloud mask.

By using a sequence of cloud masks, future cloud masks can be forecasted, from which the SSI can be determined as will be explained in the next section.

2.3.2 From cloud masks to surface solar irradiance

By finding the relation between the optical thickness of the cloudage and the SSI, the SSI can be determined from cloud masks. There exist two types of methods that try to find this relation: conventional statistical methods, and machine learning (ML) based methods. They can be divided into two categories: Conventional statistical methods determine the relation between cloudage and SSI by fitting a function based on observations of cloudage and SSI by using statistical methods. ML-based methods use a similar method; however, now ML techniques are used to find the relation.

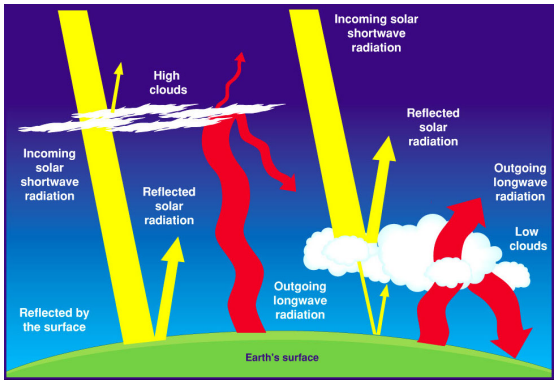


Figure 1: The effect of different types of clouds on surface solar irradiance [7].

2.4 Cloud motion

To be able to forecast cloudage, the basic underlying principles of the motion of clouds have to be understood. The chaotic nature of the atmosphere governs the motion of clouds. Due to the state of the atmosphere, clouds are moved by the wind, are formed or dissipated, and rise or descend due to temperature differences. These effects lead to a complex four-dimensional cloud dynamic, which makes forecasting on a local level very hard.

2.5 History of cloudage forecasting

Shortly after the launch of the first weather satellite, the first analysis of cloud motion was performed [8]. In those days, the analysis of cloud motion was performed to analyse and forecast wind [9]. This analysis was done by tracing a cloud position at time t_0, and superimposing the cloud position at time t_1. A vector between the two cloud blobs shows the motion between the two timestamps and is used as a forecast for the next timestamp. These vectors are also known as cloud motion vectors (CMVs). In figure 2 an illustration of this technique can be seen. Due to the ever-increasing amount of satellite data, methods to obtain CMVs in an automated manner were necessary. These automated techniques are based on computer vision techniques. The current state-of-the-art computer-vision based techniques to forecast cloudage use optical flow, of which a background is given in the next section (2.6).

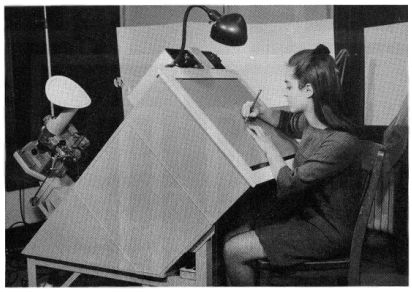
Recently, machine learning techniques have been successfully applied to the domain of cloudage forecasting as well [10], by viewing it as a spatiotemporal sequence forecasting problem. While computer vision techniques have been extensively researched in the domain of cloudage forecasting, the recent advancements in the domain of spatiotemporal sequence forecasting using deep learning are promising, and will, therefore, be the basis for this research. A background regarding the building blocks of these deep learning networks is given in section 2.7.

2.6 Optical flow

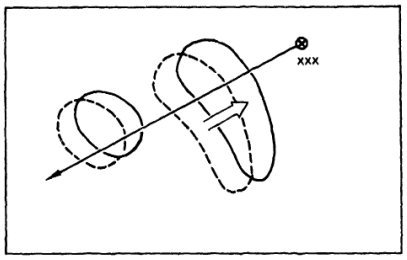
Optical flow (OF) is a technique to obtain motion vectors between two images, which can be used to forecast cloud positions. OF techniques are based on the brightness assumption:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t) \tag{1}$$

meaning that a pixel at position x, y at time t with intensity $I(x, y, t)$ has moved by $\Delta x, \Delta y$ after Δt time. This motion can be used to obtain the motion vector for that pixel. However, when trying to resolve this equation, it becomes apparent this is an equation in two unknowns, namely Δx and Δy. This is also known as the aperture problem [12]. To resolve equation 1 additional constraints are necessary. All methods that employ optical flow introduce such a constraint to estimate the optical flow.



(a) Tracing clouds from satellite imagery by hand to determine the motion vectors using a loop projector [8].



(b) An example of two clouds blobs at two time steps and a motion vector that shows the motion between the two time steps [11].

Figure 2: An illustration how motion vectors were historically obtained.

2.7 Deep learning spatiotemporal sequence forecasting

As a background, the building blocks for (STSF) deep neural networks are briefly described in this section.

2.7.1 Activations

Activation functions are applied to the output of a neural network layer. The activation functions that are used in this research can be found in table 2.

Name	**Formula**
σ (sigmoid)	$\sigma(x) = \frac{1}{1+e^{-x}}$
tanh	$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
ReLU [13]	$\text{ReLU}(x) = \max(0, x)$

Table 2: Several activation functions used in deep learning models.

2.7.2 Convolutions and pooling

The convolutional and pooling layer are used to capture the translation invariance of spatial data. For a 2D convolutional layer, the output $\mathcal{H} \in \mathbb{R}^{C_o \times H_o \times W_o}$ is determined by scanning over the input $\mathcal{X} \in \mathbb{R}^{C_i \times H_i \times W_i}$ and applying a filter $\mathcal{W} \in \mathbb{R}^{C_i \times C_o \times K_h \times K_w}$:

$$\mathcal{H} = \mathcal{W} * \mathcal{X} + \mathbf{b} \tag{2}$$

where $*$ denotes the convolution operation and $\mathbf{b}$ is the bias.

The pooling layer is similar to the convolutional layer, where the input is scanned, and a mapping function is applied. Pooling layers do not use parameters for the mapping but use aggregate functions such as max, sum or average.

The pooling layer is usually used as a downsampling layer to reduce the spatial dimensions.

2.7.3 Deconvolutions and unpool

The deconvolution and unpool layers are proposed as the inverse of the convolution and pooling layer. Deconvolution, or rather transposed convolutions, calculates its output by reversing the forward and backward pass of a regular convolution. This operation can be considered as taking the gradient of the convolution with regards to its input.

In principle, the pooling layer is not invertible. However, an approximation can be made by using additional information in the form of switch variables [14]. The switch variables contain the indices of the maxima for each pooling region. An illustration of this process can be seen in figure 3. Unpooling layers are used as upsampling layers to increase spatial dimensions. However, other upsampling methods are also used in deep learning. These are discussed in the next section.

2.7.4 Upsampling

While unpooling is an efficient manner to restore spatial resolution [15], other spatial upsampling methods are also employed in deep learning. A short description of the available methods is given here, refer to [16] for a more in depth analysis.

Transposed convolution Deconvolution, or transposed convolution, as discussed in the previous section can be used for upsampling by setting the stride of the transposed convolution larger than one [17]. An illustration of this process can be seen in figure 4.

Decomposed transposed convolution By splitting the transposed convolution in a horizontal and vertical 1D transposed convolution, a similar result can be achieved as regular transposed convolution with a reduced number of parameters [18].

Bilinear upsampling & convolution More conventional interpolation methods are combined with deep learning methods by upsampling the spatial resolution using bilinear upsampling, followed by a convolutional layer.

Bilinear upsampling & separable convolution A regular 2D convolution attempts to learn correlations across two spatial dimensions and a channel dimension. Separable convolution separates this by a specific operation for spatial correlations and a specific operation for cross-channel correlations, depthwise convolution and pointwise convolution respectively [19]. Separable convolution is combined with bilinear upsampling, which increases the spatial resolution.

Bilinear additive upsampling Bilinear additive upsampling increases spatial resolution by employing bilinear upsampling. Then, the average of a specific number of consecutive channels is taken. The ratio determines the number of consecutive channels. By setting the ratio to four, the available amount of information is preserved: the spatial resolution is increased by a factor four by the bilinear upsampling ($w \times h \rightarrow 2w \times 2h$). And at the same time, the channel dimension is decreased by a factor four.

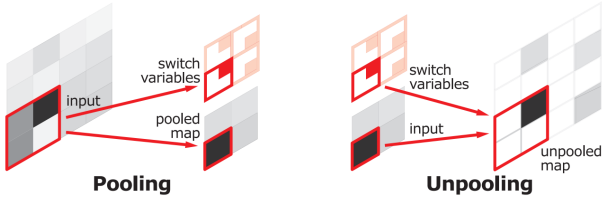
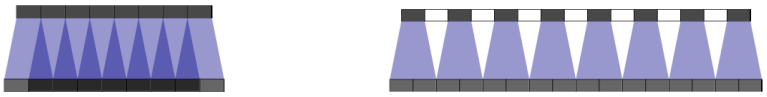


Figure 3: An illustration of upsampling using the pooling indices. The max-pool layer returns the maximum values for a specific kernel size, as well as the indices of the maximum values, captured by a switch variable. In this case, the kernel size and stride is two, which means the four by four matrix is downsampled to a two by two matrix. At the unpool layer, the switch variables are used to reconstruct the data. The maximum value is set at its original index, while the other values are filled with zeros. Taken from [20].



(a) Stride = 1 **(b)** Stride = 2

Figure 4: Illustration of upsampling by using transposed convolution. This is an example for 1D data, but the same holds for 2D data. On top is the input, below is the output. This example shows a transposed convolution with kernel size 2. When using stride=2, there is not a literal gap in the input data, however, the output is projected with a gap. In the case of stride=2, the output is then adjoined, and not overlapping, such when stride=1. The resulting output has twice the resolution of the input.

2.7.5 Recurrent neural networks

Recurrent neural networks (RNN) are a generalisation of the feedforward networks (FNN), which allow cyclical connections in the network. These so-called recurrent connections make RNN suitable for capturing of patterns over time. RNNs are trained by using stochastic gradient descent. The gradient is calculated by unfolding the RNN and applying backpropagation on this unfolded network. This type of backpropagation is also known as backpropagation through

time (BPTT). However, after processing several time steps in this manner, the unfolded network becomes very large, causing vanishing or exploding gradient problems [21]. An often applied solution for this problem is using gates to control the information flow between iterations. Two gated RNNs are discussed in the next paragraphs.

LSTM Long short-term memory networks consists of units that capture the long-term dependencies of the data. A unit consists of a cell and three gates. The three gates; the input gate, the forget gate, and the output gate, control the information flow between the current input, the previous unit's output and the cell. This approach causes the gradient to be trapped in the cell, and not to vanish. The equations that govern LSTMs are:

$$\begin{aligned} i_t &= \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci} \circ c_{t-1} + b_i\right) \\ f_t &= \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf} \circ c_{t-1} + b_f\right) \\ c_t &= f_t \circ c_{t-1} + i_t \circ tanh\left(W_{xc}w_t + W_{hc}h_{t-1} + b_c\right) \\ o_t &= \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co} \circ c_t + b_o\right) \\ h_t &= o_t \circ tanh\left(c_t\right) \end{aligned} \tag{3}$$

where $\circ$ is the hadamard product, σ is the sigmoid function, x_t is the input at time t, h_{t-1} is the hidden state of the previous time step, i_t is the input gate, f_t is the forget gate, c_t is the cell state, o_t is the output gate, and h_t is the hidden state. W and b are the weights and biases between the different gates as well as the cell, which are updated during training.

GRU Gated recurrent unit networks are very similar to LSTM networks. However, they lack the output gate, resulting in less parameters compared to the LSTM. For some cases, GRUs have similar performance compared to LSTM [22]. However, LSTMs are strictly stronger [23]. The equations that govern GRUs are very similar to LSTMs:

$$\begin{aligned} z_t &= \sigma\left(W_{xz}x_t + W_{hz}h_{t-1} + b_z\right) \\ r_t &= \sigma\left(W_{xr}x_t + W_{hr}h_{t-1} + b_r\right) \\ \hat{h}_t &= tanh\left(W_{xh}x_t + r_t \circ \left(W_{hh}h_{t-1}\right) + b_h\right) \\ h_t &= \left(1 - z_t\right) \circ \hat{h}_t + z_t \circ h_{t-1} \end{aligned} \tag{4}$$

where z_t is the update gate, r_t is the reset gate, $\hat{h}$ is the new information, and h_t is the output. W and b are the weights and biases between the gates, which are updated during training.

Stacking An important factor in recurrent networks is the number of stacked recurrent elements. By stacking multiple recurrent elements, a deep recurrent network can be build which can obtain patterns over time well [24]. An example of a stacked network can be found in figure 5.

In a multilayer RNN, the input $x_t^{(l)}$ for layer l ($l \geq 2$) at time t is the output of the previous layer $h_t^{(l-1)}$. This $\text{RNN}_t^{(l)}$ also receives the hidden state of its previous iteration $h_{t-1}^{(l)}$, or an initial state (usually set to $\mathbf{0}$) if no previous iteration is available.

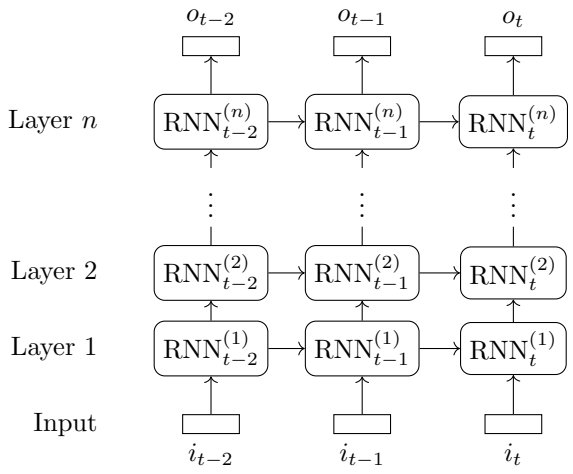


Figure 5: This illustration shows an unfolded stacked recurrent network. The RNN units in the same layer are in essence the same unit, but at different time steps.

2.7.6 Loss

The gradient for updating the weights of a deep network is the gradient towards a loss function that defines the difference between the output of the network and the expected output (ground truth). The loss function is chosen such that it fits the learning goal of the network.

MSE The de facto standard for image reconstruction is mean squared error (MSE) loss [25]:

$$MSE\left(I, \hat{I}\right) = \frac{1}{w \cdot h} \sum_{i=1}^{w} \sum_{j=1}^{h} \left(I_{i,j} - \hat{I}_{i,j}\right)^2 \tag{5}$$

where I is the ground truth image, $\hat{I}$ is the output image of the network, and w, h are the dimensions of the image.

SSIM Structural similarity index measure (SSIM) is designed as a loss function that more closely measures the perceived loss for humans [26]. SSIM is

based on three comparison measures that measure the luminance, contrast and structure between two images, resulting in the following equation:

$$SSIM(I, \hat{I}) = 1 - \frac{\left(2\mu_I\mu_{\hat{I}} + c_1\right)\left(2\sigma_{I\hat{I}} + c_2\right)}{\left(\mu_I^2 + \mu_{\hat{I}}^2 + c_1\right)\left(\sigma_I^2 + \sigma_{\hat{I}}^2 + c_2\right)} \tag{6}$$

where μ_I and $\mu_{\hat{I}}$ are the average of I and $\hat{I}$ respectively, σ_I^2 and $\sigma_{\hat{I}}^2$ are the variance of I and $\hat{I}$ respectively, $\sigma_{I\hat{I}}$ is the covariance of I and $\hat{I}$, and c_1, c_2 are used to stabilise the equation and are set by:

$$c_1 = (k_1 L)^2 \qquad\qquad c_2 = (k_2 L)^2 \tag{7}$$

where k_1 and k_2 are small constants, default at $0,01$ and $0,03$ respectively, and L is the dynamic range of the pixel values, typically $L = 2^{\#bits\ per\ pixel} - 1$.

2.7.7 Performance metrics

The same metric usually measures the performance of a network as its loss function. However, other variants exist as well. Forecast skill, for example, is often used to indicate the performance of some model in regards to a baseline model. A skill measure determines the skill score. For example, using the MSE as the skill metric:

$$\text{skill score} = 1 - \frac{MSE_{model}}{MSE_{baseline}} \tag{8}$$

2.8 Scope

Based on the requirements of the solar team, the methodologies that will be used in this research are determined. The resulting scope is a deep learning STSF model that can forecast a cloud mask, based on previous cloud masks obtained from geostationary satellite imagery.

3 Related work

In this section, the related work concerning cloud motion forecasting is discussed.

3.1 Persistence

Persistence forecasting is a basic forecasting model. It assumes that the obtained clouds do not move after the last input time frame. For short forecast horizons, this model performs reasonably well. The research shows that up to 1 hour, this model is a good baseline. For longer horizons, the results diminish.

3.2 TV-L_1

TV-L_1 is an optical flow-based model. As explained above, to resolve equation 1, additional constraints are needed. The TV-L_1 model proposed by Zack et al. [27] uses total variation (TV) as a regularisation term as this additional constraint. The L_1 norm is used as the error term between the two images, resulting in the following error function, which is minimised towards u and v to obtain the optical flow:

$$E(I_0, I_1) = \iint \{\lambda \left|I_0(x, y) - I_1(x + u, y + v)\right| + |\nabla u| + |\nabla v|\} \, dxdy \quad (9)$$

where I_0 and I_1 are the input images, u and v are the resulting flow, and λ is the weight between the L_1 error term and the TV regularisation term.

While this approach is good at finding displacements at a pixel level, it fails for larger displacements. Perez et al. developed an algorithm to determine TV-L_1 OF for larger scales as well, by employing a pyramid of scales [28]. A pyramid is a set of downscaled versions of the input image at different scales. The optical flow obtained at a coarse level in the pyramid are used as a starting point for obtaining the optical flow at a more detailed level.

Urbich et al. [29] applied the TV-L_1 method to cloud motion forecasting using satellite imagery. It outperforms another OF method, and has an error of 10% for 30 minutes forecasts, to 20% for 2-hour forecasts.

3.3 Convolutional LSTM

Shi et al. researched the usage of machine learning for the purpose of nowcasting of precipitation. They proposed an extension on LSTMs to use convolutional structures at the gates inside the LSTM cells called convolutional LSTM

(ConvLSTM) [30]. The LSTM equations in 3 are updated as follows:

$$
\begin{aligned}
i_t &= \sigma\left(W_{xi} * \mathcal{X}_t + W_{hi} * \mathcal{H}_{t-1} + W_{ci} \circ \mathcal{C}_{t-1} + b_i\right) \\
f_t &= \sigma\left(W_{xf} * \mathcal{X}_t + W_{hf} * \mathcal{H}_{t-1} + W_{cf} \circ \mathcal{C}_{t-1} + b_f\right) \\
\mathcal{C}_t &= f_t \circ \mathcal{C}_{t-1} + i_t \circ tanh\left(W_{xc} w_t + W_{hc} h_{t-1} + b_c\right) \\
o_t &= \sigma\left(W_{xo} * \mathcal{X}_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \circ \mathcal{C}_t + b_o\right) \\
\mathcal{H}_t &= o_t \circ tanh\left(\mathcal{C}_t\right)
\end{aligned}
\tag{10}
$$

where $*$ is the convolutional operation, input $\mathcal{X}$, hidden state $\mathcal{H}$, cell state $\mathcal{C}$, and gates i_t, f_t, o_t are 3D tensors, with two spatial dimensions and a channel dimension.

By using the convolution operation in the gate-to-gate transitions, the spatial awareness is improved compared to the regular LSTM. Another advantage is the reduced number of parameters, while the convolutional kernel, represented by the weights, is shared within a gate-to-gate transitions. For a regular LSTM, each connection in the gate-to-gate transitions has its own weight, also known as a fully connected (FC) network.

Shi et al. build a network using ConvLSTM as a building block, using an encoder-forecaster structure [31], as illustrated in figure 6. The encoder compresses the input sequence into a hidden state, and the forecaster uses this to make a prediction. The link between the encoder and forecaster is achieved by copying the last hidden state of the encoder to the initial states and cell outputs of the forecaster.

This network of stacked ConvLSTM layers in a encoder-decoder structure is compared against a similar regular LSTM network, as well as a state-of-the-art optical flow method. The ConvLSTM network outperformed both other methods, showing that it is suitable for precipitation nowcasting, and thus potential for nowcasting the complex dynamics of cloudage.

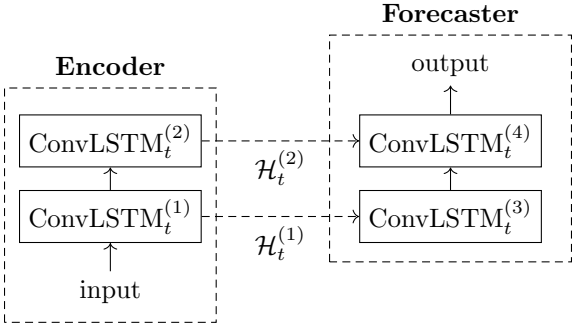


Figure 6: The architecture of the ConvLSTM based network, taken from [30].

3.4 Trajectory GRU

Shi et al. followed ConvLSTM up with an extension on GRUs. They describe a characteristic of ConvLSTM that harms its potential for nowcasting precipitation, namely that that the convolutional recurrence structure is location-invariant, while motion of clouds is location-variant. Scaling and rotation of clouds will therefore be correlated less in consecutive frames. They propose an extension on ConvGRU which takes the location of features into account. ConvGRU is an extension on GRU, similarly as ConvLSTM to LSTM. The GRU equations from 4 become:

$$\begin{aligned}
\mathcal{Z}_t &= \sigma\left(\mathcal{W}_{xz} * \mathcal{X}_t + \mathcal{W}_{hz} * \mathcal{H}_{t-1} + b_z\right) \\
\mathcal{R}_t &= \sigma\left(\mathcal{W}_{xr} * \mathcal{X}_t + \mathcal{W}_{hr} * \mathcal{H}_{t-1} + b_r\right) \\
\mathcal{H}'_t &= f\left(\mathcal{W}_{xh} * \mathcal{X}_t + \mathcal{R}_t \circ \left(\mathcal{W}_{hh} * \mathcal{H}_{t-1}\right) + b_h\right) \\
\mathcal{H}_t &= \left(1 - \mathcal{Z}_t\right) \circ \mathcal{H}'_t + \mathcal{Z}_t \circ \mathcal{H}_{t-1}
\end{aligned} \tag{11}$$

where f is the activation function.

The deficiency of ConvGRU and ConvLSTM is that the connection structure and weights are fixed for all locations in the input image. However, due to scaling or rotation, different neighbours around a location are relevant, due to the direction of the flow of the motion. Shi et al. propose Trajectory GRU [32], which employs a sub network that generates neighbourhoods with relevant neighbours for each location, using the previous hidden state and the current input. The neighbourhoods are incorporated into the GRU equations as follows:

$$\begin{aligned}
\mathcal{U}_t, \mathcal{V}_t &= \gamma\left(\mathcal{X}_t, \mathcal{H}_{t-1}\right), \\
\mathcal{Z}_t &= \sigma\left(\mathcal{W}_{xz} * \mathcal{X}_t + \sum_{l=1}^{L} \mathcal{W}_{hz}^l * \operatorname{warp}\left(\mathcal{H}_{t-1}, \mathcal{U}_{t,l}, \mathcal{V}_{t,l}\right)\right) \\
\mathcal{R}_t &= \sigma\left(\mathcal{W}_{xr} * \mathcal{X}_t + \sum_{l=1}^{L} \mathcal{W}_{hr}^l * \operatorname{warp}\left(\mathcal{H}_{t-1}, \mathcal{U}_{t,l}, \mathcal{V}_{t,l}\right)\right) \\
\mathcal{H}'_t &= f\left(\mathcal{W}_{xh} * \mathcal{X}_t + \mathcal{R}_t \circ \left(\sum_{l=1}^{L} \mathcal{W}_{hh}^l * \operatorname{warp}\left(\mathcal{H}_{t-1}, \mathcal{U}_{t,l}, \mathcal{V}_{t,l}\right)\right)\right) \\
\mathcal{H}_t &= \left(1 - \mathcal{Z}_t\right) \circ \mathcal{H}'_t + \mathcal{Z}_t \circ \mathcal{H}_{t-1}
\end{aligned} \tag{12}$$

where L is the number of allowed neighbours, $\mathcal{U}_t, \mathcal{V}_t \in \mathbb{R}^{L \times H \times W}$ contain the neighbourhoods generated by sub network γ, $\mathcal{W}_{hz}^l$, $\mathcal{W}_{hr}^l$, $\mathcal{W}_{hh}^l$ are the weights for projecting the channels, and the warp function selects the neighbours pointed by $\mathcal{U}_t, \mathcal{V}_t$ from $\mathcal{H}_{t-1}$ using bilinear interpolation.

TrajGRU is incorporated in a similar Encoder-Forecaster structure as [30]. However, downsampling using convolution with stride is done between the TrajGRU layers in the encoder, as well as upsampling using deconvolution with stride in the forecaster. Furthermore, the forecaster is in the reverse order of the encoder, which was not the case for the ConvLSTM network. This is because hidden state of the higher layers contain more coarse information, which

is refined by the hidden states of the lower layers. An illustration of this network can be found in figure 7. Because of the architecture of this network, the performance of TrajGRU is compared directly with ConvLSTM, by swapping the ConvRNN layers with the respective implementations. It was shown that TrajGRU outperforms ConvGRU, as well as ConvLSTM. All these convolutional recurrent neural networks are denoted as ConvRNN models.

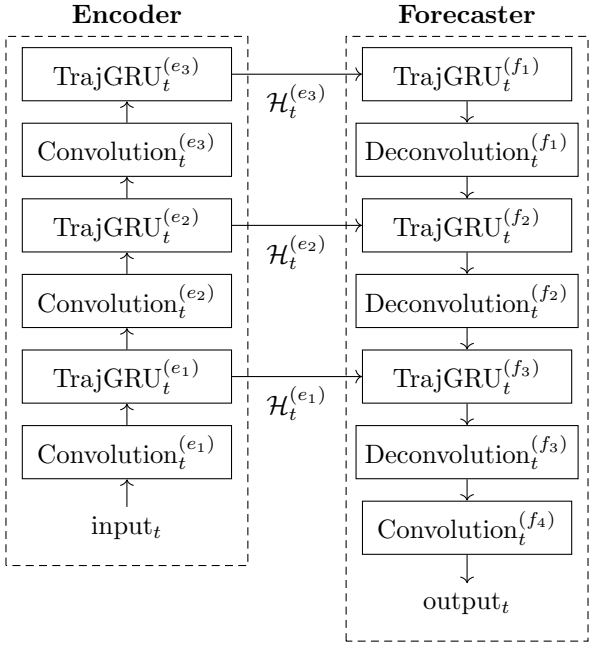


Figure 7: Architecture of TrajGRU, taken from [32].

4 Methodology

Based on the related work, a novel framework is conceptualised, which will be discussed in-depth in the next section. The approach to answering the research question using this framework is as follows. First, suitable models are selected for the encoder, forecaster, and decoder elements. Second, the data that will be used for the training, testing, and evaluation are explained. Third, from the possibly multiple suitable models, the optimal combination is determined. This is done by implementing each element, train the combined elements as a network, and compare the results on a test set. Fourth, the obtained combination of elements is optimised by tuning the hyperparameters of this network. And last, the performance of this novel network is compared against existing models, such as described in the related work.

4.1 Encoder-Forecaster-Decoder framework

A good starting point in developing a novel neural network is to look at existing models that try to solve the same or a similar problem. From the research discussed in the related work, two relevant deep neural networks are described: ConvLSTM based, and TrajGRU based. They both employ some type of encoding structure, which encodes the input data to reduces its complexity. As well as a corresponding decoding element, which reconstructs the encoded data to a usable forecast image. This observation has also been made by Shi et al. in their survey of machine learning for STSF [10], who labelled it as the Encoder-Forecast framework. The encoder and forecaster parts of the TrajGRU and ConvLSTM network have been labelled in their respective illustrations of the networks (figure 6, 7).

These intertwined encoder and decoder networks of ConvRNN layers result in a complex network architecture. However, given this notion, and the usage of encoder and decoder elements by these networks, it begs the question of whether the problem can be solved by using distinct encoder, forecaster, and decoder elements. By using a dedicated encoder and decoder structure which would tackle the spatial complexity, and a dedicated forecast element which would tackle the temporal complexity, the forecasting problem might be solved as well. The advantages of this Encoder-Forecaster-Decoder framework are: a more simple network which might be easier to train, the possibility to exchange elements for other variants which might perform better, and the possibility to use pre-trained elements, which decreases overall training time. An abstract overview of this framework is illustrated in figure 8.

4.2 Model selection

This section describes the selection criteria for the encoder, decoder and forecaster, as well as the approach for finding suitable models for these elements. The selected models will also be discussed here, while these form the basis for the rest of the methodology.

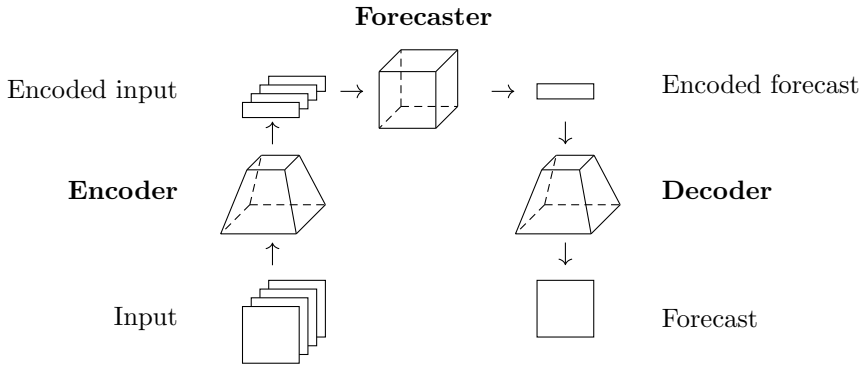


Figure 8: A schematic overview of the proposed framework.

The general approach for finding suitable models for the three elements is made by dissecting the state-of-the-art deep learning STSF models for their basic building blocks. These building blocks will have a state-of-the-art approach themselves. These will be the models considered for selection.

First, the selection criteria, suitable models, and the selected models for the encoder and decoder are discussed, then for the forecaster.

4.2.1 Encoder & decoder

The primary criterion for the encoder is to downsample the spatial resolution to decrease the complexity for the forecaster, and for the decoder to upsample the spatial resolution to restore the original size. The second criterion is for the encoder and decoder to be independent because the decoder should be able to decode the forecast made by the forecaster, which computes the forecast based on encoded input. If there is some kind of dependency between the encoder and decoder, the encoded forecast might not be able to be decoded correctly. The last criterion is that the encoder and decoder should be compatible with the forecaster. This means that the output of the encoder should be able to be processed by the forecaster and that the output of the forecaster should be able to be processed by the decoder.

The basic building block for down- and upsampling of spatial resolution used in the state-of-the-art STSF models are convolutions, (un)pooling, deconvolutions, and up- and downsampling [10, 33, 32, 34, 35]. The state-of-the-art models of these building blocks, which fit the criteria, are convolutional encoder-decoders. Convolutional encoder-decoders stack stages of convolutions and pooling to form an encoder, and the inverse as a decoder. Many of such convolutional encoder-decoders exist [20, 15, 36, 37, 38, 39], however, according to Badrinarayanan et al., their SegNet model outperforms all others. Therefore, the SegNet model is selected for the basis of the encoder and the decoder.

4.2.2 Forecaster

The basic criterion for the forecast is to be able to capture temporal patterns. Furthermore, it should have some way to capture spatial patterns as well. While the encoder and decoder reduce the complexity of the spatial resolution, the forecaster should have spatial awareness. Otherwise, spatial correlations might be lost to some degree.

From the state-of-the-art STSF models, two building blocks are obtained that fit these criteria for the forecaster, namely convolutional LSTM [30] and trajectory GRU [32]. In the state-of-the-art, these building blocks are combined with convolutional layers which down- and upsample the data in between the ConvRNN layers. In the proposed framework, these operations are strictly separated. Therefore, the ConvLSTM and TrajGRU building blocks will be considered standalone as the forecaster.

4.3 Data

As described in section 2.3, cloud masks from geostationary satellites are the data type utilised in this research. There are roughly five geostationary platforms that could provide the necessary data, which are summarised in table 3.

Country	Operator	Platform	Satellite	Location
USA	NOAA	GOES	GOES-14	105°W
			GOES-15	128°W
			GOES-16	75,2°W
			GOES-17	137,2°W
EU	EUMETSAT	MSG	Meteosat-8	41,5°E
			Meteosat-9	3,5°E
			Meteosat-10	9,5°E
			Meteosat-11	0°E
Russia	Roscosmos	Elektro-L	Elektro-L No.1	14,5°W
			Elektro-L No.2	77,8°E
			Elektro-L No.3	165,80°E
India	INSAT	INSAT	INSAT-3A	93,5°E
			INSAT-3D	82°E
			INSAT-3DR	74°E
Japan	JMA	Himawari	Himawari 8	140,7°E
			Himawari 9	140°E

Table 3: An overview of the current geostationary weather satellites in use.

When deciding which platform to use, the following factors are relevant:

1. *Spatial resolution.* The spatial resolution is the number of pixels available for a certain area. When the resolution is higher, more information is available, which improves the forecast.

2. *Temporal resolution.* The temporal resolution is the measure in the time between subsequent images. With a higher temporal resolution, there is more information available on the motion of the clouds, which results in a more accurate forecast.

3. *Spatial coverage.* The spatial coverage is the amount of terrain that is covered by the images. This measure is less relevant for the accuracy of the forecast and more relevant for the applicability of the forecast model in certain areas. If the spatial coverage encompasses the full disk of earth, the forecast will not be accurate for local forecasts. And vice versa, if the spatial coverage is a few hundred kilometres, the forecast model will probably not handle full-disk satellite images very well.

4. *Temporal coverage.* The temporal coverage is the amount of time that the data set covers. This is relevant because the cloud types and the prevailing wind are sometimes depended on the season.

5. *Availability.* The ease of obtaining the data, and the format of the provided data, is important for processing the data into the required format for the models.

The satellites of the Japanese Meteorological Agency (JMA) at 140°east cover Australia as well. However, their data sets were not publicly accessible, and requests for access proved to be fruitless. While it would be beneficial to have a data set that covers the race location in Australia, it is not essential. The weather around Australia on a synoptic scale is unique due to its size and geographic position. However, on a mesoscale at which this research is focused, the dynamics that influence the motion of clouds occur at other places in the world as well.

Given that the desired data set was unavailable, and other data sets also capture the dynamics of cloud motion adequately, the CLoud property dAtAset using SEVIRI - Edition 2 (CLAAS-2) [40] data set has been chosen. This data set is primarily chosen because it was readily available. Furthermore, it has high temporal coverage and temporal resolution.

CLAAS-2 is a dataset developed by the Satellite Application Facility on Climate Monitoring (CM SAF), a joint effort of multiple national meteorological services coordinated by the EUMETSAT. This data set is based on observations made by the SEVIRI sensor on the geostationary Meteosat second generation (MSG) satellites. A quick overview of the characteristics of this data set can be found in table 5.

The cloud masks in this data set consist of one of five possible values:

0: **Non-processed**. Because the cloud masks are stored in a square matrix, some matrix cells fall outside the earth's disk and are labelled non-processed.

1: **Cloud free**.

2: **Cloud contaminated**.

3: **Cloud filled**.

4: **Snow or ice**. For this research, snow or ice cells are considered cloud-free, as they do not influence the SSI.

A visualisation of such a cloud mask can be found in figure 9.

The algorithm that determines the cloud mask from the satellite observations uses a combination of thresholds for seven of the twelve channels available on SEVIRI. An overview of the used channels and their purpose can be found in table 4. This algorithm was validated by comparing its obtained cloud mask with the observations from CALIOP, a LIDAR sensor on CALIPSO. CALIOP has much higher accuracy, due to its active technology, and the fact that CALIPSO is a polar-orbiting satellite. The SEVIRI-based algorithm has a cloud detection rate of 87%, and a false positive rate of 17% [41], using the CALIOP observations as a baseline.

Because the spatial resolution of the cloud mask is quite high, they are divided into 10x10 sub cloud masks. This is done for two reasons. Foremost due to the goal of the model: by having smaller sections, the model should be able to detect smaller movement, which aligns with the requirements of the solar team. Secondly, this is done due to hardware limitations. The original size of the cloud masks requires much memory to store. The sub cloud masks that contain non-processed data points are not considered in this research. An illustration of the selected sub masks can be found in figure 9.

The data set is divided into three categories: train data, validation data, and test data: 72%, 18% and 10% respectively. The training data is used to train the models. The validation data is used to validate the configuration of the models. And the test data is used to analyse the models.

Channel	**Usage**
0,6 µm	The two visible wavelength channels are used
0,8 µm	to observe clouds during daytime.
1,6 µm	The near infrared channel is used to distinguish clouds from snow or ice.
3,9 µm	The medium infrared channels are used to
8,7 µm	oberve clouds during the day, and especially
10,8 µm	during night when the visible channels are
12,0 µm	not usefull.

Table 4: The different channels of the SEVIRI satellite used by the cloud mask algorithm of CM SAF [42].

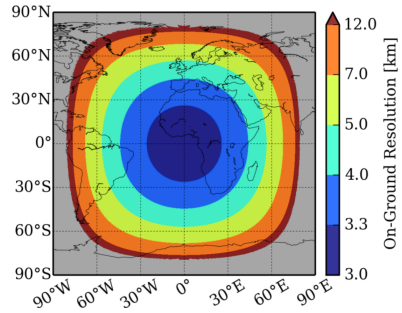
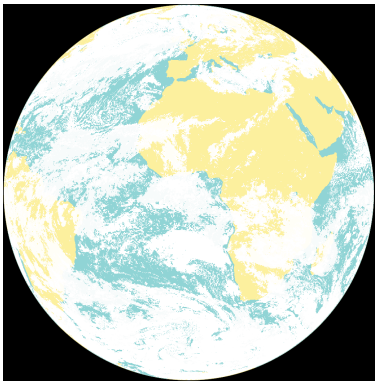
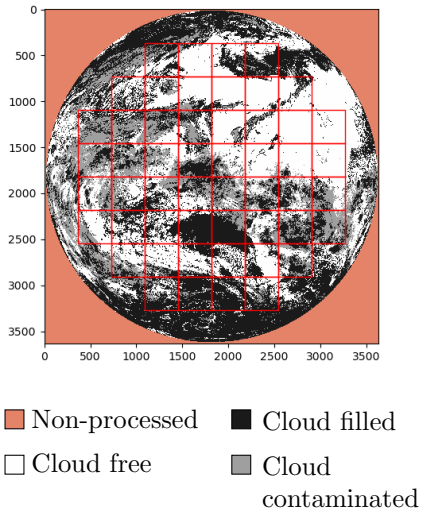
Property	Value
Spatial resolution	Due to the curvature of the earth, the spatial resolution diminishes from 3 km at nadir towards 11 km at the edge of the field of view:  Image taken from [41].
Temporal resolution	A new data point is available every 15 minutes.
Spatial coverage	MSG disk, which is the visible part of earth from the view point of a geostationary satellite at 0° longitude.
Temporal coverage	2004-01-19 - 2015-12-31. However, due to storage limitations, this is limited to 2015-01-01 - 2015-12-31 for this research.

Table 5: Characteristics of the CLAAS-2 data set.



(a)



(b)

Figure 9: An example of a could mask. This cloud mask is based on satellite observations on 1st of Januari 2015, at 00:00. (a) shows the cloud mask over a map of earth. (b) shows how the selected sub cloud masks.

4.4 Integration

In this section, the experiments are described that will determine the best combination of elements for the novel network. First, the experiment that will determine the configuration of the encoder and decoder is discussed. Then, the experiment for the integration of the encoder, forecaster and decoder is explained.

4.4.1 Encoder & decoder

In section 4.2 a single suitable model was selected for the encoder and decoder, namely the SegNet model. The SegNet model consists of an encoder and a decoder, which are coupled to form a complete network. The original goal of the SegNet model is to segment the pixels of an image into one of several classes. The usage of the SegNet model in this research is quite different. That is to say; the encoder is used to efficiently downsample an input image into an encoding, which is fed to a forecaster. The result of the forecaster is upsampled by the decoder, to reconstruct the forecast. This change in the goal of the model does not require many adjustments to the architecture of the network. Only one implementation detail regarding the upsample method in the decoder has to be adjusted. Because many different upsample methods exist (see section 2.7.4), an analysis of those methods integrated into the decoder is warranted, to determine which upsample method will be used in the decoder. The experiment that forms the basis of this analysis is explained in this section.

First, the adjustment to the SegNet model and the resulting neural network architectures is explained. Secondly, the methodology for training the networks is described. Lastly, the methodology of the analysis, which determines which upsample method will be used in the decoder is elaborated.

Architecture The architecture of the SegNet model consists of an encoder based on the VGG-16 network [43] and a corresponding decoder. The encoder consists of multiple convolutional stages, which in itself consists of multiple convolutional layers wrapped by a ReLU activation function, followed by a max-pool layer. The decoder is very similar; an unpool layer is followed by multiple convolutional layers wrapped by a ReLU activation function. As explained in section 2.7.3, the unpool operation uses switch variables, produces during a corresponding max-pool operation to compute the inverse of the max-pool. However, this goes against the requirement of the encoder and decoder to be independent. How the switch variables affect the reconstruction of an encoding is briefly investigated, by implementing the SegNet model, sans the softmax layer at the end that is used to classify a pixel in a class. This model is trained in a similar method as explained in the next paragraph. Then, two cloud masks are encoded and reconstructed, using the same switch variables. The results can be found in figure 10. The reconstructed cloud masks are very accurate. However, there is almost no difference between the reconstructed images, while the input images are very different. This is due to how much information (3,9 MB) is available

in the switch variables compared to the encoding (0,25 MB), which the model will learn to use optimally. Therefore, the SegNet architecture is adjusted by incorporating the upsample methods as mentioned in section 2.7.4:

- Transposed convolution
- Decomposed transposed convolution
- Bilinear upsampling & convolution
- Bilinear upsampling & separable convolution
- Bilinear additive upsampling

This means that there will be five models for this experiment. The difference between these models compared to the original SegNet model is: the replacement of the unpool layer in the decoder with one of the mentioned upsample method and the removal of the softmax layer at the end of the SegNet model that is used for classifying the pixels. The generic structure of these models can be found in figure 11. Each model has the same encoder but differs in the decoder. The detailed architecture of the encoder and all the different decoders can be found in appendix B.

Training As mentioned before, the goal of the encoder is to efficiently encode an input cloud mask. The goal of the decoder is the reconstruct an encoded forecast as accurately as possible. However, to minimise the number of variables that can impact the performance of the decoder, no forecaster will be used in this experiment. The encoder and decoder are coupled directly, as is the case for the original SegNet model.

Each of the five different decoders has an associated instance of the encoder. While the architecture of the instances of the encoders do not differ, they might adapt to its specific decoder, making them incompatible with other decoders. The following networks are trained:

$$\text{Encoder-Decoder}_{\text{BilinearAdditiveUpsampling}}$$
$$\text{Encoder-Decoder}_{\text{BilinearUpsamplingConvolution}}$$
$$\text{Encoder-Decoder}_{\text{BilinearUpsamplingSeparableConvolution}}$$
$$\text{Encoder-Decoder}_{\text{DecomposedTransposedConvolution}}$$
$$\text{Encoder-Decoder}_{\text{TransposedConvolution}}$$

The encoder and decoder are trained as one network. This is achieved by feeding a cloud mask to the encoder, which encodes it into an encoding. The encoding is fed to the decoder, which tries to reconstruct the encoding as accurately as possible. The difference between the reconstructed cloud mask and the input cloud mask is measured by the MSE loss. This loss is backpropagated

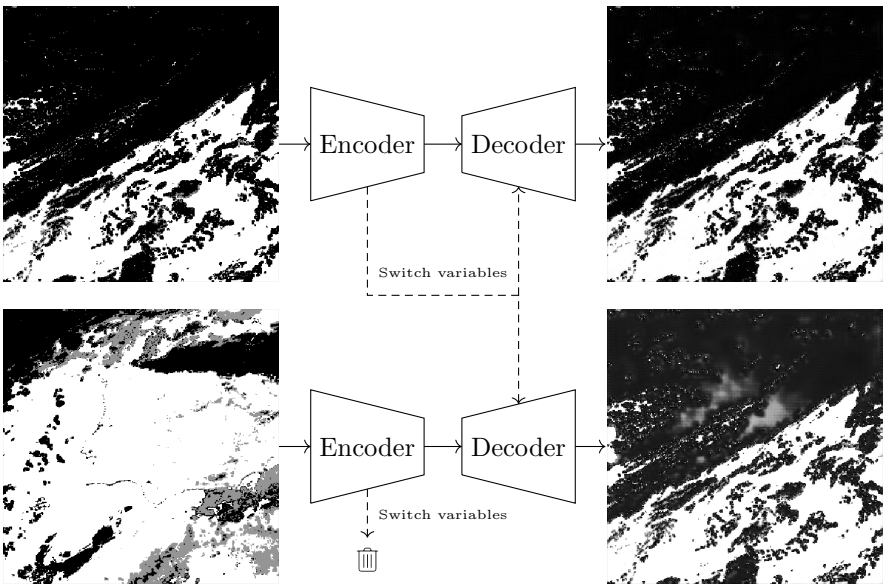


Figure 10: An illustration of the issue with upsampling using unpooling. The two cloud masks are encoded using the same network. Both encoding are decoded using the same switch variables produced during the encoding of the first image. The resulting reconstructed images are very accurate, however, very similar. When this upsample method would be incorporated in the encoder-forecaster-decoder framework, the forecaster would be bypassed by the switch variables, loosing the forecasting ability of the model.

through the decoder and encoder to update their weights using the Adam optimiser [44] with learning rate $0,0005$. This process is repeated for a train data set of 100 000 cloud masks selected randomly from all available cloud masks. Each of the five models is trained using the same train data set, again to minimise the variability between the models. After training, the performance of the model is analysed, as will be explained in the next paragraph.

Analysis The analysis consists of comparing the performance of the different variants of encoder-decoder on a test data set, consisting of 13 888 randomly selected cloud masks, which did not appear in the train data set. The performance metrics that are used to capture the performance of the models are the average MSE and average SSIM. The average MSE is determined by:

$$\frac{1}{N}\sum_{n}^{N} MSE\left(I_n, \hat{I}_n\right) \tag{13}$$

where N is the test data set size, and I_n, $\hat{I}_n$ are the cloud mask and reconstructed cloud mask at index n from the test data set.

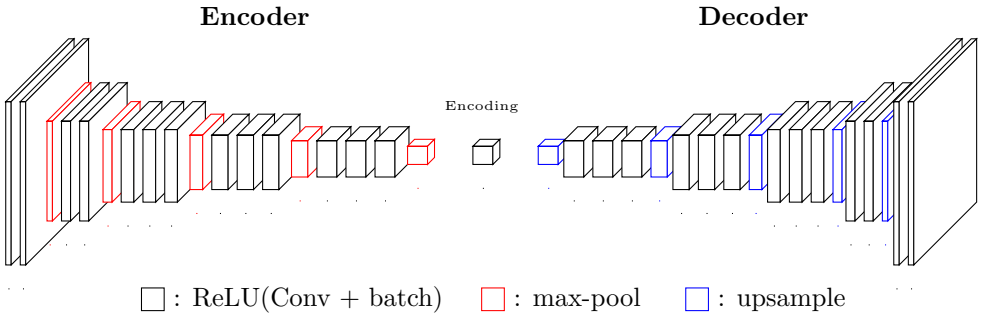


Figure 11: An overview of the architecture for the encoder and decoder, based on the SegNet model. In the SegNet model, the upsample layer is implemented by the unpool operation. In this experiment, five other upsample methods are used.

The average SSIM is determined by:

$$\frac{1}{N}\sum_{n}^{N} SSIM\left(I_n, \hat{I}_n\right) \tag{14}$$

Both the average MSE, as well as the average SSIM, will be used to determine the upsample method to be used in the decoder for the rest of the research.

4.4.2 Forecaster

In section 4.2 two suitable building blocks were selected for the forecaster, namely ConvLSTM and TrajGRU. The goal of the forecaster aligns with the goal of the two ConvRNN blocks, that is to say, finding patterns in a spatiotemporal sequence. Therefore, the two blocks can be integrated into the framework without many adjustments.

In this section, an experiment is described that is used to compare the performance of the two ConvRNNs when they are integrated into the encoder-forecaster-decoder framework. The best performing model will be used in the rest of the research.

First, the resulting architecture of integrating the two building blocks in the framework is explained. Secondly, it is explained how these models are trained. And finally, the methodology of the analysis is elaborated.

Architecture This paragraph explains how the ConvLSTM and TrajGRU layers are implemented, given the concept as explained in the related work, and how they are adjusted to fit the proposed framework. First ConvLSTM is explained, then TrajGRU.

ConvLSTM The proposed adjustment to the LSTM architecture as explained in equation 10 is implemented by using convolutional layers that contain

the weights and biases as required. The layers perform the required convolutional operations. The following pseudocode shows how this is achieved:

Algorithm 1: ConvLSTM algorithm.

ConvLSTM(x_t, h_{t-1}, c_{t-1}):
$i_t = \text{sigmoid}(Wxi(x_t) + Whi(h_{t-1}) + c_{t-1} * Wci)$
$f_t = \text{sigmoid}(Wxf(x_t) + Whf(h_{t-1}) + c_{t-1} * Wcf)$
$c_t = f_t * c_{t-1} + i_t * \tanh(Wxc(x_t) + Whc(h_{t-1}))$
$o_t = \text{sigmoid}(Wxo(x_t) + Who(h_{t-1}) + c_t * Wco)$
$h_t = o_t * \tanh(c_t)$
return h_t, c_t

where Wxi, Whi, Wxf, Whf, Wxc, Whc, Who, Wxo are convolutional layers, and Wci, Wcf, Wco are tensors containing weights. The hyperparameters for these convolutional layers are set the same as in the original research, which investigated different settings to determine the most optimal. For the ConvLSTM to fit the framework, the number of channels in the convolutional layers must match the number of channels in the encoding, which is 512. The details of the convolutional layers can be found in appendix B.3.

TrajGRU The TrajGRU equations from 12 are implemented in a similar manner as ConvLSTM, where the convolutional operations are implemented using convolutional layers, resulting in the following pseudocode:

Algorithm 2: TrajGRU algorithm

TrajGRU(x_t, h_{t-1}):
u_t, v_t = flow_generator(x_t, h_{t-1})
z_t=sigmoid($Wxz(x_t)$+$Whz(\text{warp}(h_{t-1}, u_t, v_t))$)
r_t=sigmoid($Wxr(x_t)$+$Whr(\text{warp}(h_{t-1}, u_t, v_t))$)
h'_t=sigmoid($Wxh(x_t)$+$Whh(\text{warp}(h_{t-1}, u_t, v_t))$)
h_t=(1-z_t)*h'_t+z_t*h_{t-1}
return h_t

where Wxz, Whz, Wxr, Whr, Wxh, Whh are convolutional layers, and u_t, v_t, are the flow field generated by the flow generator. The hyperparameters of the convolutional layer are set the same as the original research. For the TrajGRU to fit the framework, the input channels of the convolutional layers have to be set to the number of channels in the encoding, namely 512. The details can be found in appendix B.3.

The flow generator is a sub network that determines the motion flow from the current input and the previous hidden state. The same flow generator architecture is used as in the original research: a one hidden layer convolutional network that takes the concatenation of x_t and h_{t-1} as its input. This results in the following pseudocode:

Algorithm 3: Flow generator algorithm.

flow_generator(x_t, h_{t-1}):
    input2flow = tanh(i2f_conv(x_t))
    hidden2flow = tanh(h2f_conv(h_{t-1}))
    u = u_conv(i2f + h2f))
    v = v_conv(i2f + h2f)
    return u, v

where i2f_conv, i2h_conv, u_conv, v_conv are convolutional layers, whose details can be found in appendix B.3.

The obtained flow are used to warp the previous hidden state. The warp function is given by the following pseudocode:

Algorithm 4: Warp algorithm

warp(x, u, v):
    # mesh_grid returns coordinate matrices from coordinate vectors.
    xx, yy = mesh_grid([0, ..., x.width], [0, ..., x.height])
    u_grid = xx + u
    v_grid = yy + v
    # scale grids to [-1,1]
    u_grid = 2 * u_grid / max(x.width - 1, 1) - 1
    v_grid = 2 * v_grid / max(x.height - 1, 1) - 1

    # For each output location [i, j], the values from u_grid[i] v_grid[j] specify a pixel location in input x, which is obtained using bilinear interpolation.
    # The sampling grid specifies the sampling pixel locations normalised by the input spatial dimensions. Therefore, the values of u/v grid should range between [-1, 1].
    output = grid_sample(x, u_grid, v_grid)
    return output

Training The goal of the integrated models is to compute a forecast from a number of input cloud masks. Therefore, ConvLSTM and TrajGRU are integrated with the encoder and decoder, as obtained by the experiment in section 4.4.1, to form a complete model. To minimise the variability between the two models, the parameters of the encoder and decoder are fixed, such that only the parameters of the forecaster are adjusted during training. The parameters of the encoder and decoder are set by the parameters obtained during the training of the encoder and decoder. Otherwise the encoding would not be meaningful. Thus the following composed models are considered for this experiment:

$$\text{Encoder}_{\text{fixed}} \rightarrow \text{ConvLSTM} \rightarrow \text{Decoder}_{\text{fixed}}$$
$$\text{Encoder}_{\text{fixed}} \rightarrow \text{TrajGRU} \rightarrow \text{Decoder}_{\text{fixed}}$$

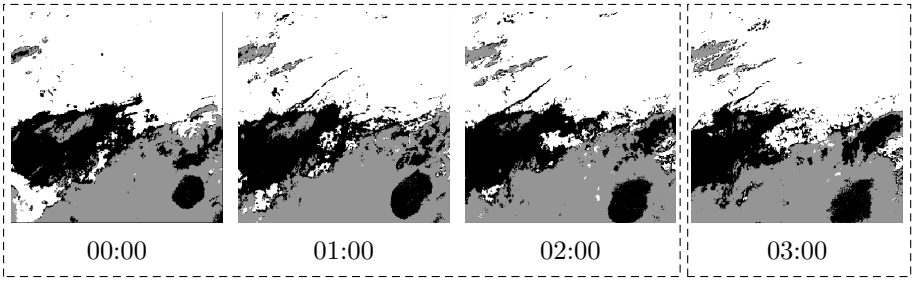


Figure 12: An example of an unencoded input series used during training of the models. The cloud masks are from the first of June 2015. The fourth image is the ground truth for the forecast.

To train the model, it is fed a sequence of three cloud masks as input. The cloud masks are each one hour appart. These settings are chosen as a middle ground between the amount of information available for the forecaster, and the time of training the models. Other settings are considered during optimisation. Each cloud mask is encoded by the encoder. The sequence of encodings is fed to the forecaster, where the initial state is set to 0:

$h_{t-1} = 0$
for encoding **in** encodings:
    h_{t-1} = forecaster(encoding, h_{t-1})
return h_{t-1}

The last state h_{t-1} after iterating through the encodings is considered as the encoded forecast. This encoded forecast is fed to the decoder, which decodes it to a reconstructed forecast. The loss of the reconstructed forecast compared to the ground truth is computed using the MSE loss. The loss is backpropagated through the model to update the weights of the forecaster using the Adam optimiser with a learning rate of $0,0005$. This process is repeated for a train data set of 100 000 cloud mask sequences. The sequences are not sequential. An example of such a sequence can be found in figure 12. After training, the performance of the models is analysed, as will be explained in the next paragraph.

Analysis The analysis consists of comparing the performance of the trained models on a test data set, consisting of 13 888 sequences of cloud masks with the same properties as the train sequences, which did not appear in the train data set. The metrics that are used to capture the performance of the models are the average MSE and average SSIM.

Both the average MSE, as well as the average SSIM, will be used to determine the integrated model that is considered to be further optimised.

4.5 Optimisation

Once the best performing model has been determined, it is optimised by adjusting its hyperparameters. At this point, the second sub research question is also considered, that is to say: *how many images as the input sequence is optimal?*

The approach for finding the optimal configuration is to take the basic network configuration that was used in the previous experiment and apply grid-search for finding the optimal combination of hyperparameters. Because most of the hyperparameters in the models have already been optimised by previous research, the focus will be on the hyperparameters that are relevant to the proposed framework. For example, the kernel size for any of the convolutional layers is not considered, as this has already been established. The hyperparameters that will be considered are the number of stacked RNN layers in the forecaster and the number of input images for the model.

First, the two hyperparameters that are considered are briefly explained, as well as the adjustments to the architecture. Second, the manner of training and analysis of the proposed models are discussed.

4.5.1 Architecture

In this section, the architecture of the networks that will be tested for finding the optimal hyperparameters configuration is discussed. The network as obtained by the previous experiment will be regarded as the baseline. First, the number of layers and its impact is discussed. And second, the number of input images is discussed.

Stacking layers As described in the background (2.7.5), by stacking multiple RNN elements a more complex network can be constructed, which is expected to increase performance [45]. The ConvRNN used as the forecaster is the element that will be stacked. The element itself would not need any adjusting, only the forward pass trough the forecaster has to be adjusted to the number of layers, as shown in figure 5. The following pseudocode shows the implementation of the adjusted forward pass:

Algorithm 5: Forward pass through a stacked RNN.

```
StackedRNN(x_t, stacked_h_{t-1}):
    input = x_t
    for i in [0, ..., l]:
        output = ConvRNN_i(input, stacked_h_{t-1}[i])
        stacked_h_t[i] = output
        input = output
    return output, stacked_h_t
```

Where l is the number of layers, and ConvRNN_i is the ConvRNN at layer i.

The number of layers will range from one to five during grid search. The hyperparameters of the ConvRNN at each layer will not differ from the baseline, as most of the hyperparameters are already optimised, as mentioned before.

Input As described in the background (2.7.5), the loss of the forecast is backpropagated through the network for the number of inputs. By increasing the input size, the backpropagation can run further back, which could increase performance. However, this also increases computation time. The baseline configuration has an input size of three. An input size from one up to five will be tested during grid search. The models that will receive only one input image would not be able to learn temporal correlations, while there is no temporal information. These models would serve as a baseline for the other configurations.

4.5.2 Training

The proposed configurations will be trained analogous to the method as the previous experiment, as described in section 4.4.2. However, the parameters of the encoder and decoder will no longer be fixed. The expectation is that the encoder-decoder will adapt to the forecaster, resulting in a more accurate forecast. The configuration of the 25 models are as follows:

	Layers				
Input	(1,1)	(2,1)	(3,1)	(4,1)	(5,1)
	(1,2)	(2,2)	(3,2)	(4,2)	(5,2)
	(1,3)	(2,3)	(3,3)	(4,3)	(5,3)
	(1,4)	(2,4)	(3,4)	(4,4)	(5,4)
	(1,5)	(2,5)	(3,5)	(4,5)	(5,5)

4.5.3 Analysis

Once the networks have been trained, their performance is measured on a validation data set using the average MSE. However, next to the performance metrics to compare the performance of the different configurations between each other, the performance is also measured against an intrinsic encoded and reconstructed persistence forecast. This encoded persistence forecast is obtained by encoding the last input image and reconstructing it again using the decoder. By comparing this encoded persistence forecast with the forecast made by the complete network, the forecastability of the forecaster can be better analysed. Because the encoded persistence forecast has gone through the same encoding and decoding, it has undergone the same loss of information and accuracy as the forecast made by the whole network. It can, therefore, be used as a baseline to measure the performance of the forecaster. The performance of the forecaster will be compared against the encoded persistence forecast by using an MSE based skill score:

$$\text{skill} = \frac{MSE_{forecaster}}{MSE_{encoded\ persistence\ forecast}} \tag{15}$$

An illustration of obtaining the encoded persistence forecast can be found in figure 13.

The optimal hyperparameter configuration will be chosen based on the average MSE loss and skill. The performance of this configuration is further analysed in the next section.

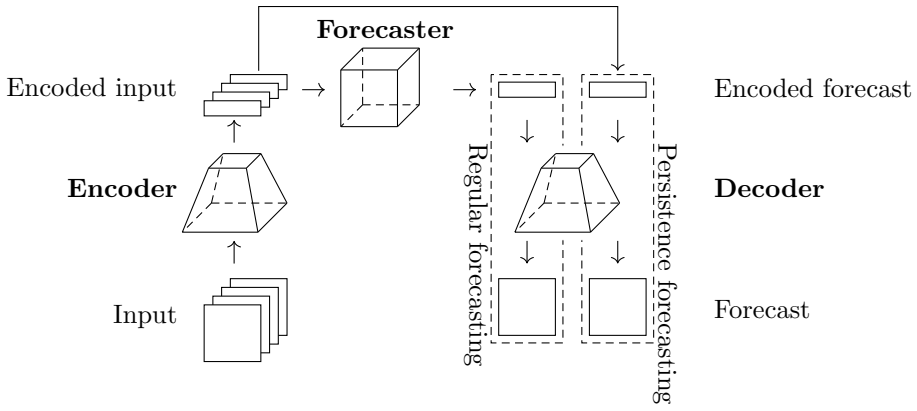


Figure 13: An illustration of the process for obtaining the persistence forecast.

4.6 Evaluation

In this section, the evaluation method for the novel model as determined by the previous experiment is elaborated. The performance of this model is compared against four other models, all discussed in the related work. First, as a baseline, persistence forecasting is used. This method should not be confused with the encoded persistence forecast, as proposed in the previous section. Secondly, as the optical flow method, the state-of-the-art algorithm TV-L_1 is used. And lastly, the models based on ConvLSTM and TrajGRU are considered as the state-of-the-art deep learning models. To distinguish between the ConvLSTM and TrajGRU building blocks as used in the forecaster, the model based on the encoder-forecaster-decoder framework will be denoted as the novel model, and the state-of-the-art models from the related work based on ConvLSTM and TrajGRU will be denoted as the ConvLSTM evaluation model and the TrajGRU evaluation model.

At this point, the third sub-question is also considered: *what are the possible forecasting horizons?* For the novel model, as well as the ConvLSTM and TrajGRU evaluation models, to be able to make a forecast for different horizons, the models need to be trained afresh on a train data set adjusted to the new forecast horizon.

First, the details of the implementation of the four comparison models are discussed. Secondly, the methodology for different forecast horizons is elaborated. Lastly, the methodology for the analysis of the performance of the models, as well as the comparison, is elaborated.

4.6.1 Models

Persistence The persistence forecast model is easily implemented, it returns the last input image as a forecast.

TV-L_1 The implementation of the TV-L_1 method is done using the OpenCV library, as proposed by Urbich et al. [29]. The parameters of the TV-L_1 algorithm are set similar to the previously mentioned research.

The cloud mask forecast is determined by computing the motion vectors between the last two input cloud masks using the TV-L_1 algorithm and applying the obtained motion vectors to the latter input image. This approach does not need to be adapted for cloud mask forecasting, while the TV-L_1 algorithm is able to compute the motion vectors on cloud masks as well.

This approach is applicable if the time between the input images is the same as the forecast horizon. Otherwise, the process can be iterated, by using the obtained forecast as a new input cloud mask, to obtain the forecast at the required forecast horizon.

ConvLSTM & TrajGRU evalution models ConvLSTM and TrajGRU are both used as building blocks for complex networks (3.3, 3.4). However, as described by Shi et al. [32], the proposed framework from the ConvLSTM research is subpar to the framework proposed in the TrajGRU research. Therefore, both the ConvLSTM and the TrajGRU building blocks are implemented in a model based on the framework as proposed in the TrajGRU research (figure 7). The details of the implementation of these models can be found in the appendix B.4. These models are trained analogous to the training method of the novel model.

4.6.2 Forecast horizon

The forecast horizon is measured from the last input image to the ground truth. From the requirements of the Solar Team, a forecast horizon up to 6 hours is desirable. However, given the limitations of cloudage forecasting using satellite imagery, as described in section 2.2, forecast horizons up to three hours are considered.

The machine learning-based models, the novel model and the ConvLSTM and TrajGRU evaluation models, have to be trained for each forecast horizon specifically. The adjustments that have to be made to achieve a desired forecast horizon are only to the training and test data set. When the ground-truth cloud mask is set at the specific forecast horizon, the model will learn to forecast for that horizon. The time steps between the input images will be kept at one hour when possible.

The forecast horizons that will be considered are 30 minutes, one hour, two hours, and three hours. An overview of the timestamps of the input images, as well as the ground truth, can be found in table 6.

Forecast horizon	Input time stamps			Ground truth time stamp
30 minutes	t_0	t_{30min}	t_{60min}	t_{90min}
1 hour	t_0	t_{1h}	t_{2h}	t_{3h}
2 hours	t_0	t_{1h}	t_{2h}	t_{4h}
3 hours	t_0	t_{1h}	t_{2h}	t_{5h}

Table 6: An overview of the timestamps for the input images, as well as the ground truth, for the different forecast horizons. This assumes an input size of three, while this might not be the most optimal configuration that will be determined by optimising the hyperparameters.

4.6.3 Analysis

The analysis of the performance of the models on the different forecast horizons is done using the average MSE on a test data set. Furthermore, the MSE is used to determine cloud masks which the models perform well on, as well as cloud masks that the models perform poorly on. These cloud masks are analysed to find matching cloudage, to determine which type of cloudage is difficult for the models to forecast, as well as the type of cloudage that is easy to forecast.

5 Results

In this section, the results of the experiments, as described in the methodology, are discussed. The experiments were implemented using python and the PyTorch machine learning framework[1], and ran on the high-performance cluster of the University of Twente.

5.1 Integration

5.1.1 Encoder & decoder

The experiment for the encoder and decoders is designed to find the most optimal upsample method in the decoder. Five different encoder-decoder pairs were trained, and their performance on a test data set is analysed in this section. The performance of the models can be found in table 7. Examples of the reconstructed cloud masks can be found in the appendices in figure 17.

The performance of the upsampling methods is quite close, except for bilinear additive upsampling. This is somewhat surprising, while the performance of bilinear additive upsampling in other research [16] was up to par, or even better, compared to the other methods, for a wide variety of problems. When looking into the loss during training, as well as intermediate examples of cloud mask reconstruction, no anomalies were found which could be the cause for this lack in performance of the bilinear additive upsampling method.

Of all the methods, transposed convolution performs the best. It performs slightly worse in regards to the average SSIM compared to bilinear upsampling & separable convolution, but this is negligibly small. However, a visual inspection of the reconstructed cloud masks shows the typical chequerboard artefacts associated with deconvolution [46]. Furthermore, while the decomposed transposed convolution performs similarly as the regular transposed convolution in regards to the average MSE, it shows these artefacts more clearly. Consequently, it suffers in regards to the average SSIM, showing the advantage of using SSIM in addition to MSE.

In the end, the transposed convolution is chosen as the upsample method to be used in the decoder, as the reconstruction artefacts are deemed tolerable compared to the gained performance.

[1] repository: https://github.com/appeljus/encoder-forecaster-decoder

Upsample method	Average MSE	Average SSIM
Bilinear additive upsampling	0,683	0,276
Bilinear upsampling & convolution	0,397	0,307
Bilinear upsampling & separable convolution	0,334	0,334
Decomposed transposed convolution	0,336	0,283
Transposed convolution	0,315	0,333

Table 7: The performance measured by the average MSE and the average SSIM of the models consisting of an encoder and decoder with the specified upsampling method on the test data set. A lower average MSE indicates good performance. A lower SSIM indicates poor performance.

5.1.2 Forecaster

The experiment for the forecaster is designed to find the best performing element to be used as the forecaster: ConvLSTM or TrajGRU. These building blocks were integrated with the encoder and decoder as determined by the previous experiment into the encoder-forecaster-decoder framework as complete models. These models were trained and tested. The performance of the models on the test set can be found in table 8. Examples of the reconstructed forecast can be found in the appendix in figure 18.

The performance of the models is significantly worse in regards to the performance metric, as well as a (subjective) visual inspection of the reconstructed cloud masks, compared to the previous experiment. While the encoder-forecaster-decoder models will inherently never be able to perform better than the encoder-decoder models, three times as worse for the average MSE is quite poor. This is probably caused by fixing the parameters of the encoder and decoder. The parameters of the encoder and decoder are set by the parameters obtained during the previous experiment. However, the encoder and decoder were directly coupled in that context. While in this experiment, a forecaster is placed in between. During training, the forecaster will adjust to the encoder and decoder. In contrast, the reverse is not the case: the encoder and decoder will not adjust to the forecaster. This is probably the cause for the lack of performance, as the performance during optimisation, when the parameters for the encoder and decoder were not fixed, is a lot better, as can be seen in the next section.

When inspecting examples of the reconstructed forecast, the chequerboard pattern is even more clear for this experiment. It clearly shows an eleven by eleven pattern, which is due to the size of the encoding when processed by the forecaster. However, as with the performance, this disappears when the parameters of the encoder and decoder are not fixed.

Although there are some remarks regarding the performance of the integrated models, the TrajGRU element is chosen to be used as the forecaster in the subsequent experiments. Even though its performance is poor, it is still better than the performance of the ConvLSTM element.

Forecaster	Average MSE	Average SSIM
ConvLSTM	1,397	0,295
TrajGRU	0,923	0,285

Table 8: The performance of the models consisting of the encoder, the specified forecaster, and the decoder on the test data set measured by the average MSE and average SSIM.

5.2 Optimisation

The experiment for the optimisation of the hyperparameters used an exhaustive grid search on the specified range of the number of input images and the number of layers of the TrajGRU element in the forecaster. In table 9, the performance of the 25 different models measured by the average MSE on a validation set is given. In table 10, the skill of the models compared to its intrinsic encoded persistence forecast is given. Examples of the reconstructed forecast can be found in the appendix in figure 19.

The first observation given results is that more layers result in a worse forecast, even though the opposite was expected. The same can be said for the number of input images: increasing the number of input images decreases the forecast performance. This indicates that the models are not learning to forecast but to reconstruct the last input image as accurately as possible. Especially the number of input images is an indication of this behaviour: by grouping the models by the number of input images, the models with one input images perform on average the best. It is worrying that models with no temporal information (single input image) perform the best when developing a model that should find patterns over time.

The encoded persistence forecast was constructed to form a baseline which should indicate the forecasting ability of the models by using a skill score between the model and the baseline. However, when investigating the loss of the encoded persistence forecast during the training of the models, it seems that it would not be a reliable baseline. When the parameters of the encoder and decoder are adjusted during training, the performance of the encoded persistence forecast changes as well. The expectation was that the encoder and decoder would remain in sync, meaning that an encoding by the encoder can be accurately reconstructed by the decoder, resulting in a stable performance of the encoded persistence forecast. However, this is not the case as can bee seen in figure 14, where the loss of the encoded persistence forecast during training is plotted. It shows erratic behaviour, meaning that the adjustments to the parameters of the encoder and decoder during training result in a highly variable persistence encoding forecast performance. Therefore, the skill between the encoded persistence forecast and the model will not be used as a performance measure to determine the optimal configuration.

Although the models do not show the ability to forecast, an optimal configuration is still chosen to be further evaluated. While the model with three

layers and five input image has the lowest average MSE, this configuration is not chosen, as it has been concluded that in general increasing the number of layers reduces the performance of the model. In conclusion, the configuration that is further evaluated has one layer and four input images.

		Layers				
		1	2	3	4	5
Input	1	0,424	0,428	0,563	0,507	0,699
	2	0,412	0,586	0,469	0,400	0,820
	3	0,564	0,576	0,489	0,514	0,676
	4	0,375	0,541	0,479	0,540	0,847
	5	0,400	0,648	0,369	1,062	0,842

Table 9: The performance of the encoder-forecaster-decoder model using the specified number of layers in the forecaster and number of input images on a validation set measured by the average MSE. The colors indicate the performance, green means top performance, yellow means reasonable performance, and red mean poor performance.

		Layers				
		1	2	3	4	5
Input	1	0,342	0,119	0,179	0,240	-0,409
	2	0,100	0,117	0,405	0,647	-0,001
	3	0,123	-0,159	0,637	0,546	-0,004
	4	0,096	0,495	0,507	0,524	0,009
	5	0,161	-0,301	0,507	0,071	-0,022

Table 10: The performance of the encoder-forecaster-decoder model using the specified number of layers in the forecaster and number of input images on a validation set measured by the skill compared to their encoded persistence forecast. The colours indicate performance. The colour scale is independent of the scale used in table 9. Green means top performance, yellow means reasonable performance, and red mean poor performance. In the end, the skill was not used as a performance measure to determine the optimal configuration.

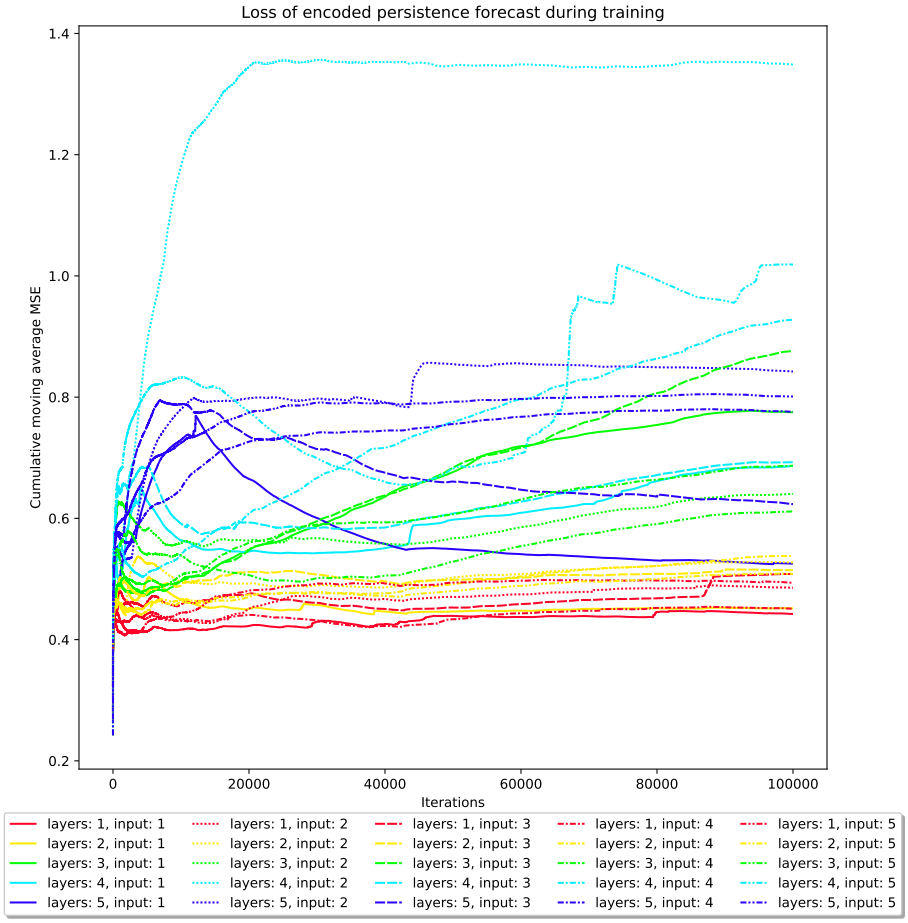


Figure 14: The loss of the encoded persistence forecast during training. The loss is measured by the cumulative moving average (CMA) MSE. The CMA at iteration n is determined by averaging the loss of all preceding iterations: $\frac{MSE_1+...+MSE_n}{n}$. The CMA is used to show the development of the performance during training, instead of the performance of individual iterations. The plot is best viewed in colour.

5.3 Evaluation

The evaluation of the chosen configuration is done by comparing its performance for different forecast horizons with the performance of a baseline method and three state-of-the-art models. In table 11, the performance of the five different models on the four forecast horizons is given. In the appendices in figure 20 examples of the forecasts can be found. The analysis of the performance on the different types of cloud masks can be found in section 5.5.

What stands out first is the poor performance of the optical flow method TV-L_1 for forecast horizons after 1 hour. This is probably due to compounding errors. The optical flow method finds motion vectors between two images and uses the motion vectors to make a forecast. The forecast for one hour is used to compute the forecast for two hours, and the forecast for two hours for the forecast for three hours. This approach was chosen as it was used in research similar to cloudage forecasting. However, it seems the implementation is not applicable for our case.

Secondly, the conclusion by Shi et al. [32] that TrajGRU outperforms other ConvRNN models is once again confirmed. It shows an improvement over ConvLSTM on all forecast horizons, as well as the proposed novel model.

Third, the performance of the novel model is not as expected. Unlike the other models, the performance increases for longer forecast horizons, until the forecast horizon of three hours. This could indicate some anomaly during the training of the models. To investigate this atypical behaviour, an additional experiment was conducted. This experiment trains and tests the novel model ten distinct times. This experiment is further explained in the next section.

		Model				
		Persistence	**TV-L_1**	**ConvLSTM**	**TrajGRU**	**Novel model**
Forecast horizon	**30 min**	0,473	0,381	0,296	0,234	0,512
	1 h	0,563	0,480	0,373	0,321	0,479
	2 h	0,594	1,340	0,383	0,322	0,455
	3 h	0,621	1,630	0,405	0,368	0,571

Table 11: The performance of the baseline, three state-of-the-art, and the novel model on the test set measured by the average MSE.

5.4 Repeating runs

To see whether the atypical behaviour of the novel model is due to an anomaly during training, the configuration of the novel model for the 1-hour forecast horizon is taken and trained and tested ten times. For each iteration, the model is reinitialised to ensure that there is no connection between the runs. By repeating the training and testing, it can be determined whether the random initialisation of the model before training has any impact on the performance of the model during testing. As can be seen in table 12, there is a large difference

in the performance of the ten iterations on the test set. However, this cannot be traced back to the training method, as the performance during training is very similar. What could be the cause for similar performance during training, but deviating results during testing has not been uncovered.

	Average train MSE	**Average test MSE**
1.	0,384	0,347
2.	0,384	0,369
3.	0,385	0,385
4.	0,387	0,447
5.	0,388	0,457
6.	0,384	0,472
7.	0,386	0,407
8.	0,387	0,419
9.	0,386	0,381
10.	0,387	0,422

Table 12: Results of repeating the training and testing with the same configuration of the novel model ten times. The average train MSE shows the performance of the models during training. The average test MSE shows the performance of the models on an unseen test set.

5.5 Cloud specific evaluation

From the performance of the models on different types of cloudage, a few remarks can be made.

First, all the models perform equivalently on the cloud masks. That is to say, an easy cloud mask is easy for all models, and a difficult cloud mask is difficult for all models.

Secondly, cloud mask containing almost no cloudage are very easy to forecast, as can be expected. However, almost completely clouded cloud masks are forecasted with an average performance.

The most difficult cloud masks can be divided into two categories: cloud masks with multiple cloud layers (figure 15), and cloud masks containing multiple small clouds (figure 16). For the former type of cloud masks, the models are bad at distinguishing between the layers, and forecast the cloud layers as one blob, based on the previous motion of one of the layers. For the latter type of cloud masks, the forecasts blur between cloud and cloud-free. Due to this blurriness, the error on both cloud and cloud-free rises, resulting in bad performance.

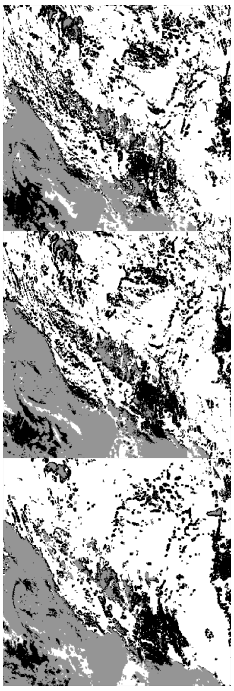


Figure 15: An example of a difficult to forecast cloud mask containing multiple layers of clouds. While it is difficult to see 3D layers in a black and white 2D image, the black blob in the lower middle stays mostly in the same place, while the grey blob moves across it.

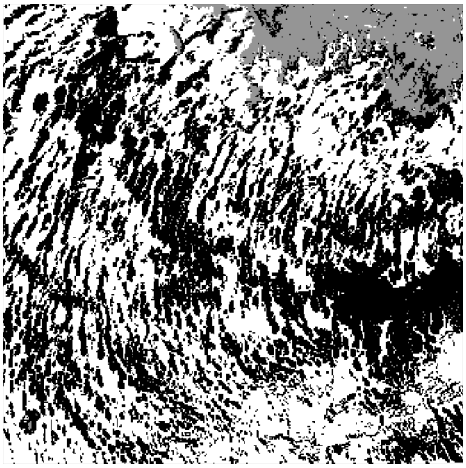


Figure 16: An example of a difficult to forecast cloud mask containing a lot of small clouds.

6 Discussion

In this section, the results as presented in the previous section, the design of the encoder-forecaster-decoder model, and the limitations of the proposed methodology are discussed. In the end, a recommendation for the Solar Team is made.

6.1 Results

Given the results, several remarks can be made. First, the encoded persistence forecast is not a reliable baseline for the forecasting abilities of the network. The encoded persistence forecast will change due to the learning of the encoder-decoder. Especially because it seems that the encoder-forecaster-decoder model tries to reconstruct the input image, instead of making a forecast, the encoder and decoder adapt to the forecaster, resulting in inconsistent encoded persistence forecasts. The adapting of the encoder and decoder could be a constraint, by adding the loss of the encoded persistence forecast to the backpropagation algorithm. That way, the encoding would be regularised, similarly to variational autoencoders.

Secondly, the performance of the deep learning state-of-the-art models on cloudage forecasting is excellent. While the comparison to the optical flow method cannot be adequately made, as the performance of the TV-L_1 model is substandard, the comparison to the baseline model shows that the STSF models perform similarly on cloudage forecasting as on other STSF problems. The performance of the novel model is discussed in the next section.

6.1.1 Encoder-forecaster-decoder framework

As discussed, the encoder-forecaster-decoder does not seem to have forecasting abilities. There are a few characteristics that could contribute to this. First, compared to the TrajGRU evaluation model, the encoder-forecaster-decoder does not have connections between the encoder and decoder. Not having connections between the encoder and decoder was an explicit design choice, because it would interfere with the forecasting. However, as has been shown by the TrajGRU evaluation model, it could be done in such a manner that it enhances the forecasting. Although the type of connection and information should be carefully selected, as has been shown by the impact of the switch variables (figure 3).

Secondly, the forecaster is tasked to find patterns over time. These patterns are in the spatial dimension. However, because the encoder heavily downsamples the spatial dimensions, there is not much information left to find these patterns. The encodings do have many channels. However, the forecaster is not specifically designed to employ the channels to find patterns, which might leave a lot on the table in terms of finding those patterns.

6.2 Methodology

The proposed methodology has many limitations. First, the used data type is not practical for determining SSI. The used cloud masks only have three possible values. There are data sets available that contain cloud masks with more detail, also known as cloud optical thickness. However, because the optical thickness is obtained using the visible light channels on the satellite's sensor, they are not available during night time, limiting the available data points. Therefore, the choice was made to use the cloud masks, as described. While the resulting SSI using cloud mask would be worse, it does show the ability of the models to forecast cloudage. Both cloud masks and optical thickness based models are in principle trying to forecast the underlying motion of clouds.

Secondly, the methodology of training the models can be improved. Currently, the train data set is large enough that a single pass is sufficient for the models to achieve satisfactory performance. However, it is common to use a smaller train data set and multiple passes (or epochs) over the data set. While either approach is valid, by using epochs, the training of the models can be better tuned and monitored. In the used methodology, the models were trained for a specific number of iterations. However, a stopping criterion is usually used to determine when the models have achieved satisfactory performance during training. By using a stopping criterion, over- or underfitting can be prevented. Using epochs and a stopping criterion could both aid in discovering the anomalies as discussed in section 5.4 of the results.

Third, the novel model is compared against three state-of-the-art models, which is a good indication of its relative performance. Nonetheless, perhaps the most crucial model is missing: the model currently in use by the Solar Team. While the intention is to use a nowcasting model in combination with the current model in use by the solar team, it would be beneficial to known the performance difference. A comparison between the novel model and the NWP model as used by the Solar Team is missing because it requires much processing to match the available forecasts of the NWP model to the test data set as used in this research. Since the general performance of nowcasting models using satellite data compared to NWP models is already well established, it was decided not to include such a comparison in this research.

6.3 Recommendation

Based on the results and the experience of this research, a recommendation towards the Solar Team can be made. The considerations for the Solar Team to employ an additional forecast model in their race strategy are the gained performance and the required time for implementation. Because the Solar Team is very busy with many aspects of designing the solar car, as well as the strategy model, decisions have to be made regarding what to work on. Therefore, it is not recommended for them to implement a new weather forecast model themselves. Both deep learning and optical flow methods require a deep understanding of the inner workings, as well as much time to implement them. Consequently, a solar

surface irradiance forecasting service could be employed, similarly to the current service provided by the ECMWF. While deep learning-based methods would outperform other methods, currently there is no service available that employs deep learning. Most services that provide SSI forecast use satellite obtained SSI, and displace them using NWP obtained winds. However, the services that employ optical flow methods on satellite obtained SSI achieve higher accuracy, and are therefore recommended to be used by the Solar Team.

7 Conclusion

In this thesis, the encoder-forecaster-decoder framework was proposed, a deep learning approach for forecasting cloudage. This framework was proposed based on observations from related work, and its exact configuration was determined based on the research questions as stated in section 1.

The first sub research question regarded what models could be used in the framework: *What are suitable models to be used as the encoder, decoder, and forecaster elements?* Based on several requirements and desired characteristics, different models were selected. The different combinations of these models were implemented in the framework, and their performance was compared between themselves. In the end, a SegNet based encoder and decoder were used, and a TrajGRU element as the forecaster.

Secondly, the obtained model was optimised, based on the second sub research question: *How many images in the input sequence is optimal?* An exhaustive grid search was used on the number of input images, as well as the number of recurrent layers. The performance of the 25 different models was compared between them, and it was determined that four input images and one layer were optimal.

Third, the performance of the optimised novel model was evaluated on four different forecast horizons to answer the third sub research question: *What are the possible forecasting horizons?* Because the behaviour of the novel model was atypical, and the underlying problem of this behaviour could not be determined, this question remains unanswered.

In conclusion, the main research question is answered: *How does an encoder-forecaster-decoder modular spatiotemporal sequence forecasting deep learning model compare against state-of-the-art spatiotemporal sequence forecasting models with regards to cloudage forecasting?* To this end, the performance of the novel model was compared to the performance of a baseline and three state-of-the-art models, and it was determined that the novel model does not achieve similar performance, and seems to lacks the ability to actual forecast which is likely due to the underlying framework.

8 Future work

Through the performed research, several possibilities for future work have been discovered.

First, while the current design of the encoder-forecaster-decoder framework is not optimal, additional research regarding connections between the encoder and decoder could result in better forecasts. Although the TrajGRU model already uses such connection, its architecture does not allow for a modular approach to find the optimal elements to be used in the network.

Secondly, as it has been shown that the state-of-the-art STSF models can forecast cloudage based on cloud masks, a more practical use-case for the forecasting of SSI would be to forecast optical thickness, or even SSI directly. The forecasting of optical thickness would probably not require many adjustments to the TrajGRU model, as it is very similar to cloud mask forecasting. However, the forecasting of SSI might require adjustments to the architecture to take the additional dependency on the position of the sun into account.

Third, the benefits of the encoded persistence forecast as a baseline are evident: it provides a baseline that has gone through the same loss of information as a forecasting model, more clearly showing the ability of that model to make accurate forecasts. However, as shown in this research, its performance is dependant on the ever-adapting model. Additional research to better employ the encoded persistence forecast is required, for example, by adding the loss of the encoded persistence forecast to the loss of the model, making it perhaps a more stable baseline.

References

[1] E. M. Guillot, T. H. Vonder Haar, J. M. Forsythe, and S. J. Fletcher, "Evaluating satellite-based cloud persistence and displacement nowcasting techniques over complex terrain," *Weather and forecasting*, vol. 27, no. 2, pp. 502–514, 2012.

[2] S. Pelland, J. Remund, J. Kleissl, T. Oozeki, and K. De Brabandere, *Photovoltaic and Solar Forecasting: State of the Art*. INTERNATIONAL ENERGY AGENCY, 10 2013.

[3] R. Perez, P. Ineichen, K. Moore, M. Kmiecik, C. Chain, R. George, and F. Vignola, "A new operational model for satellite-derived irradiances: description and validation," *Solar Energy*, vol. 73, no. 5, pp. 307–317, 2002.

[4] C. Gautier, G. Diak, and S. Masse, "A simple physical model to estimate incident solar radiation at the surface from goes satellite data," *Journal of Applied Meteorology*, vol. 19, no. 8, pp. 1005–1012, 1980.

[5] S. D. Miller, M. A. Rogers, J. M. Haynes, M. Sengupta, and A. K. Heidinger, "Short-term solar irradiance forecasting via satellite/model coupling," *Solar Energy*, vol. 168, pp. 102–117, 2018.

[6] G. Huang, Z. Li, X. Li, S. Liang, K. Yang, D. Wang, and Y. Zhang, "Estimating surface solar irradiance from satellites: Past, present, and future perspectives," *Remote Sensing of Environment*, vol. 233, p. 111371, 2019.

[7] C. I. Team, "Cloud effects on earth's radiation," Apr 2000.

[8] T. Fujita, D. L. Bradbury, and C. Murino, *A study of mesoscale cloud motions computed from ATS-I and terrestrial photographs*. Department of the Geophysical sciences, University of Chicago, 1968.

[9] W. P. Menzel, "Cloud tracking with satellite imagery: From the pioneering work of ted fujita to the present," *Bulletin of the American Meteorological Society*, vol. 82, no. 1, pp. 33–48, 2001.

[10] X. Shi and D.-Y. Yeung, "Machine learning for spatiotemporal sequence forecasting: A survey," *arXiv preprint arXiv:1808.06865*, 2018.

[11] H. S. Muench, *Short-range Forecasting of Cloudiness and Precipitation Through Extrapolation of GOES Imagery*. Air Force Geophysics Laboratories, Air Force Systems Command, United States, 1981.

[12] M. D. Binder, N. Hirokawa, and U. Windhorst, eds., *Aperture Problem*, pp. 159–159. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[13] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323, 2011.

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

[15] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[16] Z. Wojna, V. Ferrari, S. Guadarrama, N. Silberman, L.-C. Chen, A. Fathi, and J. Uijlings, "The devil is in the decoder: Classification, regression and gans," *International Journal of Computer Vision*, vol. 127, no. 11-12, pp. 1694–1706, 2019.

[17] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *2010 IEEE Computer Society Conference on computer vision and pattern recognition*, pp. 2528–2535, IEEE, 2010.

[18] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[19] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

[20] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE international conference on computer vision*, pp. 1520–1528, 2015.

[21] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*, pp. 1310–1318, 2013.

[22] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[23] G. Weiss, Y. Goldberg, and E. Yahav, "On the practical computational power of finite precision rnns for language recognition," *arXiv preprint arXiv:1805.04908*, 2018.

[24] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to construct deep recurrent neural networks," *arXiv preprint arXiv:1312.6026*, 2013.

[25] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss functions for image restoration with neural networks," *IEEE Transactions on computational imaging*, vol. 3, no. 1, pp. 47–57, 2016.

[26] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[27] C. Zach, T. Pock, and H. Bischof, "A duality based approach for realtime tv-l 1 optical flow," in *Joint pattern recognition symposium*, pp. 214–223, Springer, 2007.

[28] J. S. Pérez, E. Meinhardt-Llopis, and G. Facciolo, "Tv-l1 optical flow estimation," *Image Processing On Line*, vol. 2013, pp. 137–150, 2013.

[29] I. Urbich, J. Bendix, and R. Müller, "A novel approach for the short-term forecast of the effective cloud albedo," *Remote Sensing*, vol. 10, no. 6, 2018.

[30] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, pp. 802–810, 2015.

[31] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using lstms," in *International conference on machine learning*, pp. 843–852, 2015.

[32] X. Shi, Z. Gao, L. Lausen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo, "Deep learning for precipitation nowcasting: A benchmark and a new model," in *Advances in neural information processing systems*, pp. 5617–5627, 2017.

[33] C. Tan, X. Feng, J. Long, and L. Geng, "Forecast-clstm: A new convolutional lstm network for cloudage nowcasting," in *2018 IEEE Visual Communications and Image Processing (VCIP)*, pp. 1–4, IEEE, 2018.

[34] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh, "Action-conditional video prediction using deep networks in atari games," in *Advances in neural information processing systems*, pp. 2863–2871, 2015.

[35] R. Goroshin, M. F. Mathieu, and Y. LeCun, "Learning to linearize under uncertainty," in *Advances in Neural Information Processing Systems*, pp. 1234–1242, 2015.

[36] J. Yang, B. Price, S. Cohen, H. Lee, and M.-H. Yang, "Object contour detection with a fully convolutional encoder-decoder network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 193–202, 2016.

[37] A. A. Shvets, V. I. Iglovikov, A. Rakhlin, and A. A. Kalinin, "Angiodysplasia detection and localization using deep convolutional neural networks," in *2018 17th ieee international conference on machine learning and applications (icmla)*, pp. 612–617, IEEE, 2018.

[38] R. Yasrab, "Ecru: An encoder-decoder based convolution neural network (cnn) for road-scene understanding," *Journal of Imaging*, vol. 4, no. 10, p. 116, 2018.

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

[40] S. Finkensieper, J. Meirink, G. van Zadelhoff, T. Hanschmann, N. Benas, M. Stengel, P. Fuchs, R. Hollmann, and M. Werscheck, "Claas-2: Cm saf cloud property dataset using seviri–edition 2, satellite application facility on climate monitoring," *Satellite Appl. Facil. Clim. Monit*, 2016.

[41] C. SAF, "Algorithm theoretical basis document, seviri cloud physical products, claas edition 2, eumetsat satellite application facility on climate monitoring," tech. rep., SAF/CM/KNMI/ATBD/SEVIRI/CPP, 2016.

[42] J. Schmid, "The seviri instrument," in *Proceedings of the 2000 EUMETSAT meteorological satellite data user's conference, Bologna, Italy*, vol. 29, 2000.

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

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

[45] M. Hermans and B. Schrauwen, "Training and analysing deep recurrent neural networks," in *Advances in neural information processing systems*, pp. 190–198, 2013.

[46] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts," *Distill*, 2016.

Appendices

A Glossary

CALIPSO	Cloud-Aerosol Lidar and Infrared Pathfinder Satellite Observations
CALIOP	Cloud-Aerosol Lidar with Orthogonal Polarization
CLAAS-2	CLoud property dAtAset using SEVIRI - Edition 2
CM SAF	Satellite Application Facility on Climate Monitoring
CNN	Convolutional Neural Network
ConvLSTM	Convolutional Long Short-Term Memory
ConvRNN	Convolutional Recurrent Neural Network
ECMWF	European Centre for Medium-Range Weather Forecasts
EUMETSAT	European Organisation for the Exploitation of Meteorological Satellites
GOES	Geostationary Operational Environmental Satellite
GRU	Gated Recurrent Unit.
INSAT	Indian National Satellite System
JMA	Japan Meteorological Agency
LIDAR	Laser Imaging, Detection, And Ranging
LSTM	Long Short-Term Memory
ML	Machine Learning
MSG	Meteosat Second Generation
NOAA	National Oceanic and Atmospheric Administration
NWP	Numerical Weather Prediction
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SSI	Surface Solar Irradiance
STSF	SpatioTemporal Sequence Forecasting
TrajGRU	Trajectory Gated Recurrent United
TV	Total Variation

B Network implementation details

In this appendix the details of the architectures of the used deep neural networks are described. The tables contain all the layers present in the models. In principle, the input would arrive at the top of the table, and is passed between the layers from the top to bottom row. The activation functions have been ommited for readability.

Table 14: Encoder - The hyperparameters of the layers of the SegNet based encoder.

Name	Type	Kernel	Stride	Padding	In shape	Out shape
Stage 1 Encoder -						
conv 1	Conv	3×3	1×1	1×1	$1 \times 363 \times 363$	$64 \times 363 \times 363$
conv 2	Conv	3×3	1×1	1×1	$64 \times 363 \times 363$	$64 \times 363 \times 363$
pool	Max-pool	2×2	2×2	0×0	$64 \times 363 \times 363$	$64 \times 181 \times 181$
Stage 2 Encoder -						
conv 1	Conv	3×3	1×1	1×1	$64 \times 181 \times 181$	$128 \times 181 \times 181$
conv 2	Conv	3×3	1×1	1×1	$128 \times 181 \times 181$	$128 \times 181 \times 181$
pool	Max-pool	2×2	2×2	0×0	$128 \times 181 \times 181$	$128 \times 90 \times 90$
Stage 3 Encoder -						
conv 1	Conv	3×3	1×1	1×1	$128 \times 90 \times 90$	$256 \times 90 \times 90$
conv 2	Conv	3×3	1×1	1×1	$256 \times 90 \times 90$	$256 \times 90 \times 90$
conv 3	Conv	3×3	1×1	1×1	$256 \times 90 \times 90$	$256 \times 90 \times 90$
pool	Max-pool	2×2	2×2	0×0	$256 \times 90 \times 90$	$256 \times 45 \times 45$
Stage 4 Encoder -						
conv 1	Conv	3×3	1×1	1×1	$256 \times 45 \times 45$	$512 \times 45 \times 45$
conv 2	Conv	3×3	1×1	1×1	$512 \times 45 \times 45$	$512 \times 45 \times 45$
conv 3	Conv	3×3	1×1	1×1	$512 \times 45 \times 45$	$512 \times 45 \times 45$
pool	Max-pool	2×2	2×2	0×0	$512 \times 45 \times 45$	$512 \times 22 \times 22$
Stage 5 Encoder -						
conv 1	Conv	3×3	1×1	1×1	$512 \times 22 \times 22$	$512 \times 22 \times 22$
conv 2	Conv	3×3	1×1	1×1	$512 \times 22 \times 22$	$512 \times 22 \times 22$
conv 3	Conv	3×3	1×1	1×1	$512 \times 22 \times 22$	$512 \times 22 \times 22$
pool	Max-pool	2×2	2×2	0×0	$512 \times 22 \times 22$	$512 \times 11 \times 11$

Table 15: Bilinear Upsampling - The hyperparameters of the layers of the bilinear upsampling decoder.

Name	Type	Kernel	Stride	Padding	In shape	Out shape
Stage 5 Decoder -						
upsample	Bilinear	-	-	-	$512 \times 11 \times 11$	$512 \times 22 \times 22$
conv 3	Conv	3×3	1×1	1×1	$512 \times 22 \times 22$	$512 \times 22 \times 22$
conv 2	Conv	3×3	1×1	1×1	$512 \times 22 \times 22$	$512 \times 22 \times 22$
conv 1	Conv	3×3	1×1	1×1	$512 \times 22 \times 22$	$512 \times 22 \times 22$
Stage 4 Decoder -						
upsample	Bilinear	-	-	-	$512 \times 22 \times 22$	$512 \times 45 \times 45$
conv 3	Conv	3×3	1×1	1×1	$512 \times 45 \times 45$	$512 \times 45 \times 45$
conv 2	Conv	3×3	1×1	1×1	$512 \times 45 \times 45$	$512 \times 45 \times 45$
conv 1	Conv	3×3	1×1	1×1	$512 \times 45 \times 45$	$256 \times 45 \times 45$
Stage 3 Decoder -						
upsample	Bilinear	-	-	-	$256 \times 45 \times 45$	$256 \times 90 \times 90$
conv 3	Conv	3×3	1×1	1×1	$256 \times 90 \times 90$	$256 \times 90 \times 90$
conv 2	Conv	3×3	1×1	1×1	$256 \times 90 \times 90$	$256 \times 90 \times 90$
conv 1	Conv	3×3	1×1	1×1	$256 \times 90 \times 90$	$128 \times 90 \times 90$
Stage 2 Decoder -						
upsample	Bilinear	-	-	-	$128 \times 90 \times 90$	$128 \times 181 \times 181$
conv 2	Conv	3×3	1×1	1×1	$128 \times 181 \times 181$	$128 \times 181 \times 181$
conv 1	Conv	3×3	1×1	1×1	$128 \times 181 \times 181$	$64 \times 181 \times 181$
Stage 1 Decoder -						
upsample	Bilinear	-	-	-	$64 \times 181 \times 181$	$64 \times 363 \times 363$
conv 2	Conv	3×3	1×1	1×1	$64 \times 363 \times 363$	$64 \times 363 \times 363$
conv 1	Conv	3×3	1×1	1×1	$64 \times 363 \times 363$	$1 \times 363 \times 363$

Table 16: Bilinear Additive Upsampling - The hyperparameters of the layers of the bilinear additive upsampling decoder.

Name	Type	Kernel	Stride	Padding	In shape	Out shape	Other params
Stage 5 Decoder -							
upsample	Bilinear	-	-	-	$512 \times 11 \times 11$	$512 \times 22 \times 22$	
average	Average channels	-	-	-	$512 \times 22 \times 22$	$128 \times 22 \times 22$	ratio $= 4$
conv 3	Conv	3×3	1×1	1×1	$128 \times 22 \times 22$	$128 \times 22 \times 22$	
conv 2	Conv	3×3	1×1	1×1	$128 \times 22 \times 22$	$128 \times 22 \times 22$	
conv 1	Conv	3×3	1×1	1×1	$128 \times 22 \times 22$	$128 \times 22 \times 22$	
Stage 4 Decoder -							
upsample	Bilinear	-	-	-	$128 \times 22 \times 22$	$128 \times 45 \times 45$	
average	Average channels	-	-	-	$128 \times 45 \times 45$	$32 \times 45 \times 45$	ratio $= 4$
conv 3	Conv	3×3	1×1	1×1	$32 \times 45 \times 45$	$32 \times 45 \times 45$	
conv 2	Conv	3×3	1×1	1×1	$32 \times 45 \times 45$	$32 \times 45 \times 45$	
conv 1	Conv	3×3	1×1	1×1	$32 \times 45 \times 45$	$32 \times 45 \times 45$	
Stage 3 Decoder -							
upsample	Bilinear	-	-	-	$32 \times 45 \times 45$	$32 \times 90 \times 90$	
average	Average channels	-	-	-	$32 \times 90 \times 90$	$8 \times 90 \times 90$	ratio $= 4$
conv 3	Conv	3×3	1×1	1×1	$8 \times 90 \times 90$	$8 \times 90 \times 90$	
conv 2	Conv	3×3	1×1	1×1	$8 \times 90 \times 90$	$8 \times 90 \times 90$	
conv 1	Conv	3×3	1×1	1×1	$8 \times 90 \times 90$	$8 \times 90 \times 90$	
Stage 2 Decoder -							
upsample	Bilinear	-	-	-	$8 \times 90 \times 90$	$8 \times 181 \times 181$	
average	Average channels	-	-	-	$8 \times 181 \times 181$	$2 \times 181 \times 181$	ratio $= 4$
conv 2	Conv	3×3	1×1	1×1	$2 \times 181 \times 181$	$2 \times 181 \times 181$	
conv 1	Conv	3×3	1×1	1×1	$2 \times 181 \times 181$	$2 \times 181 \times 181$	
Stage 1 Decoder -							
upsample	Bilinear	-	-	-	$2 \times 181 \times 181$	$2 \times 363 \times 363$	

average	Average channels	-	-	-	$2 \times 363 \times 363$	$1 \times 363 \times 363$	ratio = 2
conv 2	Conv	3×3	1×1	1×1	$1 \times 363 \times 363$	$1 \times 363 \times 363$	
conv 1	Conv	3×3	1×1	1×1	$1 \times 363 \times 363$	$1 \times 363 \times 363$	

Table 17: Bilinear Upsampling & Seperable Convolution - The hyperparameters of the layers of the bilinear & seperable convolution upsampling decoder.

Name	Type	Kernel	Stride	Padding	In shape	Out shape	Other params
Stage 5 Decoder -							
upsample	Bilinear	-	-	-	$512 \times 11 \times 11$	$512 \times 22 \times 22$	
depthwise	Conv	3×3	1×1	1×1	$512 \times 22 \times 22$	$1536 \times 22 \times 22$	groups = 512
pointwise	Conv	1×1	1×1	0×0	$1536 \times 22 \times 22$	$512 \times 22 \times 22$	
conv 3	Conv	3×3	1×1	1×1	$512 \times 22 \times 22$	$512 \times 22 \times 22$	
conv 2	Conv	3×3	1×1	1×1	$512 \times 22 \times 22$	$512 \times 22 \times 22$	
conv 1	Conv	3×3	1×1	1×1	$512 \times 22 \times 22$	$512 \times 22 \times 22$	
Stage 4 Decoder -							
upsample	Bilinear	-	-	-	$512 \times 22 \times 22$	$512 \times 45 \times 45$	
depthwise	Conv	3×3	1×1	1×1	$512 \times 45 \times 45$	$1536 \times 45 \times 45$	groups = 512
pointwise	Conv	1×1	1×1	0×0	$1536 \times 45 \times 45$	$512 \times 45 \times 45$	
conv 3	Conv	3×3	1×1	1×1	$512 \times 45 \times 45$	$512 \times 45 \times 45$	
conv 2	Conv	3×3	1×1	1×1	$512 \times 45 \times 45$	$512 \times 45 \times 45$	
conv 1	Conv	3×3	1×1	1×1	$512 \times 45 \times 45$	$256 \times 45 \times 45$	
Stage 3 Decoder -							
upsample	Bilinear	-	-	-	$256 \times 45 \times 45$	$256 \times 90 \times 90$	
depthwise	Conv	3×3	1×1	1×1	$256 \times 90 \times 90$	$769 \times 90 \times 90$	groups = 256
pointwise	Conv	1×1	1×1	0×0	$768 \times 90 \times 90$	$256 \times 90 \times 90$	
conv 3	Conv	3×3	1×1	1×1	$256 \times 90 \times 90$	$256 \times 90 \times 90$	
conv 2	Conv	3×3	1×1	1×1	$256 \times 90 \times 90$	$256 \times 90 \times 90$	
conv 1	Conv	3×3	1×1	1×1	$256 \times 90 \times 90$	$128 \times 90 \times 90$	
Stage 2 Decoder -							
upsample	Bilinear	-	-	-	$128 \times 90 \times 90$	$128 \times 181 \times 181$	
depthwise	Conv	3×3	1×1	1×1	$128 \times 181 \times 181$	$384 \times 181 \times 181$	groups = 128

pointwise	Conv	1×1	1×1	0×0	$384 \times 181 \times 181$	$128 \times 181 \times 181$	
conv 2	Conv	3×3	1×1	1×1	$128 \times 181 \times 181$	$128 \times 181 \times 181$	
conv 1	Conv	3×3	1×1	1×1	$128 \times 181 \times 181$	$64 \times 181 \times 181$	
Stage 1 Decoder -							
upsample	Bilinear	-	-	-	$64 \times 181 \times 181$	$64 \times 363 \times 363$	
depthwise	Conv	3×3	1×1	1×1	$64 \times 363 \times 363$	$192 \times 363 \times 363$	groups = 64
pointwise	Conv	1×1	1×1	0×0	$192 \times 363 \times 363$	$64 \times 363 \times 363$	
conv 2	Conv	3×3	1×1	1×1	$64 \times 363 \times 363$	$64 \times 363 \times 363$	
conv 1	Conv	3×3	1×1	1×1	$64 \times 363 \times 363$	$1 \times 363 \times 363$	

Table 18: Decomposed Transposed Convolution Upsampling - The hyperparameters of the layers of the decomposed transposed convolution upsampling decoder.

Name	Type	Kernel	Stride	Padding	In shape	Out shape	Other params
Stage 5 Decoder -							
horizontal upsample	Transposed Conv	1×2	1×2	0×0	$512 \times 11 \times 11$	$512 \times 11 \times 22$	output padding $= 0 \times 1$
vertical upsample	Transposed Conv	2×1	2×1	0×0	$512 \times 11 \times 22$	$512 \times 22 \times 22$	output padding $= 1 \times 0$
conv 3	Conv	3×3	1×1	1×1	$512 \times 22 \times 22$	$512 \times 22 \times 22$	
conv 2	Conv	3×3	1×1	1×1	$512 \times 22 \times 22$	$512 \times 22 \times 22$	
conv 1	Conv	3×3	1×1	1×1	$512 \times 22 \times 22$	$512 \times 22 \times 22$	
Stage 4 Decoder -							
horizontal upsample	Transposed Conv	1×2	1×2	0×0	$512 \times 22 \times 22$	$512 \times 22 \times 45$	output padding $= 0 \times 1$
vertical upsample	Transposed Conv	2×1	2×1	0×0	$512 \times 22 \times 45$	$512 \times 45 \times 45$	output padding $= 1 \times 0$
conv 3	Conv	3×3	1×1	1×1	$512 \times 45 \times 45$	$512 \times 45 \times 45$	
conv 2	Conv	3×3	1×1	1×1	$512 \times 45 \times 45$	$512 \times 45 \times 45$	
conv 1	Conv	3×3	1×1	1×1	$512 \times 45 \times 45$	$256 \times 45 \times 45$	
Stage 3 Decoder -							
horizontal upsample	Transposed Conv	1×2	1×2	0×0	$256 \times 45 \times 45$	$256 \times 45 \times 90$	output padding $= 0 \times 1$
vertical upsample	Transposed Conv	2×1	2×1	0×0	$256 \times 45 \times 90$	$256 \times 90 \times 90$	output padding $= 1 \times 0$
conv 3	Conv	3×3	1×1	1×1	$256 \times 90 \times 90$	$256 \times 90 \times 90$	
conv 2	Conv	3×3	1×1	1×1	$256 \times 90 \times 90$	$256 \times 90 \times 90$	
conv 1	Conv	3×3	1×1	1×1	$256 \times 90 \times 90$	$128 \times 90 \times 90$	
Stage 2 Decoder -							
horizontal upsample	Transposed Conv	1×2	1×2	0×0	$128 \times 90 \times 90$	$128 \times 90 \times 181$	output padding $= 0 \times 1$
vertical upsample	Transposed Conv	2×1	2×1	0×0	$128 \times 90 \times 181$	$256 \times 181 \times 181$	output padding $= 1 \times 0$
conv 2	Conv	3×3	1×1	1×1	$128 \times 181 \times 181$	$128 \times 181 \times 181$	
conv 1	Conv	3×3	1×1	1×1	$128 \times 181 \times 181$	$64 \times 181 \times 181$	
Stage 1 Decoder -							

horizontal upsample	Transposed Conv	1×2	1×2	0×0	$64 \times 181 \times 181$	$64 \times 363 \times 363$	output padding $= 0 \times 1$
vertical upsample	Transposed Conv	2×1	2×1	0×0	$64 \times 181 \times 363$	$64 \times 363 \times 363$	output padding $= 1 \times 0$
conv 2	Conv	3×3	1×1	1×1	$64 \times 363 \times 363$	$64 \times 363 \times 363$	
conv 1	Conv	3×3	1×1	1×1	$64 \times 363 \times 363$	$1 \times 363 \times 363$	

Table 19: Transposed Convolution Upsampling - The hyperparameters of the layers of the transposed convolution upsampling decoder.

Name	Type	Kernel	Stride	Padding	In shape	Out shape	Other params
Stage 5 Decoder -							
upsample	Transposed Conv	2×2	2×2	0×0	$512 \times 11 \times 11$	$512 \times 22 \times 22$	output padding $= 1 \times 1$
conv 3	Conv	3×3	1×1	1×1	$512 \times 22 \times 22$	$512 \times 22 \times 22$	
conv 2	Conv	3×3	1×1	1×1	$512 \times 22 \times 22$	$512 \times 22 \times 22$	
conv 1	Conv	3×3	1×1	1×1	$512 \times 22 \times 22$	$512 \times 22 \times 22$	
Stage 4 Decoder -							
upsample	Transposed Conv	2×2	2×2	0×0	$512 \times 22 \times 22$	$512 \times 45 \times 45$	output padding $= 1 \times 1$
conv 3	Conv	3×3	1×1	1×1	$512 \times 45 \times 45$	$512 \times 45 \times 45$	
conv 2	Conv	3×3	1×1	1×1	$512 \times 45 \times 45$	$512 \times 45 \times 45$	
conv 1	Conv	3×3	1×1	1×1	$512 \times 45 \times 45$	$256 \times 45 \times 45$	
Stage 3 Decoder -							
upsample	Transposed Conv	2×2	2×2	0×0	$256 \times 45 \times 45$	$256 \times 90 \times 90$	output padding $= 1 \times 1$
conv 3	Conv	3×3	1×1	1×1	$256 \times 90 \times 90$	$256 \times 90 \times 90$	
conv 2	Conv	3×3	1×1	1×1	$256 \times 90 \times 90$	$256 \times 90 \times 90$	
conv 1	Conv	3×3	1×1	1×1	$256 \times 90 \times 90$	$128 \times 90 \times 90$	
Stage 2 Decoder -							
upsample	Transposed Conv	2×2	2×2	0×0	$128 \times 90 \times 90$	$128 \times 181 \times 181$	output padding $= 1 \times 1$
conv 2	Conv	3×3	1×1	1×1	$128 \times 181 \times 181$	$128 \times 181 \times 181$	
conv 1	Conv	3×3	1×1	1×1	$128 \times 181 \times 181$	$64 \times 181 \times 181$	
Stage 1 Decoder -							
upsample	Transposed Conv	2×2	2×2	0×0	$64 \times 181 \times 181$	$64 \times 363 \times 363$	output padding $= 1 \times 1$
conv 2	Conv	3×3	1×1	1×1	$64 \times 363 \times 363$	$64 \times 363 \times 363$	
conv 1	Conv	3×3	1×1	1×1	$64 \times 363 \times 363$	$1 \times 363 \times 363$	

B.3 Forecaster

Table 20: ConvLSTM Cell

Name	Type	Kernel	Stride	Padding	Channel I/O
Wxi	Conv	3×3	1×1	1×1	512/512
Whi	Conv	3×3	1×1	1×1	512/512
Wxf	Conv	3×3	1×1	1×1	512/512
Whf	Conv	3×3	1×1	1×1	512/512
Wxc	Conv	3×3	1×1	1×1	512/512
Whc	Conv	3×3	1×1	1×1	512/512
Wxo	Conv	3×3	1×1	1×1	512/512
Who	Conv	3×3	1×1	1×1	512/512

Table 21: TrajGRU Cell

Name	Type	Kernel	Stride	Padding	Channel I/O
Flow generator					
i2f_conv	Conv	5×5	1×1	2×2	512/32
h2f_conv	Conv	5×5	1×1	2×2	512/32
u_conv	Conv	5×5	1×1	2×2	32/L, L = 9
v_conv	Conv	5×5	1×1	2×2	32/L, L = 9
TrajGRU					
Wxz	Conv	3×3	1×1	1×1	512/512
Wxr	Conv	3×3	1×1	1×1	512/512
Wxh	Conv	3×3	1×1	1×1	512/512
Whz	Conv	1×1	1×1	0×0	$512 \cdot \mathrm{L}/512$, L = 9
Whr	Conv	1×1	1×1	0×0	$512 \cdot \mathrm{L}/512$, L = 9
Whh	Conv	1×1	1×1	0×0	$512 \cdot \mathrm{L}/512$, L = 9

Table 22: TrajGRU evaluation model.

Name	Type	Kernel	Stride	Padding	In shape	Out shape	Other params
Encoder							
conv1	Conv	7×7	5×5	1×1	$1 \times 363 \times 363$	$8 \times 72 \times 72$	
rnn1	TrajGRU	3×3	1×1	1×1	$8 \times 72 \times 72$	$64 \times 72 \times 72$	L = 13
conv2	Conv	5×5	3×3	1×1	$64 \times 72 \times 72$	$64 \times 24 \times 24$	
rnn2	TrajGRU	3×3	1×1	1×1	$64 \times 24 \times 24$	$192 \times 24 \times 24$	L = 13
conv3	Conv	3×3	2×2	1×1	$192 \times 24 \times 24$	$192 \times 12 \times 12$	
rnn3	TrajGRU	3×3	1×1	1×1	$192 \times 12 \times 12$	$192 \times 12 \times 12$	L = 9
Forecaster							
rnn1	TrajGRU	3×3	1×1	1×1	$192 \times 12 \times 12$	$192 \times 12 \times 12$	L = 9
conv1	Transposed Conv	4×4	2×2	1×1	$192 \times 12 \times 12$	$192 \times 24 \times 24$	
rnn2	TrajGRU	3×3	1×1	1×1	$192 \times 24 \times 24$	$192 \times 24 \times 24$	L = 13
conv2	Transposed Conv	5×5	3×3	1×1	$192 \times 24 \times 24$	$64 \times 72 \times 72$	
rnn3	TrajGRU	3×3	1×1	1×1	$64 \times 72 \times 72$	$64 \times 72 \times 72$	L = 13
conv3	Transposed Conv	7×7	5×5	1×1	$64 \times 72 \times 72$	$8 \times 363 \times 363$	
conv4	Conv	1×1	1×1	0×0	$8 \times 363 \times 363$	$1 \times 363 \times 363$	

Table 23: ConvLSTM evaluation model.

Name	Type	Kernel	Stride	Padding	In shape	Out shape
Encoder						
conv1	Conv	7×7	5×5	1×1	$1 \times 363 \times 363$	$8 \times 72 \times 72$
rnn1	ConvLSTM	3×3	1×1	1×1	$8 \times 72 \times 72$	$64 \times 72 \times 72$
conv2	Conv	5×5	3×3	1×1	$64 \times 72 \times 72$	$64 \times 24 \times 24$
rnn2	ConvLSTM	3×3	1×1	1×1	$64 \times 24 \times 24$	$192 \times 24 \times 24$
conv3	Conv	3×3	2×2	1×1	$192 \times 24 \times 24$	$192 \times 12 \times 12$
rnn3	ConvLSTM	3×3	1×1	1×1	$192 \times 12 \times 12$	$192 \times 12 \times 12$
Forecaster						
rnn1	ConvLSTM	3×3	1×1	1×1	$192 \times 12 \times 12$	$192 \times 12 \times 12$
conv1	Transposed Conv	4×4	2×2	1×1	$192 \times 12 \times 12$	$192 \times 24 \times 24$
rnn2	ConvLSTM	3×3	1×1	1×1	$192 \times 24 \times 24$	$192 \times 24 \times 24$
conv2	Transposed Conv	5×5	3×3	1×1	$192 \times 24 \times 24$	$192 \times 72 \times 72$
rnn3	ConvLSTM	3×3	1×1	1×1	$192 \times 72 \times 72$	$64 \times 72 \times 72$
conv3	Transposed Conv	7×7	5×5	1×1	$64 \times 72 \times 72$	$8 \times 363 \times 363$
conv4	Conv	1×1	1×1	0×0	$8 \times 363 \times 363$	$1 \times 363 \times 363$

C Example of results

C.1 Integration

C.1.1 Encoder-decoder

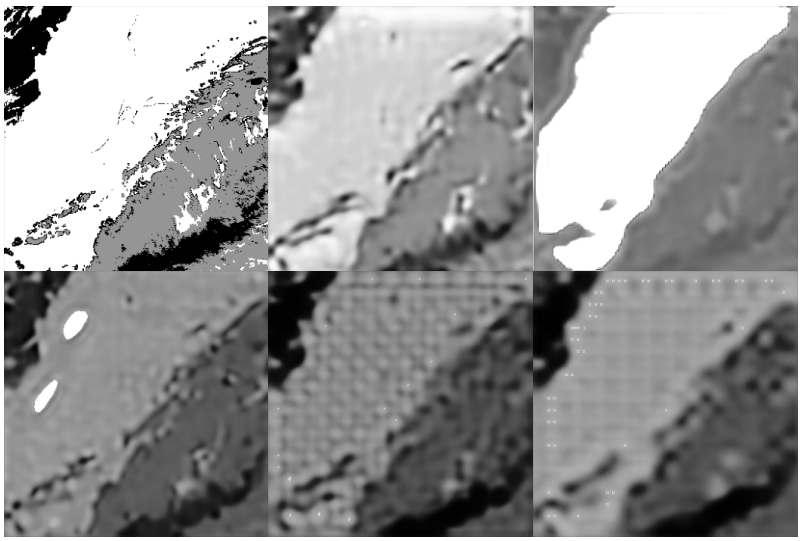


Figure 17: Examples of a reconstructed cloud mask by the different upsampling methods. First row from left to right: ground truth, bilinear upsampling & convolutions, bilinear additive upsampling. Second row from left to right: bilinear upsampling & separable convolutions, decomposed transposed convolution, transposed convolution.

C.1.2 Integrated networks

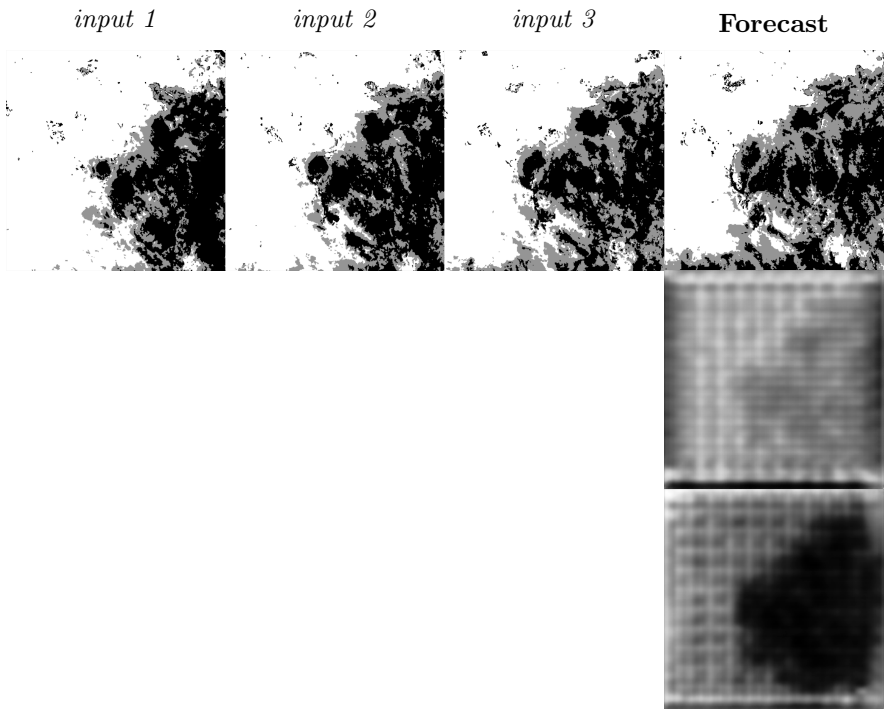


Figure 18: Examples of the reconstructed forecast made by the integrated models. The forecast are from top to bottom: ground truth, ConvLSTM, TrajGRU.

C.2 Optimisation

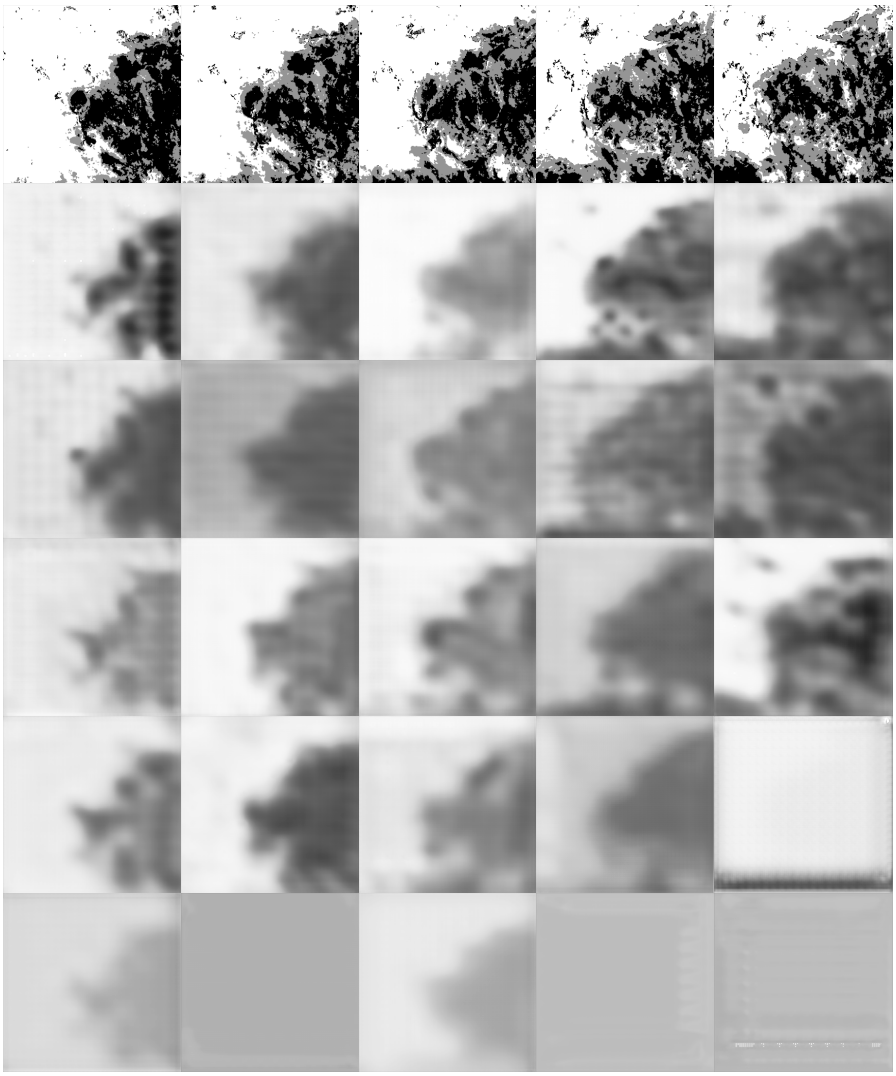


Figure 19: Examples of the reconstructed forecast made by the different models in the grid search. The rows are the number of layers, the columns are the number of input cloud masks. The first row are the ground truths, the second row are the models with one layer, down to the sixth row with the models with five layers. The first column are the models with one input cloud masks, the fifth columns are the models with five input cloud masks.

C.3 Evaluation

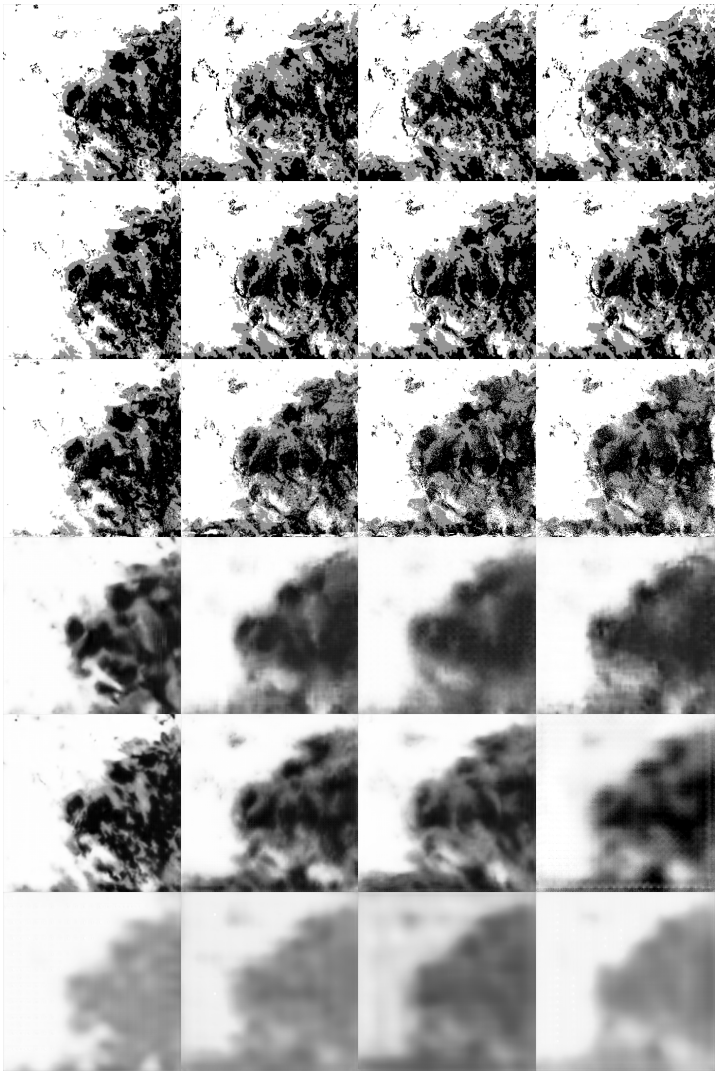


Figure 20: Examples of the forecasts made by the different model used during evaluation. The columns are the forecast horizons, from left to right: 30 minutes, 1 hour, 2 hours, 3 hours. The rows are the different models. From top to bottom: ground truth, persistence, TV-L_1, ConvLSTM evaluation models, TrajGRU evaluation models, and the novel model.