Diese Arbeit wurde vorgelegt am
Lehr- und Forschungsgebiet Theorie der hybriden Systeme

# Vorhersage von Wolkenschatten mittels Neuronaler Netze

## Nowcasting of Cloud Shadows using Neural Networks

Masterarbeit

März 2020


| | |
|---|---|
| Vorgelegt von<br>Presented by | Niels von Stein, 372338<br>niels.von.stein@rwth-aachen.de |
| Erstprüfer<br>First examiner | Prof. Dr. rer. nat. Erika Ábrahám<br>Lehr- und Forschungsgebiet: Theorie der hybriden Systeme<br>RWTH Aachen University |
| Zweitprüfer<br>Second examiner | Prof. Dr. rer. nat., apl. Thomas Noll<br>Lehr- und Forschungsgebiet: Theorie der hybriden Systeme<br>RWTH Aachen University |
| Koreferent<br>Co-supervisor | Dr. rer. nat. Pascal Richter<br>Lehr- und Forschungsgebiet: Theorie der hybriden Systeme<br>RWTH Aachen University |

# Eigenständigkeitserklärung

Hiermit versichere ich, dass ich diese Masterarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Stellen meiner Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen sind, habe ich in jedem Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht. Dasselbe gilt sinngemäß für Tabellen und Abbildungen. Diese Arbeit hat in dieser oder einer ähnlichen Form noch nicht im Rahmen einer anderen Prüfung vorgelegen.

Aachen, im März 2020

Niels von Stein

*"Wenn du etwas ganz fest willst, dann wird das Universum darauf hinwirken, dass du es erreichen kannst."*

**- Paulo Coelho**

# Contents


**1. Introduction** **1**

**2. Preliminaries** **3**
2.1. Solar Irradiance . . . . . 3
2.2. Clouds . . . . . 4
2.3. Neuronal Networks . . . . . 6

**3. Related Work** **9**

**4. Model** **12**
4.1. Forecasting Stage . . . . . 13
4.2. Extraction Stage . . . . . 15
4.3. Projection Stage . . . . . 17

**5. Evaluation** **19**
5.1. Simulation . . . . . 19
5.2. Dataset . . . . . 22
5.3. Pre-processing . . . . . 22
5.4. Error Metrics . . . . . 25
5.5. Forecasting Stage . . . . . 26
5.6. Extraction Stage . . . . . 32
5.7. Projection Stage . . . . . 35

**6. Discussion** **39**

**7. Conclusion** **41**

**A. Appendix** **42**

**Nomenclature** **55**

**Abbreviations** **56**

**References** **57**

# 1. Introduction

Since the adoption of the Paris Agreement on Climate Change in 2015 and its acceptance by 195 states, the containment of increasing temperature and consequently the reduction of carbon-dioxide emissions have become a global goal [1]. It is without doubt that the generation of power from renewable sources (renewable energy) plays a crucial role in this aspiration. In 2018 the overall share of renewable energy in the global power generating capacity amounts to 33%. A key technology in this development is solar power generation which showed the most absolute growth in the recent years [2, 3, 4]. This technology is divided into two types: Photovoltaics (PV) and Concentrated Solar Power (CSP) generation. In PV the incoming sunlight is directly converted into electricity using solar cell arrays mostly made from silicon [5]. In contrast, CSP uses reflective devices like mirrors to concentrate sunlight for heat production. This heat is then converted to electricity via a power block (see Figure 1) [6].

While each technology comes with its own advantages and disadvantages, both approaches have one thing in common: the performance capability depends on the availability and predictability of sunlight. On one hand there are deterministic shortcomings of sunlight caused by the natural day and night cycle. Energy producers and distribution grid operators can easily plan countermeasures for this type of event based on accurate solar position prediction. On the other hand there are less deterministic, intra-day weather events which affect a plants power output. This intermittency is a challenge for conventional distribution grids as energy supply and demand must be balanced. But energy balance is not the only concern when it comes to solar power generation at changing weather conditions. In CSP generation, a constant supply of heat is important to keep the production up. Sudden changes in the exposure of the collector system therefore require countermeasures like mirror re-adjustment or tapping of heat storages. Mirror adjustment is in need for spatially comprehensive forecasting while heat storages require a certain ramp up time making them unsuitable for short occlusion periods. The ability to predict solar irradiance with high spatial and temporal resolution is therefore an important factor for maintaining a stable power supply at low additional economic costs [7, 8].

Solar power forecasting can be grouped into three types. Long-term forecasting estimates the monthly or annual available resources on a low spatial resolution. This data usually supports commercial negotiations and is of less interest in operations. In contrast, short-term forecasting estimates hourly solar power from 6 hours up to 7 days ahead in global to meso-scale ($> 1km^2$) spatial resolution. This type allows for load forecasting, transmission scheduling and day ahead price calculations. The underlying models often origin from the domains of numerical weather prediction (NWP), satellite imagery or stochastics. The last type is nowcasting with a forecast lead time between 1 minute to 2 hours and up to intra-minute temporal and micro-scale ($< 1km^2$) spatial resolution. This forecast horizon has the most impact on operations as it enables prediction of e.g. energy ramping events due to passing clouds. Models in this group

often build on stochastics, satellite imagery or total sky imagery (TSI) systems [9].

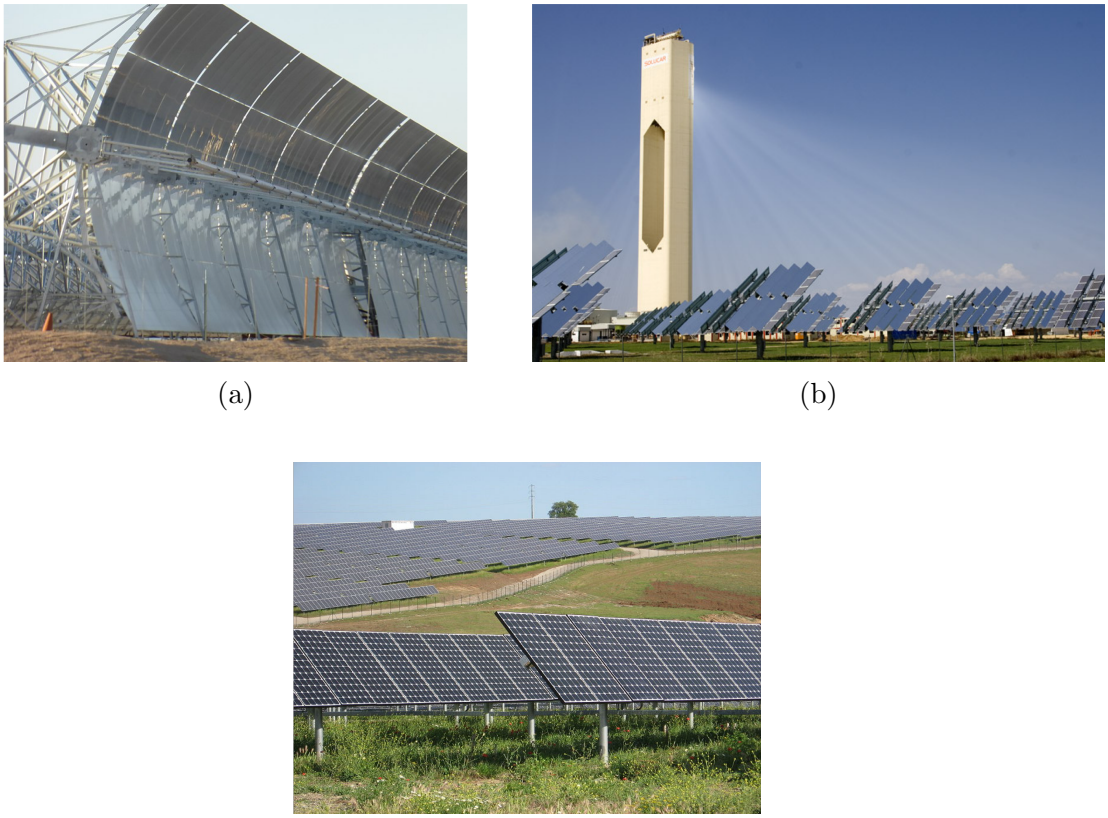
Accurate solar power nowcasting is becoming a feature with increasing importance for grid operators as the portion of volatile renewable energy source in the grid continues to rise [7, 10]. Although, state-of-the-art approaches for nowcasting solar irradiance deliver viable performance, the authors often identify similar pitfalls and potentials to improve [11, 12, 13]. Moreover, these image-based models have many fundamental methods in common like color channel thresholding for cloud pixel identification. Shortcomings of such basic operations have a huge impact on performance due to error propagation which opens up the question for different methodologies. A promising candidate are neural networks (NNs) in computer vision. The increasing capabilities of NNs have also been identified by the community and recent work fosters the use of fully connected NNs and/or convolutional neural networks (CNN) in their models. However, up to now these models are only capable of nowcasting either irradiance at a single spot or aggregated power output of an entire power plant.

The aim of this thesis is to close the gap of spatially comprehensive nowcasting based on state-of-the-art computer vision techniques. This work proposes and evaluates a end-to-end neural network model which is capable of predicting the current shadow occlusion state with high spatial resolution ($< 100m^2$) within a defined area ($2km \times 2km$). The model utilizes the visual input of a single fish-eye camera to nowcast up to 10 minutes ahead. A monocular system has the advantage of self-containment and reduced complexity while also allowing for discussions of general design choices and extensions. Training and evaluation is based on simulated images.

The document is organized as follow. Section 2 presents preliminary knowledge and terminologies used in this thesis. In Section 3 the current state of research in image-based irradiance nowcasting and computer vision is reviewed. Section 4 is dedicated to the model proposal while Section 5 evaluates the design. Section 6 discusses improvements for future research while Section 7 concludes the whole work.

# 2. Preliminaries

## 2.1. Solar Irradiance



(a) (b)

(c)

Figure 1: (a) Parabolic CSP plant in California, United States [14]. (b) PS10 power tower CSP plant in Spain [15]. (c) Dual axis tracking PV plant in Portugal [16].

Solar irradiance is the received power in form of electromagnetic radiation emitted by the sun. It is measured in watts per unit area e.g. $kW/m^2$. The amount reaching the outer atmosphere is referred as total solar irradiance and represents the total available solar energy budget. However, the accessible terrestrial power is influenced by solar position and atmospheric conditions. In order to better quantify meteorological effects on solar irradiance, there exist several *types* of irradiance. Direct Normal Irradiance (DNI) is the amount of power reaching a surface perpendicular to the path of incoming light minus atmospheric losses due to absorption and scattering. Because of the planetary shape, this is only the case at midday in areas around the equator while elsewhere the light comes in at an angle. Thus, this term is weighted by the cosine of the solar zenith angle $\theta_{\text{sun}}$. In addition to the direct form of solar irradiance, Diffuse Horizontal Irradiance (DHI) accounts for power from incoming diffuse light scattered

by the atmosphere. It is measured on a horizontal surface excluding the direct light emission of the solar disk. The sum of the above types of irradiance yields the Global Horizontal Irradiance (GHI) as described below [17, 18].

$$GHI = DHI + DNI \times cos(\theta_{\text{sun}})$$

The effectiveness of solar-based power generation depends on the type of irradiance. PV converts light into electricity by the photoelectric effect. The amount of electricity is influenced by the availability of solar radiation of any type (direct or diffuse). In contrast, thermal CSP mainly utilizes direct irradiance as it requires wavelengths in the infra-red spectrum for generating heat [7].

## 2.2. Clouds

Although clouds appear in infinite varieties and evolving shapes, certain characteristics can frequently be observed. Researchers use these features to group clouds into genera. Overall there exist 10 different genera (see Figure 2) which descriptions are as follow according to the World Meteorological Organization [19].

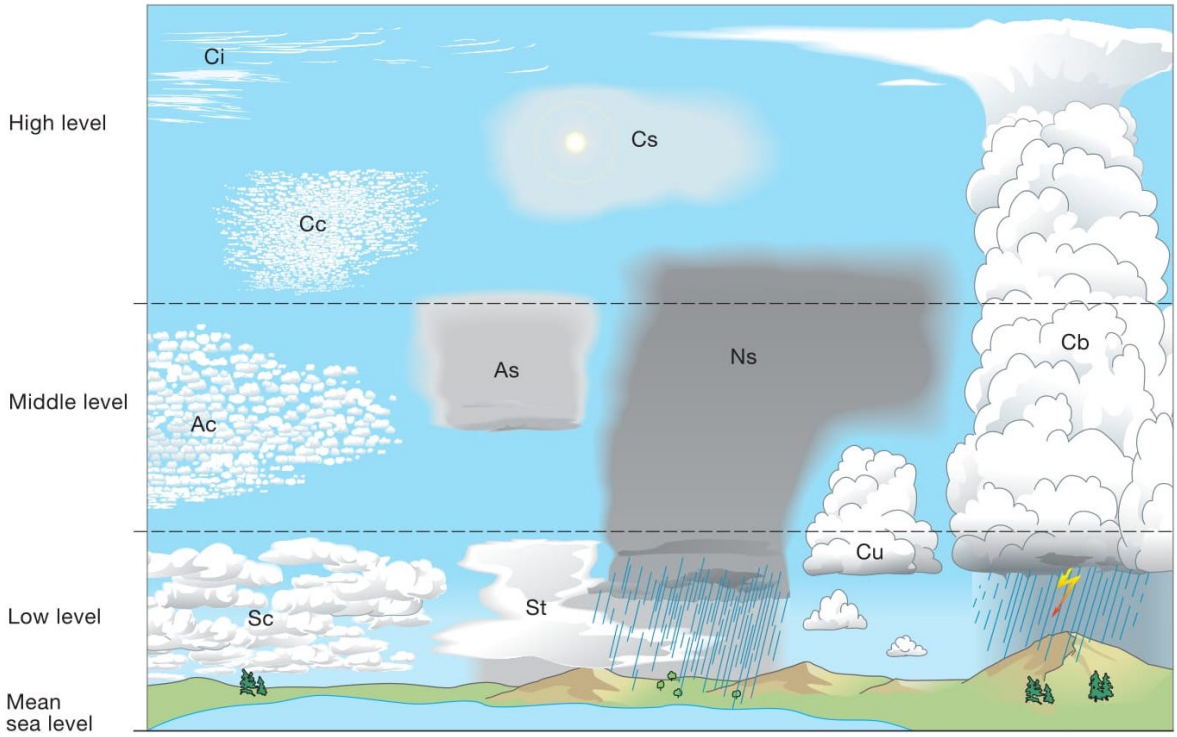



Figure 2: The 10 genera within their allocated levels [20].

**Cirrus (Ci)** "Detached clouds in the form of white, delicate filaments or white or mostly white patches or narrow bands. These clouds have a fibrous (hair-like) appearance, or a silky sheen, or both."

**Cirrocumulus (Cc)** "Thin, white patch, sheet or layer of cloud without shading, composed of very small elements in the form of grains, ripples, etc., merged or separate, and more or less regularly arranged; most of the elements have an apparent width of less than 1°."

**Cirrostratus (Cs)** "Transparent, whitish cloud veil of fibrous (hair-like) or smooth appearance, totally or partly covering the sky, often producing halo phenomena."

**Altocumulus (Ac)** "White or grey, or both white and grey, patch, sheet or layer of cloud, generally with shading, composed of laminae, rounded masses, rolls, etc., which are sometimes partly fibrous or diffuse and which may or may not be merged; most of the regularly arranged small elements usually have an apparent width between 1° and 5°."

**Altostratus (As)** "Greyish or bluish cloud sheet or layer of striated, fibrous or uniform appearance, totally or partly covering the sky, and having parts thin enough to reveal the Sun at least vaguely, as through ground glass. Altostratus does not show halo phenomena."

**Nimbostratus (Ns)** "Grey cloud layer, often dark, the appearance of which is rendered diffuse by more or less continuously falling rain or snow, which, in most cases, reaches the ground. It is thick enough throughout to blot out the Sun. Low, ragged clouds frequently occur below the layer, with which they may or may not merge."

**Stratocumulus (Sc)** "Grey or whitish, or both grey and whitish, patch, sheet or layer of cloud that almost always has dark parts, composed of tessellations, rounded masses, rolls, etc., which are non-fibrous (except for virga) and which may or may not be merged; most of the regularly arranged small elements have an apparent width of more than 5°."

**Stratus (St)** "Generally grey cloud layer with a fairly uniform base, which may give drizzle, snow or snow grains. When the Sun is visible through the cloud, its outline is clearly discernible. Stratus does not produce halo phenomena except, possibly, at very low temperatures. Sometimes Stratus appears in the form of ragged patches."

**Cumulus (Cu)** "Detached clouds, generally dense and with sharp outlines, developing vertically in the form of rising mounds, domes or towers, of which the bulging upper part often resembles a cauliflower. The sunlit parts of these clouds are mostly brilliant

white; their base is relatively dark and nearly horizontal. Sometimes, Cumulus is ragged."

**Cumulunimbus (Cb)** "Heavy and dense cloud, with a considerable vertical extent, in the form of a mountain or huge towers. At least part of its upper portion is usually smooth, or fibrous or striated, and nearly always flattened; this part often spreads out in the shape of an anvil or vast plume. Under the base of this cloud, which is often very dark, there are frequently low, ragged clouds, either merged with it or not, and precipitation sometimes in the form of virga."

Certainly, not all genera are equally important as their impact on GHI/DNI is either not significant or too intense making nowcasting unnecessary. A study conducted by Matuszko [21] on cloud types and the effect on solar irradiance showed that Ci, Cu, Ac, and Cb cause the most variances throughout a day making them of special interest for nowcasting.

## 2.3. Neuronal Networks

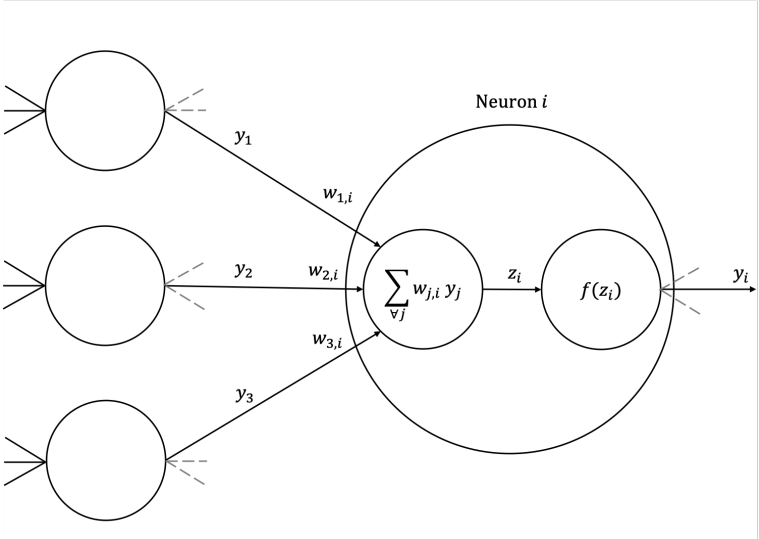



Figure 3: Standard structure of a single neuron with inputs from a previous layer of neurons.

The idea of a trainable multi-layer perceptron or neural network dates back to the late 1950s and is nowadays a key technology for complex problem solving tasks [22]. A standard NN consists of layers of so called neurons. Each neuron produces a real-valued output $y$ often referred as activation which is passed to all neurons of the subsequent layer. Input neurons receive activation via the input while inner neurons get it by weighted connections from previous neurons. The activation of a neuron is computed via a non-linear activation function $f$ e.g. $tanh$ on the sum of all incoming connections

(see Figure 3). Learning happens by adjusting these weights in a way that the entire network exhibits desired behavior, such as recognizing faces in images. The behavior is expressed by a loss function $\mathcal{L}$ which penalizes incorrect outputs. The loss function allows to compute the gradient with respect to the parameters (weights) $W$ of the network which minimizes the function for the dataset. Gradient-based learning can be solved analytically (due to a closed-form solution of the loss) but must usually be performed iteratively as denoted in (2.1). The gradient updates the parameters weighted by a constant learning rate $\eta$ which guides the magnitude of the update. A popular minimization procedure is to frequently perform parameter updates after processing a small batch of the dataset rather than processing the entire dataset before adjusting the parameters.

$$W_k = W_{k-1} - \eta \frac{\partial \mathcal{L}(W)}{\partial W} \tag{2.1}$$

In a regular feed-forward NN, the activation flows from input to output layer in an acyclic manner. If given a task which requires processing a sequence of inputs, this network will process each part of the sequence separately without making links between single elements. In recurrent neural networks (RNN) activations flow cyclic through the network and allow to change input activations over time while previous activations are *memorized* via the cyclic connection (see Figure 4). By that RNNs are able to build up long temporal dependencies and are more powerful in processing sequential patterns than feed-forward NNs [22]. A prominent representative of RNNs is the Long Short-term Memory (LSTM) model introduced by Hochreiter et al. [23].

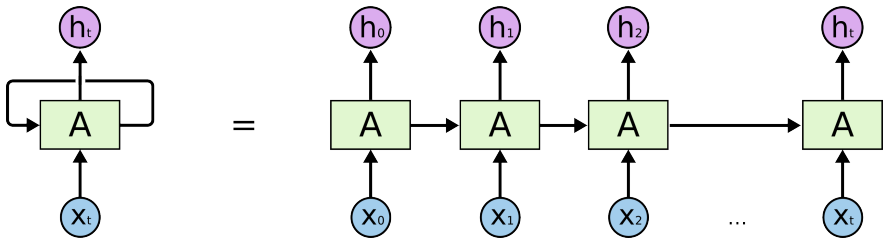



Figure 4: Unfolded RNN. The input is changed at each fold while previous inputs are *memorized* through the cyclic connection [24].

A second specialized type of NNs are convolutional neural networks. Although, their application is not limited to the domain of computer vision, CNNs are mostly known for image-based tasks. The secret behind the success is the hierarchical interpretation of image feature which is very similar to how visual perception is structured in the human brain. Instead of recognizing entire objects at once, *lower* brain layers only react

on simple edges. Such low abstracted visual features are combined to primitive shapes in *higher* layers until reaching the highest level of visual abstraction which is capable of recognizing an entire object model. CNNs realize this by the use of so called filters or kernels. A filter is a square matrix containing a set of trainable weights. Filter size refers to the shape of the matrix and basically determines the number of weights and the amount of convolved input pixel. A single convolution step produces an activation for the corresponding pixel block (see Figure 5). The convolved feature map serves as input to higher convolutional layers which work after the same principle. A layer usually comprises multiple filter which is often referred as layer depth. Learning happens by adjusting the weights of each filter in a way that it reacts to certain patterns like edges. The offset of a filter between two convolution steps is called stride and mainly determines the dimensions of the final convolved feature map. This output dimensions are usually lower than the input dimensions. In many situations it is relevant to maintain the dimensions of an input. This is accomplished by padding the input e.g. with zeros so that the resulting feature map has the same shape as the original input before padding. With this method the shape of the output can be even increased compared to the input which is often referred as deconvolution. A common concept which combines, but is not limited to these two layer types, is the encoder-decoder architecture. In this, the encoder is used to learn an efficient, low-dimensional representation of the input while the decoder learns to map this representation back to the original format with the desired content.

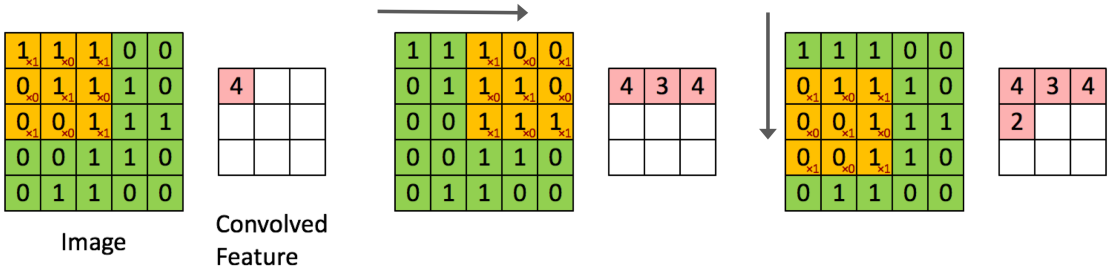



Figure 5: Convolving a filter (orange) of size $3 \times 3$ and stride 1 with a single channel image (green). For demonstration, the filter holds arbitrary weights denoted in red within the filter patch.

# 3. Related Work

For the task of intra-hour irradiance nowcasting, the majority of proposed models use cloud images received from TSI systems. Common steps in this approach are cloud pixel classification, cloud motion vector estimation and irradiance prediction.

In order to separate cloud pixels from sky pixels, thresholding is a prevalently used method [11, 12, 13, 25, 26, 27, 28]. There exist several types while each focuses on the fact that cloud pixels have a higher red color intensity than sky pixels. Fixed threshold methods define fixed values for the red-blue ratio, red-blue difference or multi-color criterions [29]. Li et al. [30] report good performance on clear and overcast images but attest that this method is incapable of detecting cirriform clouds. In turn, Blanc et al. [13] achieve classification errors between 6-17% via multi-color criterions during various cloud conditions and solar zenith angles. Minimum cross entropy is an adaptive thresholding technique for unsupervised classification [31]. It was shown that this method outperforms fixed threshold methods for cumuliform and cirriform clouds. However, both methods are highly sensitive to sun glare leading to significant portions of false positive pixels in the circumsolar region [30]. This is partly circumvented by the preliminary use of the clear-sky library method. This method consists of a collection of clear sky images at different solar zenith angles capturing the variations in the red-blue ratio introduced by the sun position [12, 25, 28, 32]. Chu et al. [28] report only mixed performance for this approach as it globally affects red-blue ratio which leads to an increase of false negatives depending on occlusion of the solar disk. Moreover, the clear-sky library method does not consider second-order effects on red-blue ratio like aerosol [28]. The majority of authors state that misclassified pixels greatly contribute to the overall model error, which opens the discussion for a different approach for cloud pixel classification.

The next step is cloud motion vector prediction. A simple but effective method is to cross-correlate pixel subsets on consecutive images. The distance between two matching pixel patches is aggregated to an average wind speed of the patch. This enables the computation of grid fractions which may shade the sun in the near future. One drawback of this method is the resulting sparsity as each aggregation implies homogenous velocities of the corresponding patch [11, 12]. Another approach is particle image velocimetry resulting in a set of velocity vectors. While this method captures for heterogenous velocities, it is unable to differentiate trajectories of multi-layer clouds [26, 28]. Similar applies for sparse optical flow methods which refer to this as the object occlusion problem [25]. A more sophisticated method was proposed by Blanc et al. [13]. The authors use stereoscopic photogrammetry with two TSI systems to estimate the cloud base height allowing them to compute three-dimensional (3D) cloud motion vectors via cross-correlation per cloud base height layer. Although, the overall model performance was mixed, the authors emphasise the benefits of including depth information for motion estimation and address the complexity of multi-layer cloud dynamics. Comparable approaches which utilise depth information were used by the groups of

Huang et al. [33] and Peng et al. [34] yielding an improvement of 28% in motion prediction and 26% respectively in the overall model performance compared to their baseline.

The last step aggregates all data into an irradiance prediction. All models so far predict minute-wise averages of irradiance by various means. A very straightforward way is to combine classified cloud pixel to a two-dimensional (2D) cloud map with a homogenous cloud base height. The binary occlusion state in the region of interest is calculated by a simple planar projection of the map to the surface using the current solar zenith angle [26]. More versatile approaches account for different irradiance/cloud types, heterogenous cloud base height and even atmospheric conditions. A common way is to describe irradiance through the clear-sky index capturing the ratio between achievable irradiance at clear sky conditions and actual measured irradiance. The clear sky irradiance is retrieved according to meteorological models [35]. While some authors assign fixed values to a occlusion state [11], others follow a more adaptive approach by tracking this index during a preliminary time span. Significant peaks in the histogram are then assigned to corresponding occlusion states [12, 25] or classified cloud types [13]. Regression models are also used although they are more applicable in estimation of irradiance at a single spot [34]. One significant aspect affecting the estimation is optical perspective. Many TSI systems use fish-eye cameras resulting in strong distortions close to the edge of the view field. Schmidt et al. [25] report that distant cumuliform clouds with a strong vertical extent were often incorrectly projected to long horizontally covers leading to significantly underestimated gaps in the cloud map.

Coming to NN-based approaches, one will find that the body of work applying end-to-end NNs for spatially comprehensive solar irradiance nowcasting is rather sparse. In [36] TSI images are used to estimate current single point GHI. The authors computed pixel clusters as features for a fully connected NN. Similar, Sun et al. [37] developed a CNN architecture which estimates the current power output of a PV plant based on video streams. Although, these models neither nowcast irradiance nor provide spatially comprehensive outputs, the authors showed that NNs are capable of regressing irradiance from TSI data. A more versatile model was proposed by Ryu et al. [38]. The authors nowcast single spot GHI up to 20 minutes ahead from TSI and lagged GHI data. However, the model was not able to significantly outperform the baseline [38].

More work is available by looking into individual steps mentioned in the beginning of this chapter. Tulpan et al. [39] compare learning techniques for binary cloud pixel classification on pre-computed textural features. The features are extracted via various thresholding methods. The NN method achieves competitive error rates of 7%. Dev et al. [40] recently proposed a CNN-based binary pixel classifier for day and night images. The method works without manual feature selection as in [39] and achieves similar error rates for daytime images. The same authors proposed in a very recent pre-print submission a model for multi-class cloud pixel classification based on a CNN

architecture from biomedical image processing. Preliminary evaluation on a very small dataset shows promising class-dependent error rates between 4-7% [41]. However, these error rates must be interpreted with care as non of the used datasets include images displaying clouds in the circumsolar region.

In literature there exist no NN-based framework specifically for motion estimation of clouds. Consequently, this paragraph summarizes general approaches of NN-based trajectory estimation. Recent work put increasing efforts into dense optical flow estimation with deep CNN. Notable contributions are *EpicFlow* [42] and *FlowNet2* [43] which allow pixel-wise trajectory estimation without aggregation per pixel patch. However, these models are trained in a purely supervised fashion and labeled data is expensive. This encouraged researchers to also investigate on unsupervised NN or self-supervised models. Yin et al. [44] recently proposed *GeoNet*, a self-supervised deep CNN model which jointly estimates dense optical flow, scene depth and camera pose from monocular videos. The authors achieve competitive results compared to supervised, state-of-the-art methods. Another way to account for cloud motion is to predict future sky images from a past image sequence. Although, CNNs can be used for this task, the performance is often limited due to input size restriction or a missing memory capability [45, 46]. Here, RNNs work far better for the prediction of spatiotemporal features and are widely used for image or video frame prediction. Inspired by *FlowNet2*, Patraucean et al. [47] combine LSTMs and optical flow for pixel-wise video frame prediction. Xingjian et al. [48] proposed convolutional LSTMs (ConvLSTM) which combine convolutional operations with recurrent connections. This model was also starting point of some mentionable extensions [49, 50]. More recently, Wang et al. [51] published a specialized LSTM design called Causal LSTM (CausalLSTM) which allow to model very long spatiotemporal dependencies.

The prediction of irradiance from pictures is split into planar projection of clouds and irradiance estimation. Admittedly, the body of work specifically addressing this challenges for NNs is also limited. This changes if looking into NN-based approaches for the more general task of planar object projection. A projection from 3D to 2D is mathematically speaking an easy task once the 3D coordinates and the viewpoint parameters are known. Wu et al. [52] employ a differentiable projection layer which is capable to learn external viewpoint parameters from images and projects 3D coordinates accordingly. However, in the usual case 3D coordinates are not available and many models need to derived such from 2D images beforehand. A common approach in 3D view synthesis is to jointly train: perspective-invariant feature representations from 2D images, affine transformations in 3D space and reprojection to 2D [53, 54, 55, 56, 57]. A more flexible approach for planar image projection is NN-based point cloud prediction or simply depth estimation. In this method the relative distance between camera and objects in an image is estimated per pixel. The resulting depth map can then be used in a subsequent model to compute a planar projection of the scene. In contrast to analytical methods like photogrammetry, models in this domain do not require specially calibrated camera systems.

# 4. Model

A central design goal is to tackle each step with an NN-based approach while keeping the requirements on labeled data to a minimum. The model consists of three conceptual stages. First, predicting a sequence of future cloud images from a sequence of past images (forecasting stage). Second, extract scene depth features from image features (extraction stage). Third, estimate an areal shadow map for each image (projection stage).

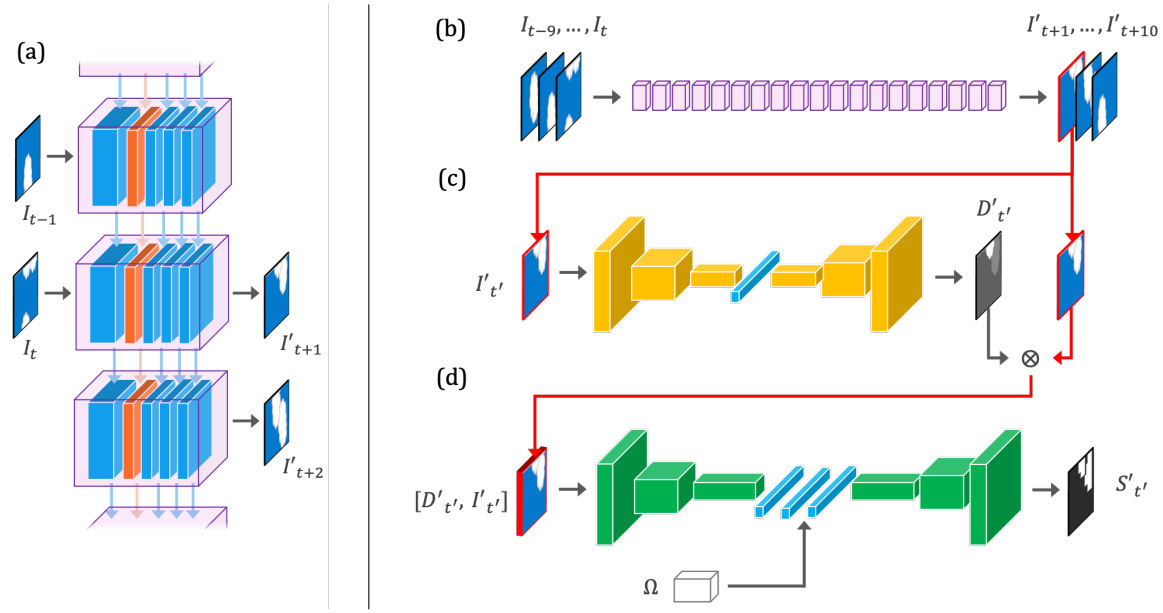



Figure 6: (a) A part of the forecasting stage showing single CausalLSTM cells and how temporal information is propagated at different abstraction levels. (b) The model of the forecasting stage predicting a sequence of future images $I'_{t+1}, \ldots, I'_{t+10}$ from a preliminary image sequence $I_{t-9}, \ldots, I_t$. (c) Extraction stage applied on each image of the predicted sequence. The resulting depth map is stacked channel-wise with the corresponding image. (d) Projection stage applied on each stacked image prediction yielding a shadow map $S'_{t'}$.

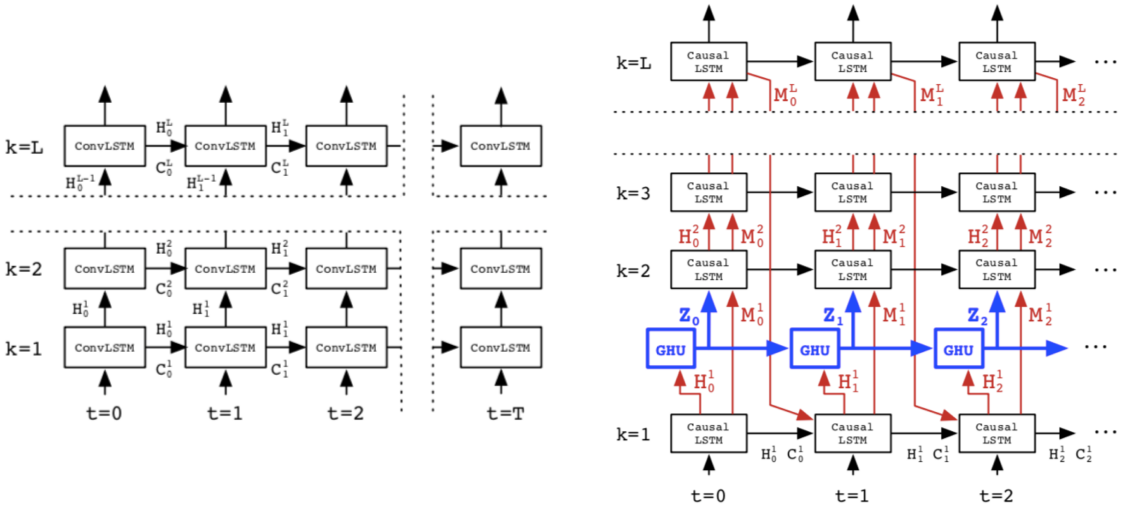
The goal of this three-staged architecture is a separation of the overall problem into smaller sub-problems with minimal learning dependencies on each other. For example, the estimation of scene depth is an enclosed problem which is not dependent on the input being a real or a synthesized image. Evaluation of a single stage can therefore take place without error accumulation. Another argument is that the network used in each step solves a more clear problem resulting in less complex learning objectives. Each neural network is specific to the problem of its stage and is individually trained.

In detail, the forecasting stage gets a sequence of images $I_{t-9}, \ldots, I_t$ to predict a sequence of future images $I'_{t+1}, \ldots, I'_{t+10}$ (see Figure 6b). Next, depth features $D'_{t'}$ are extracted per predicted image $I'_{t'}$ (see Figure 6c). This depth map is stacked channel-

wise with the original image and is passed to the projection stage. The projection network then estimates the shadow map $S'_{t'}$ (see Figure 6d) where the environmental parameters $\Omega$ contain solar position angels.

## 4.1. Forecasting Stage

Although current research offers many viable techniques for motion prediction (see Section 3), the decision to predict future image features rather than to extract features to perform temporal regression, comes with advantages. Raw image features offer the most flexibility to subsequent learning methods. Neither this network nor the network of the next stage is interdependent. Thus, the remainder of the pipeline is unrestricted in choice of the learning method.




(a) Convolutional LSTM (b) CausalLSTM

Figure 7: Overview of information flows in (a) conventional design of stacked convolutional layers with temporal transitions between each abstraction level (horizontal black arrows), (b) CausalLSTM design with deep transitions (red arrows) and GHU layer (blue) [51].

A central aspect of the network architecture in this step is the use of CausalLSTM layers which are a specialized variant of a convolutional LSTM layer. This unit has much deeper transition paths for spatiotemporal information flow compared to conventional RNN designs. The network can therefore maintain much longer temporal dependencies compared to other models. These transitions also traverse the network in both, spatial and temporal dimension to allow information flow through all spatial and temporal abstraction levels. In contrast, the information flow in a ConvLSTM design is propagated horizontally per abstraction level which may inhibits learning capabilities (see Figure

7) [51]. Deep temporal transitions often cause a vanishing gradient which hinders the network to learn long temporal dependencies [58]. The authors developed a Gradient Highway Unit (GHU) to by-pass the gradient to deep layers during back-propagation which makes this architecture superior to other designs [51]. The network consists of 19 cells of which the first 10 cells read the input sequence and the next 9 predict the future images from the inner state. For each cell, the design of the original paper with 4 CausalLSTM layers and a GHU between the first and the second CausalLSTM layer was kept (see Figure 6a). The filter size was increased from 5 to 9 compared to the original paper as the size of the input also increased ($64px$ to $256px$). Larger filter sizes could not be tested as the model size exceeded the available hardware resources. However, a large receptive field has a strong abstraction effect on spatial features and can even be disadvantageous for the task. A large filter size result in over-smoothing of fine structures specifically of small or far distant clouds spreading only a couple of pixels in the input image. A summary of further layer parameters is given in Table 1. As objective function, the authors use the sum of $L1$ and $L2$ loss of each predicted frame which they argue gives a better balance between sharpness and smoothness of generated images [51]. Since the used training data shows no fine texture, the $L1$ term enforcing image sharpness is omitted resulting in the objective function given in (4.1). Moreover, the authors employ batch normalization inside CausalLSTM layers to normalize incoming activations. In this model the batch normalization is disabled as it showed a significant slowing effect on training without yielding better generalization.

$$\mathcal{L} = \mathcal{L}_{L2} = \frac{1}{2} \sum_{\Delta t} (I_{t+\Delta t} - I'_{t+\Delta t})^2 \tag{4.1}$$

| | **Height** | **Width** | **Depth** | **Filter Size** | **Number Filter** | **Stride** |
|---|---|---|---|---|---|---|
| *Output* | 256 | 256 | 3 | 1 | 3 | 1 |
| *CausalLSTM-4* | 256 | 256 | 64 | 9 | 64 | 1 |
| *CausalLSTM-3* | 256 | 256 | 64 | 9 | 64 | 1 |
| *CausalLSTM-2* | 256 | 256 | 64 | 9 | 64 | 1 |
| *GHU-1* | 256 | 256 | 128 | - | - | 1 |
| *CausalLSTM-1* | 256 | 256 | 128 | 9 | 128 | 1 |
| *Input* | 256 | 256 | 3 | - | - | - |

Table 1: Layer configuration of single CausalLSTM cell.

## 4.2. Extraction Stage

Neural networks are very capable in detecting structural patterns in raw data and process such to solve classification or regression tasks. One advantage is that feature selection is jointly trained making manual feature selection obsolete in the general case. Consequently, asking about the necessity of this extra step to extract depth features is a legitimate question. In order to predict shadows within an area of interest based on a sky image, a network must be first of all able to learn which area in the input image is relevant. The area of interest can be imaged as a window on the input image in which clouds are projected. The position in the image is determined explicitly by the solar angles, while its dimensions additionally depend on the cloud base height. In contrast to the solar angles, this information is only given implicitly through the ground truth and requires the network to spatially link shadows to clouds in the input image. This is much more complex and also comes with ambiguity. Cloud base height has a significant impact on the range in which a shadow is casted especially at high solar zenith angles ($> 70°$). Small differences in altitude ($\sim 500m$) which do not significantly alter the input image can cause a very different shadow map. If a cloud additionally changes in size this may even hold for lower zenith angles or larger altitude changes. The main motivation behind enriching the visual input with depth features is to remove such ambiguity. It also helps the network to weight relevant areas in the image more accurately by the explicit, spatial connection of depth and image feature.

Due to the absence of ground truth depth data, its estimation must take place in a self-supervised manner. Additionally, the input data origins from a single camera which means that the model must find other means than e.g. multi-scopy. A common approach in self-supervised monocular depth estimation is to use adjacent frames in a video signal to hallucinate depth as an intermediate variable to reconstruct the middle frame from adjacent source frames. The reconstruction error serves as supervisory signal. Training is performed on image sequences of uneven length to compute the reconstruction error while testing happens on a single image. There exist two essential challenges in reconstruction-based monocular depth estimation namely camera ego-motion and object occlusion. In case of ego-motion the camera moves at a similar speed as other moving object in the scene. The object will appear as rigid part in the scene which leads to holes of infinite depth. Same applies in case of a static camera and moving, low textured objects i.e. clouds which results in no changes for certain pixel between adjacent frames [59]. In object occlusion, a source pixel being occulted in the target image (or vice versa) causes a high reconstruction penalty even if its depth was correctly estimated. Many recent proposal in this domain have addressed this challenges by more sophisticated architectures or objective functions. However, choosing the best model for estimating cloud base height is not as straightforward as one may think. Most of these models were only trained and tested on the elaborated KITTI dataset [60] containing street sceneries for autonomous driving. Good performance may be very specific to the dataset and does not hold for a novel dataset. For this analysis, two models are chosen and evaluated: *GeoNet* [44] and *MonoDepth2* [59].

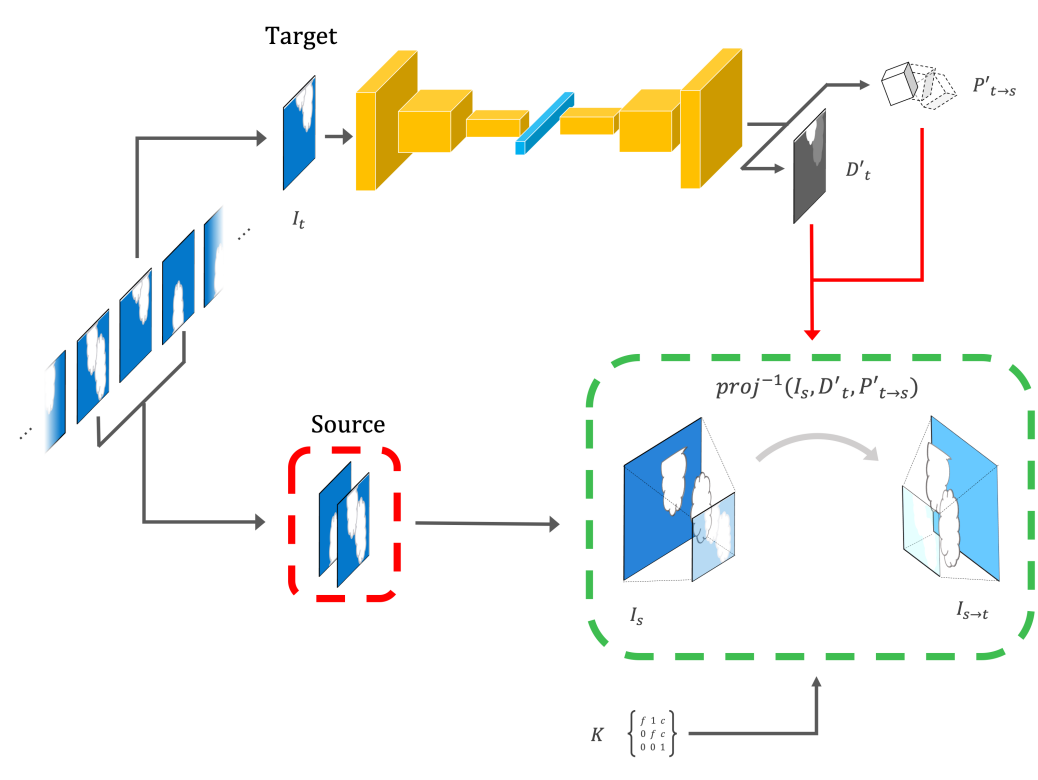



Figure 8: Overview of reconstruction-based training. The network estimates depth $D'_t$ and 6-axis pose change $P'_{t\to s}$ between target frame and source frame. Subsequently, each source frame is reprojected by inverse warping to the target plane yielding $I_{s\to t}$. $K$ denotes a matrix of intrinsic camera parameters describing focal length and sensor resolution.

*GeoNet* was originally designed for optical flow estimation but the authors use a masking mechanism based on scene depth for the task. This depth feature is acquired in a separate step by a dedicated neural network which can also be used autonomously. The architecture of the network is adapted from the proven *ResNet-50* network [61]. Since the authors only modify the loss function of the network according to the new training objective, *GeoNet* serves as a good baseline model. The network predicts a depth map $D'_t$ and a pose change vector $P'_{t\to s}$ which is used to project a source frame $I_s$ on the target frame $I_t$. The depth map is comparable to a single channel image and encodes a normalized, relative distance of each pixel to the viewpoint. The pose describes a pose change of the viewpoint between two frames in form of translation and rotation. In the application of solar irradiance nowcasting the camera is static. In this case the pose can be interpreted as the movement of the scenery described as movement of the viewpoint. Depth and pose are estimated by separate networks which are depicted in detail in the appendix (Figure A.1 and Figure A.3).

A predicted depth map must minimize the two folded loss function (4.2) for all source to target pairs. The first term is the photo-metric loss $\mathcal{L}_{\text{pm}}$ (4.3) between the projected source to target frame $I_{s\to t}$ and the target frame $I_t$. The loss is based on the structural similarity index $SSIM$ [62] and a parameter $\alpha = 0.85$ which has been determined by cross-validation. The second term is a depth smoothness loss $\mathcal{L}_{\text{ds}}$ (4.4) enforcing smoothness of the depth map by image gradient weighting. The directional color gradient is denoted by the operator $\partial_x$ and $\partial_y$ respectively [44].

$$I_{s\to t} = proj^{-1}(I_s, D'_t, P'_{t\to s})$$

$$\mathcal{L} = \frac{1}{n}\sum_{I_{s\to t}} \mathcal{L}_{\text{pm}} + \lambda_{\text{ds}}\mathcal{L}_{\text{ds}} \tag{4.2}$$

$$\mathcal{L}_{\text{pm}} = \alpha\frac{1 - SSIM(I_t, I_{s\to t})}{2} + (1-\alpha)\|I_t - I_{s\to t}\|_1 \tag{4.3}$$

$$\mathcal{L}_{\text{ds}} = |\partial_x D'_t|e^{-|\partial_x I_t|} + |\partial_y D'_t|e^{-|\partial_y I_t|} \tag{4.4}$$

The *MonoDepth2* model builds on the same method as in *GeoNet*. However, this network is solely designed for depth estimation. The architecture is inspired by *ResNet-18*, a smaller variant of *ResNet-50*. A detailed illustration can be found in the appendix (Figure A.4 and Figure A.6). In contrast to *GeoNet*, the authors of *MonoDepth2* implemented two countermeasures to the previously mentioned challenges of reconstruction-based monocular depth estimation. First, instead of calculating the average photo-metric loss $\mathcal{L}_{\text{pm}}$ among all source to target pairs, the minimum is used. By doing so a high reconstruction penalty for occulted pixels is avoided if the pixel is visible in one of the remaining frames. Second, the authors mask stationary pixel in adjacent frames to prevent these pixels to contaminate the loss. This mask is computed in the forward-pass based on the reprojection error and purges the loss in the backward-pass for such stationary pixel. Generally, this architecture can be considered state-of-the-art in depth estimation and the authors report a superior performance compared to *GeoNet* on the KITTI dataset. The loss function is the same as in (4.2) with the mentioned modification to the photo-metric loss term and a replacement of $D'_t$ in the $\mathcal{L}_{\text{ds}}$ term by a mean-normalized inverse variant $D^*_t = D'_t/\mu_{D'_t}$ [59].

## 4.3. Projection Stage

So far, the pipeline synthesizes future images features $I'$ from past image features and a subsequent feature extractor network predicts corresponding depth features $D'$.

Now, in order to conclude cloud shadows $S'$, the network in this stage must learn an perspective-invariant representation of the input features and the projective function to map this representation to 2D for various projection angles i.e. sun positions. Additionally, the network has to perform a segmentation according to which pixel in the image features actually casts a shadow. There exists no work in the domain of neural projection networks which exactly addresses all the mentioned challenges at once. However, there are parallels to the work of Tatarchenko et al. [55] which makes it a good candidate for this stage. The authors aim to synthesize an unseen view of an object after rotating it in 3D space. The model acquires 3D features from a single input image, performs a rotation and predicts the image projection in the new pose to the same viewpoint. The only differences in cloud shadow prediction are that the viewpoint is rotated instead of the object and the field of view is reduced. Moreover, the authors use unsegmented images of objects as input which is also the case for cloud images.

The original encoder-decoder design does not foresees the input of additional depth features. Since depth maps have similar characteristics as images, a straightforward approach would be to duplicate the encoder part of the network for this second input. Both input layers would then be gradually fused together during the course of the network. There exist a variety of fusion techniques and elaborating them all is a topic on its own. Karpathy et al. [63] give a good overview of different fusion techniques and their performance on multi-frame video classification. However, for the model of this stage channel-stacking is used. In this, the single channeled depth map is appended to the input image as a second (in grayscale images) or fourth (in color images) channel. One motivation behind is model complexity. While the design comprises $52M$ trainable parameters (Forecasting stage: $27M$, Extraction stage: $60M$), extending the encoder part possibly increases this number. A second aspect is the spatial correlation of depth features and image features. Separating the inputs into different encoder without sharing weights makes it harder to maintain this connection.

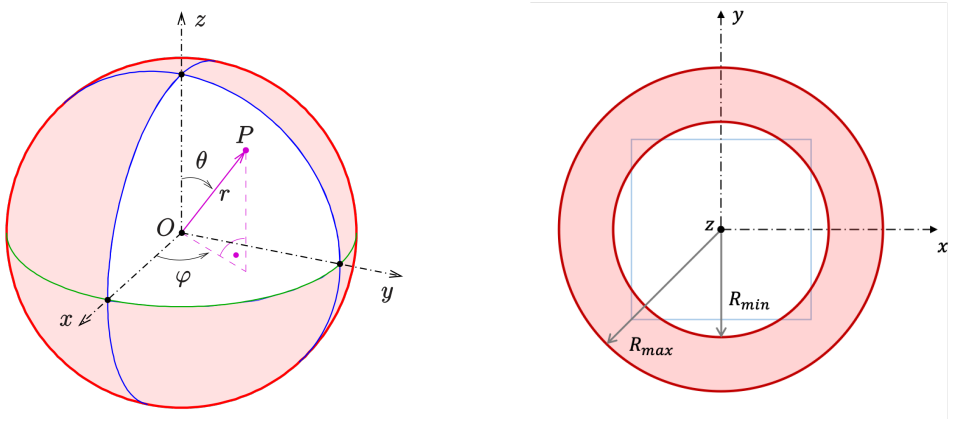
Additionally to the stacked image and depth features $[D', I']$, a vector $\Omega$ of environmental parameters is fed to the network. As depicted in Figure 6d it is given as intermediate input containing sine and cosine of the solar zenith angle $\theta_{\text{sun}}$. The trigonometric operation account for periodicity as well as normalization. The total loss (4.5) is pixel-mean squared error of the predicted shadow map $S'$ and ground truth $S$. A $L2$ regularization on the fully connected layers with weight $\lambda_{L2} = 0.0001$ showed improved performance. A detailed overview of the network architecture is given in the appendix (see Figure A.7).

$$\mathcal{L} = \frac{1}{n}(S - S')^2 \tag{4.5}$$

# 5. Evaluation

## 5.1. Simulation

The data used for the evaluation is computer-generated by a custom simulation software. The software simulates a virtual environment with sun, clouds and wind effects to output a 360° fish-eye camera image of the sky, a corresponding depth map encoding distances to the viewpoint, a shadow map of the area around the camera and a vector encoding the sun position (see Figure 12 for example output).




(a) (b)

Figure 9: (a) Definitions of translation and rotation origin [64]. (b) View of $xy$ plane displaying placement interval $R_{min}$, $R_{max}$ of cloud centers.

The position of the sun is described by azimuth angle $\varphi_{\text{sun}}$ and zenith angle $\theta_{\text{sun}}$. The environment simulates only horizontal wind with direction given by azimuth angle $\varphi_{\text{wind}}$ and surface wind speed $v_0$ at $10m$ altitude ($h_0 = 10$). The origin of the simulation environment is defined by the camera (see Figure 9a for origin of angles). The placement of cloud center points is described by radius $R$ between $R_{min}$ and $R_{max}$ in the $xy$ plane as depicted in Figure 9b.

The cloud model is described by an outer ellipsoid with widths and thickness which center will be placed in the placement area of the simulation environment. The base height $h$ of the cloud is uniform random between the interval $h_{min}$, $h_{max}$. Afterwards, a random number of points is placed inside the ellipsoid of which each point spans a sphere of random radius (see Figure 10). There are a number of cloud genera which are relevant to irradiance forecasting as described in Section 2.2. The simulation focusses on Cumulus clouds which are a common cloud type in the context of irradiance forecasting. The cloud model parameters are chosen accordingly to mimic this genus of cloud. Based on meteorological observations the outer ellipsoid widths lies between $50 - 250m$ with a thickness between $25 - 50m$. The radius of the inner spheres ranges

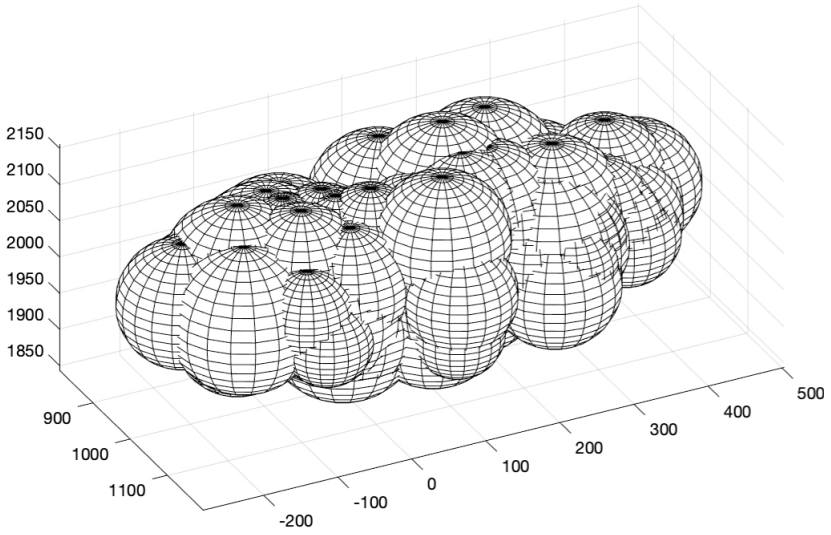



Figure 10: 3D view of a generated cloud.

between $40 - 80m$ while each cloud contains between 10 to 15 spheres. These base values are multiplied by a cloud size factor $\kappa$. The cloud movement is computed frame-wise every $60s$ and translates the center of the outer ellipsoid into direction of $\varphi_{\text{wind}}$ with velocity $v$. The actual velocity follows the wind profile power law (5.1) and is a function of surface wind speed $v_0$ at altitude $h_0$, cloud base height $h$ and $\alpha = 0.143$ being an empirical determined coefficient depending on the stability of the atmosphere [65]. Further properties of clouds (e.g. shape, density) as well as cloud (de-)formation are not considered.

$$v(h) = v_0(\frac{h}{h_0})^{\alpha} \tag{5.1}$$

A fish-eye camera is a complex optical system with multiple lenses. One important property of such cameras is the large field of view reaching up to 280° [66]. However, an image projection with such a large field of view is not possible without distortions. This distortion can be simulated by a fish-eye projection function. While Figure 11 gives an overview of common functions, the simulation uses the equisolid function (5.2). The projection radius $r$ on the image plane is a function of the focal length $f$ in $mm$ and $\theta$ the azimuthal projection angle. The usage of this function is mainly justified by the well described Total Sky Imager systems applied by Yang et al. [12] which mount a Sigma $4.5mm$ $2.8f$ lens with 180° field of view [67]. The sensor produces images at a resolution of $2,048 \times 2,048px$ and has a pixel density of $160.722\ \frac{px}{mm}$.

$$r = 2f \times sin(\frac{1}{2}\theta) \tag{5.2}$$

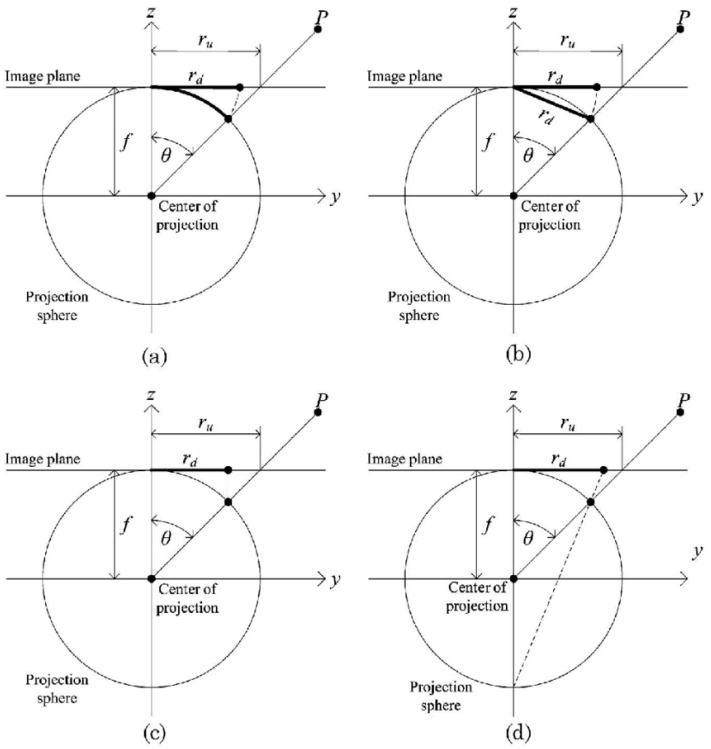



Figure 11: Overview of (a) equidistant, (b) equisolid, (c) orthographic and (d) stereographic fish-eye projection function [68].

The simulation outputs $N_I$ greyscale images ($2,048px \times 2,048px$) of the sky, a normalized depth map and a binary shadow map ($100px \times 100px$) of the area around the camera per $t \in 0...T-1$. The dimensions of the shadow map span an area of $4km \times 4km$ yielding a spatial resolution of $40m \times 40m$ per pixel. This is higher than the objective of this work which is to achieve a spatial resolution of less than $100m^2$ per pixel. The motivation behind the increased area is to raise the probability of having shadows in the output. One can argue that resizing the shape of the shadow map accordingly would be a simple solution to this deviation. However, the input to the model is down-sampled by a factor of 8 to decrease model size and training times. In consequence evaluation results most likely also hold for sky image and shadow map with increased resolution. The depth map is a matrix with the same dimensions as the sky image and contains the absolute distance of each object per pixel. Due to the shape it is encoded as single channel image. This introduces a discretization error since the used format only supports 255 different color levels per pixel. The cloud base height is normalized with $25,500$ which yields a resolution of $100m$ per color level. An in depth analysis about the impact of this error is given in Section 5.6.

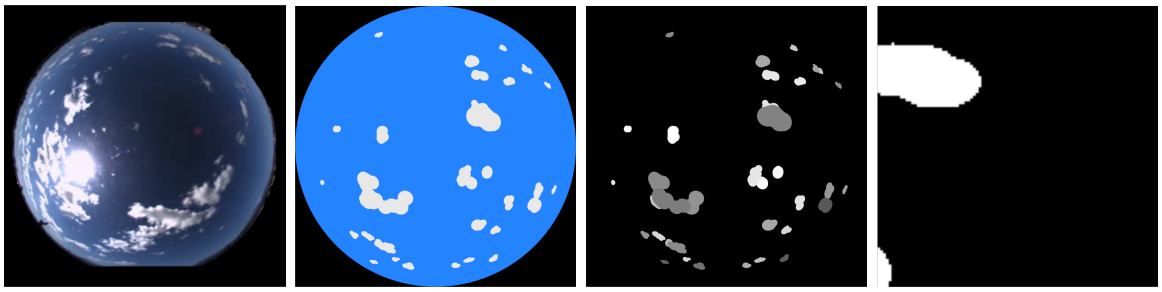
## 5.2. Dataset

For evaluation of the model, three distinctive datasets were generated each comprising a train set and test set. Below there is a short description of the datasets and its purpose while Table 2 lists all simulation parameters.

**Flat dataset** Single frames capturing a moderate number of Cumulus clouds at a static base height of $2,000m$. The sun zenith angle ranges between $9°$ to $81°$ simulating a full day. Since it is a single frame, no movement is applied.

**Depth dataset** Same as the *Flat* dataset but with differing cloud base height $h$ between $2,000m$ to $6,000m$.

**Moving dataset** This dataset consists of images showing a higher density of moving Cumulus clouds at differing altitude. The set captures in total 20 frames per scene at a moderate surface windspeed $v_0 = 2$. The placement radius $R$ is increased to ensure cloud cover even at the last frame of the sequence.



(a) (b) (c) (d)

Figure 12: Examples of (a) a real fish-eye image [27], (b) colorized simulation image, (c) depth map, (d) shadow map

## 5.3. Pre-processing

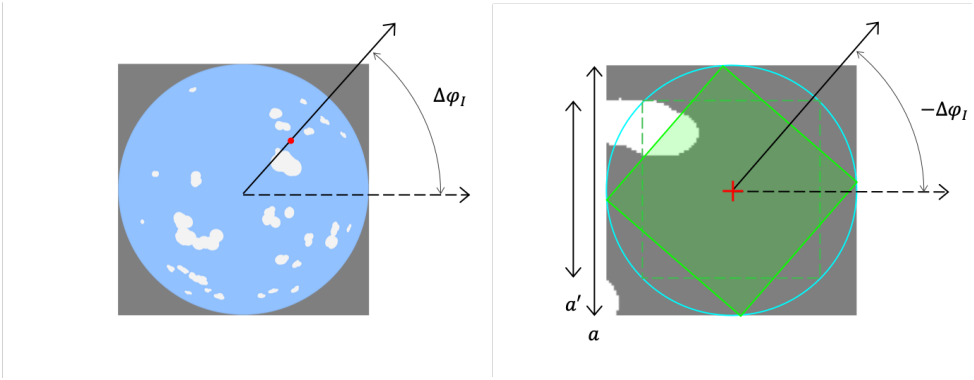
For improving the performance of the model, some pre-processing steps apply. First, the input frames must be normalized in terms of orientation. This means the input image is rotated by $\Delta\varphi_I$ so that $\varphi_{\text{sun}} + \Delta\varphi_I = 0$ holds. Due to the circular image shape, this operation can easily be applied. Consequently, the model predicts a shadow map which is rotated by $\Delta\varphi_I$ relative to the original coordinate system. Since the shadow map is rectangular with width $a$, it cannot be rotated back in the same way. Instead, a smaller rectangle of width $a' = \frac{1}{2}a$ is taken and rotated as depicted in Figure 13. Afterwards the rotated image grid is resampled from the original shadow map. This optimization reduces the data variance resulting in faster training as the model must only be aware of the solar zenith angle. The simulated data is generated by default with $\varphi_{\text{sun}} = 0$ which leaves this pre-processing step to the later system environment.

| | *Parameter* | **Flat** | **Depth** | **Moving** |
|---|---|---|---|---|
| Train | $N_I$ | $20,000$ | $20,000$ | $20,000$ |
| | $N_C$ | 10 / 20 | 10 / 20 | 30 / 40 |
| | $T$ | 1 | 1 | 20 |
| | $R$ | 0 / $15,000$ | 0 / $15,000$ | 0 / $40,000$ |
| | $h$ | $2,000$ | $2,000$ / $6,000$ | $2,000$ / $6,000$ |
| | $\kappa$ | 4 | 4 | 5 |
| | $\theta_{\text{sun}}$ | 9 / 81 | 9 / 81 | 0 |
| | $\varphi_{\text{sun}}$ | 0 | 0 | 0 |
| | $\varphi_{\text{wind}}$ | 0 | 0 | 0 / 90 |
| | $v_0$ | 0 | 0 | 2 |
| Test | $N_I$ | $4,000$ | $2,000$ | $2,000$ |
| | $\varphi_{\text{sun}}$ | | | 9 / 81 |
| | $\varphi_{\text{wind}}$ | | | 0 / 360 |

Table 2: Simulation parameters of all datasets. For test set, only deviating parameters from the train set are given. Intervals are given by upper and lower bound.

A second, important pre-processing step is camera calibration. The general CNN model builds on images from a pinhole camera in which each pixel in the image plane covers an almost equally sized field of view. However, this assumption does not hold for images from fish-eye cameras which follow a spheric projection model with strong distortions towards the edge of the view field. Consequently, the translational weight sharing of rectangular convolutional filters is inefficient for such distorted image features. Increasing the applicability of CNNs to these type of images is an active field of research which has already shed some light on the building blocks of novel CNN architectures addressing this issue [69, 70]. Nevertheless, there exists other means, namely camera calibration, to reduce the impact of distorted image features on the performance of CNNs. Camera calibration assumes an underlying distortion model which allows to reproject points of the image plane without distortion. The intrinsic parameters of such model are often determined algorithmically [71]. In this work, the distortion model is based on Zhang [72] which uses four coefficients to describe the amount of distortion. The implementation is part of OpenCV 3.4 and offers an algorithm to estimate these parameters [73]. This process is performed once on a set of reference images (Figure 14, first row) yielding the following coefficients $k_1 = -0.04180143, k_2 = -0.00074245, k_3 = 0.00168039, k_4 = -0.00061223$. Rectification is applied per image (Figure 14, last row) and is referred as input image in the remainder of this document. By this operation, the effective field of view is reduced to approximately 156° on the main axis.

At last, the forecasting stage includes a channel-wise mean centering. In this, the mean value of among all input images is calculated beforehand. During training, this mean is subtracted from the input data and added on the output prediction. Same applies to testing although it is important to use the mean of the training set not the test set. Without this procedure, the network of the forecasting stage was not able to separate moving objects in the scenery resulting in blank predictions. The authors of the network did not mention such procedure in their work.




(a) Input image (b) Shadow map

Figure 13: (a) Shows how $\Delta\varphi_I$ is derived from an input image with exemplary sun position (red dot). (b) Displays inverse rotation of smaller rectangular area (green) inside a predicted shadow map.

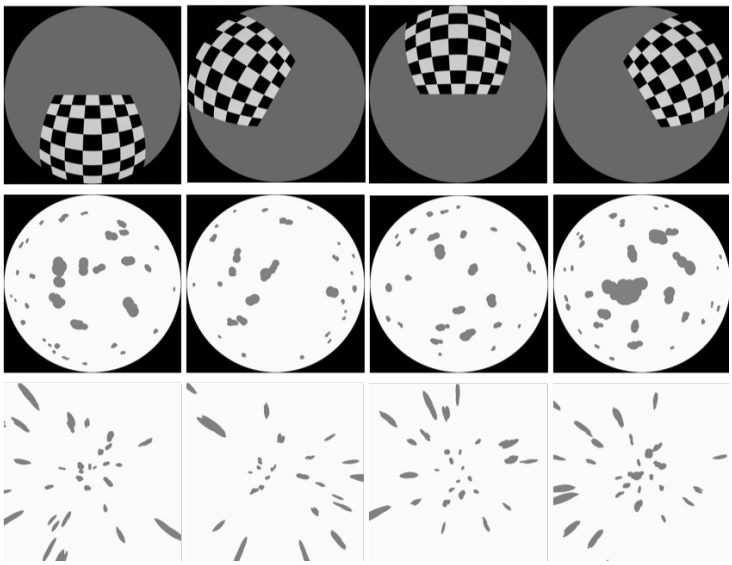


Figure 14: Different stages during camera calibration. First row, examples of calibration images. Second row, input image before rectification. Third row, corresponding image after rectification.

## 5.4. Error Metrics

This section introduces the error metrics used to asses the quality of the models predictions. Two central error metrics for image comparison are mean-squared-error $E_{\text{mse}}$ and root-mean-square-error $E_{\text{rmse}}$ where $x$ denotes ground truth, $x'$ denotes prediction and $N$ is the number of samples i.e. pixel.

$$E_{\text{mse}} = \frac{1}{N} \sum_{i}^{N} (x_i - x_i')^2 \tag{5.3}$$

$$E_{\text{rmse}} = \sqrt{\frac{1}{N} \sum_{i}^{N} (x_i - x_i')^2} \tag{5.4}$$

Comparisons between solar forecasting models is not a simple task as evaluation data, especially real world data, can vary drastically in its complexity. Thus, error rates often reported in root-mean-squared error or correlation values are biased and inappropriate if the underlying datasets differ. A much more meaningful metric for comparisons is forecast skill $FS$ which is the ratio of error metric $E$ between a forecast model $f(x)$ and a baseline $f'(x)$ [74]. In this way, performance results are much less affected by differing solar meteorology in the dataset. A positive $FS$ indicates $f(x)$ outperforms the baseline.

$$FS = 1 - \frac{E(f(x))}{E(f'(x))} \tag{5.5}$$

An often used baseline in the context is the persistent model. This model assumes that the current situation persists until the specified forecast horizon. Although, this static approach has natural drawbacks in theory, researchers attest surprisingly good performance on real world datasets [13, 25, 28, 38]. This encourages the use of this model as a baseline model in this work while remarks regarding its shortcomings are given at appropriate sections. A second baseline is to constantly predict no clouds which is by far the simplest baseline. Obviously, a constant no cloud prediction is not very useful in practice but serves for a better grading of the results.

## 5.5. Forecasting Stage

The network was trained on the *Moving* dataset in batches of 8 images at an resolution of $256px \times 256px$. Larger resolutions could not be trained as the model size exceeded the available hardware resources (2 16GB Nvidia Tesla V100 GPUs). The model was trained for about 12 epochs at a learning rate of $1.5e^{-4}$. Longer training period did not yield better performance. Data augmentation was applied by randomly rotating a sequence by $90°, 180°$ or $270°$ as the wind direction in the dataset covers just the first $90°$.

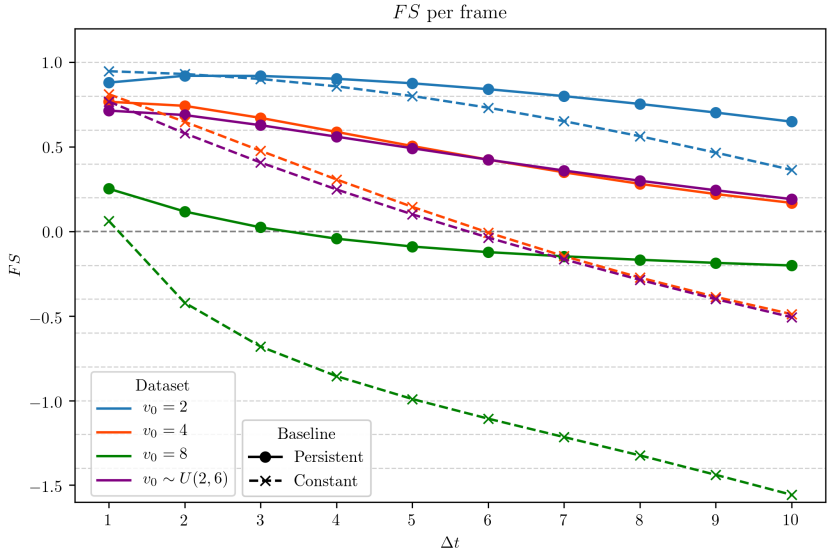



Figure 15: Forecast skill $FS$ per $\Delta t$ for each variant of the test set.

Figure 16 shows the frame-wise mean test error on different variants of the *Moving* test set. In the main test set, the same wind speed as in the training set was used ($v_0 = 2$). The first two variants were simulated with an increased wind speed ($v_0 = 4$, $v_0 = 8$) while the last variant was simulated with a uniform random wind speed between 2 to 6 $m/s$ ($v_0 \sim U(2, 6)$). The wind speed is sampled before simulating a sequence and does not change over time. Generally, the persistent model as well as the forecasting model suffer from the fact that they collect twice the error for a falsely predicted cloud. Once for the wrong cloud pixel and once for the wrong sky pixel. The constant model only collects an error for wrong sky predictions. The persistent model error therefore exceeds the constant model error as soon as half of the clouds of the persisted frame are displaced. In the $v_0 = 2$ case this happens between the second and third frame. This roughly corresponds to $150s$ due to the time interval of $60s$ between frames. In theory, doubling the speed halves this timespan to approximately $75s$. This lies in the first quarter between the first and second frame and actually the persistent model error exceeds the constant model error around this time in the $v_0 = 4$ case. With $v_0 = 8$ half of the clouds in the persisted frame are displaced after $37.5s$ which is even before the first predicted frame. This results in a failure of the persistent model

compared to the constant model even before the first predicted timestep. Furthermore, the overall cloud density slightly varies among the test sets which results in differing error magnitudes of $E_{\text{mse}}$. Within a test set the density stays stable over all frames except for the $v_0 = 8$ case in which the high cloud speed inhibited to maintain the density by the right simulation parameters. Hence, comparisons of results between different test sets should be based on forecast skill.

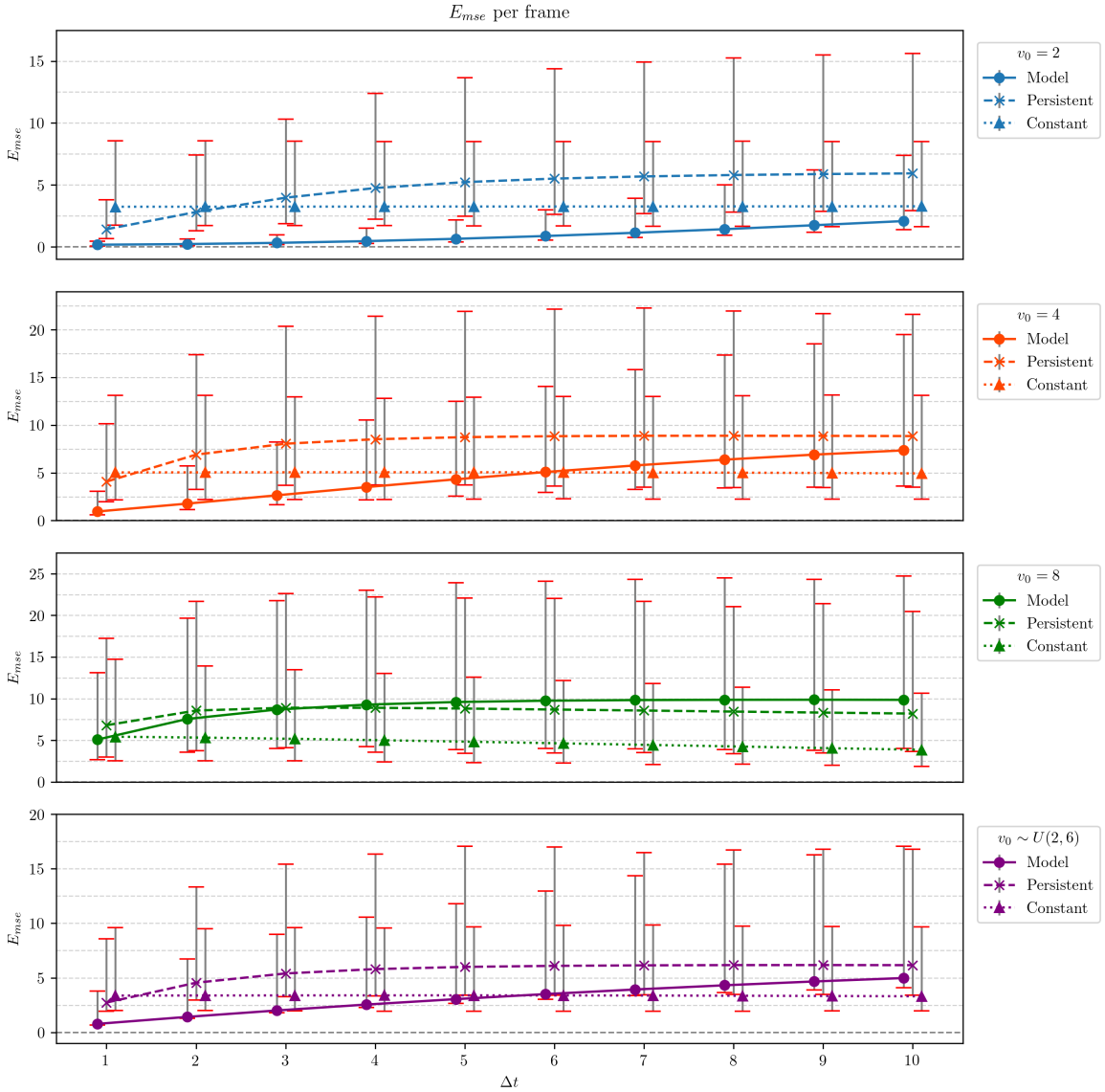



Figure 16: $E_{\text{mse}}$ per $\Delta t$ for forecasting model, baseline and a constant no-cloud prediction on four test sets with varied wind speed. The markers in the plot are horizontally displaced for illustration purposes. Each timestep shows average, minimum and maximum of the 2000 instances per set. The model was trained on the *Moving* dataset with $v_0 = 2$. A detailed list of mean values and variance is given in Table A.2.

The model outperforms the persistent baseline for each frame in the $v_0 = 2$ case. The result decreases with later frames as also indicated by forecast skill (see Figure 15). In the $v_0 = 4$ case, the model is able to outperform the baseline for all frames but is worse than the constant model after the sixth frame. In the $v_0 = 8$ case the model performs at a similar error rate as the baseline and is worse than the constant model for all frames except the first three. In the random case the forecast model $E_{\text{mse}}$ shows a very similar progression compared to the $v_0 = 4$ case but with less outliers as indicated by averages closer to the bottom of the error bar in Figure 16. The forecast skill is marginally lower for earlier frames and matches after the sixth frame.

In the $v_0 = 2$ case the forecast model $E_{\text{mse}}$ increases over time. This has two reasons. First, the shape and size of a cloud prediction changes over time, most probably due to the inability to accurately model the spatial-dependent distortion of the camera calibration. Clouds which move off the center axis are often predicted with a circular shape in later frames (see Figure 18c). Moreover, in dense cloud accumulations a *merging* effect causes an over-smoothed prediction of such (see Figure 18b). This effect gets even more visible for higher cloud velocities as shown in Figure 19a. Second reason which contributes to the model error are clouds which gradually enter the field of view during the forecasting period without being visible in the preliminary frames. The model has no chance to predict these *hidden* clouds and therefore misses them in the output (see Figure 19b). This is a technical limitation due to the cameras field of view and generally applies to all camera-based forecasting models. The effect is rather insignificant in the beginning of the $v_0 = 2$ case and gradually increases as indicated by a growing error bar. An in depth analysis revealed that sequences with the highest error rate are those which show a large number of *hidden* clouds. In contrast, movements of clouds which enter the field of view during the preliminary frames are correctly estimated as visible in Figure 18a. The model is even capable to accurately predict a clouds shape even though it has only partially entered the field of view in the preliminary frames.

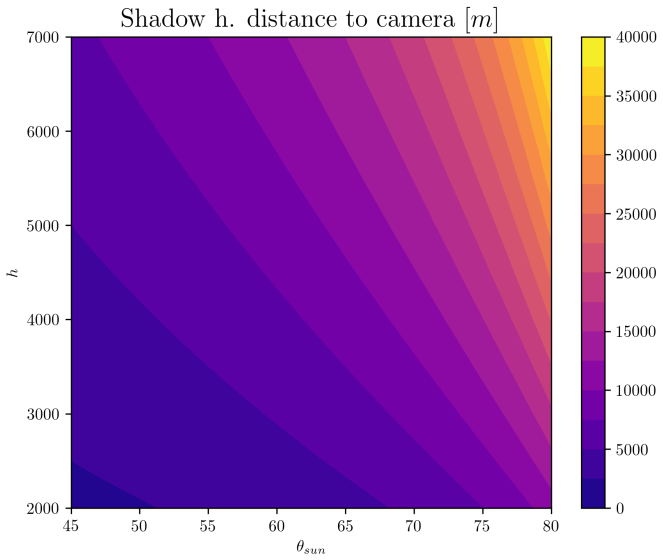
For further analysis, let $E_{\text{C}}$ denote the error due to incorrect cloud pixel prediction, $E_{\text{S}}$ denote the error due to incorrect sky pixel prediction and $E_{\text{H}}$ denote the error caused by a incorrect sky pixel prediction due to a *hidden* clouds. A *hidden* cloud pixel is not visible in the preliminary frames and is not counted in $E_{\text{S}}$. The total error is the sum of all above. By looking at the generalization capabilities for unseen cloud velocities one can say that the model performance is strongly affected by a change in wind speed. While constant model $FS$ decreases moderately over time in the $v_0 = 2$ case, the decay drastically increases for higher velocities. In the random case, the set has on average the same wind speed as the $v_0 = 4$ case due to the uniform distribution. The plot shows that both cases show more or less the same forecast skill. Thus, the effect of different wind speeds on the model error averages out within one set. This linear correlation indicates that the network systematically underestimates a clouds movement. More precisely, after each timestep $E_{\text{C}}$ and $E_{\text{S}}$ increase by a constant factor caused by

the underestimated cloud movement due to the difference between trained and actual cloud velocity. The persistent model $FS$ flattens for $v_0 = 4$ and $v_0 = 8$, since the forecast model behaves similar to the persistent model in this cases. The error rates converge as soon as most of the truth is missed. The further increase of forecast model $E_{\text{mse}}$ in the $v_0 = 8$ case for later frames is possibly caused by an increased size of the cloud predictions ($E_{\text{C}}$) as visible in Figure 19a.

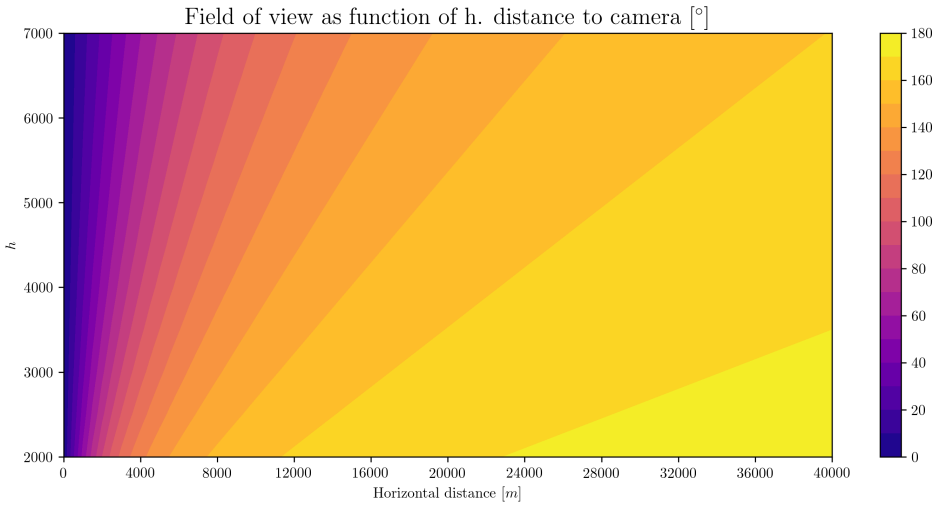
How can these error types be circumvented? With respect to cloud movement speed, the network is able to predict heterogenous movements (wind profile power law) in a sequence with low constant surface wind speed. In presence of heterogenous surface wind speed as may happen throughout a day, the forecast model performs according to the average of the set. The network is not able to sufficiently generalize for unseen data (depending on the delta between trained and actual surface wind speed). A possible solution is to train several networks for different surface wind speeds and select the network accordingly to a measured average wind speed. If the network is able to achieve the same prediction quality of the $v_0 = 2$ case likewise for higher velocities should be of special interest for future research.

Mitigate the effect of *hidden* clouds is a more fundamental question and applies in general to image-based forecasting models. How much of the predicted frame is actually relevant to the subsequent shadow prediction? The required field of view depends on the range within a cloud can cast a shadow onto the area of interest. This is a function of cloud base height $h$ and the solar zenith angle $\theta_{\text{sun}}$ and determines the horizontal distance of the cloud (see Figure 17a). In Figure 17b the required field of view with regards to horizontal distance to the camera is given. Generally, lower altitudes cause a strong increase of required field of view due to the dimensions of the area of interest. All values in the following are based on the most restrictive cloud base height. In a first perspective, without forecasting, the required field of view is mostly below 150° throughout the middle of a day ($\theta_{\text{sun}} < 70°$). For early and late daytimes ($\theta_{\text{sun}} > 70°$) the required field of view quickly reaches values above 150°. In a second perspective the maximum travel distance of a cloud during the forecast lead time is considered. Because if a *hidden* cloud occurs in the frame at the first prediction step, it is irrelevant as long as it does not enter the required field of view for the shadow prediction for the rest of the sequence. With an exemplary strong surface wind speed ($v_0 = 8$) the actual cloud velocity amounts to 17-20$m/s$ depending on cloud base height. Within the forecast lead time of 10 minutes a cloud can travel between $10,200m$ to $12,000m$. This requires a margin in the field of view of 17° ($\theta_{\text{sun}} = 70°$) resulting in a total field of view of 167°. For solar positions above 70° this value only slightly increases to 170° ($\theta_{\text{sun}} = 80°$) while a lower surface wind speed of $2m/s$ reduces these values to 157° ($\theta_{\text{sun}} = 70°$) and 165° ($\theta_{\text{sun}} = 80°$) respectively. In conclusion to the initial question of the paragraph, the relevant field of view ranges between 165 to 170° depending on the wind conditions. While this applies to forecasting in general, it reveals a significant drawback of camera calibration. By this the effective field of view is reduced (157°in this model) which can only be recovered by the use of additional cameras. A first next

step would be to quantify the forecast error of *hidden* clouds for various solar positions and wind speeds. Based on this a balance between long term prediction quality and model complexity can be determined.




(a)




(b)

Figure 17: (a) Horizontal distance of a cloud casting a shadow on the camera. (b) Required field of view as function of cloud base height and horizontal distance to camera. A cloud with base height of $3km$ has a horizontal distance of $5km$ at a solar zenith angle of 60° in order to cast a shadow onto the camera. The one-sided area of interest comprises $2km$ and the required field of view to see such cloud in the image which casts a shadow onto this areas lies between 130 and 140°.

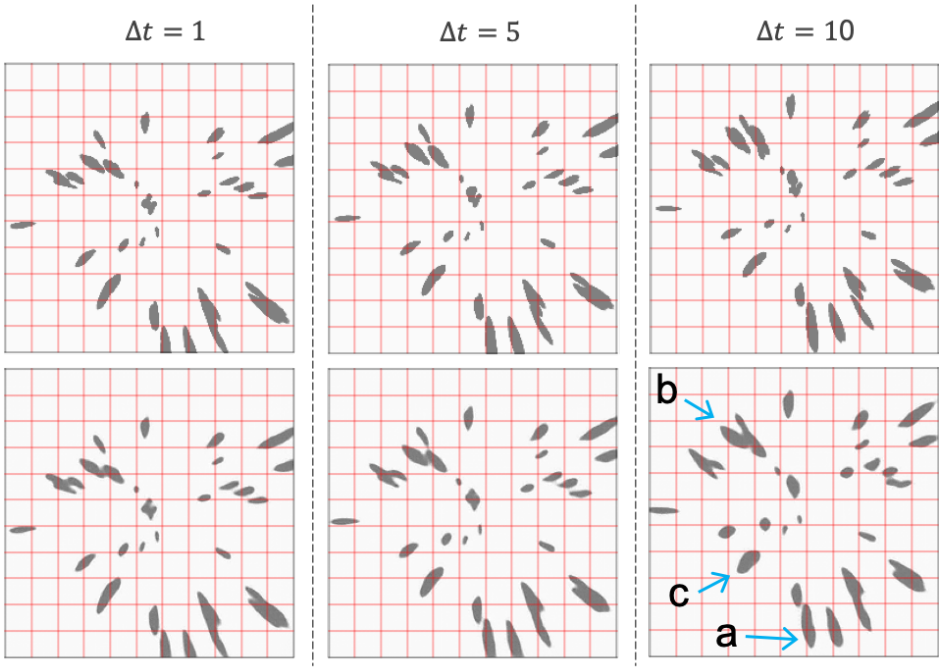



Figure 18: Qualitative comparison of ground truth (top row) and prediction (bottom row) of forecasting stage at different timesteps for $v_0 = 2$. (a) Correctly estimated cloud shape. (b) *Merging* effect. (c) Lacking affine transformation.

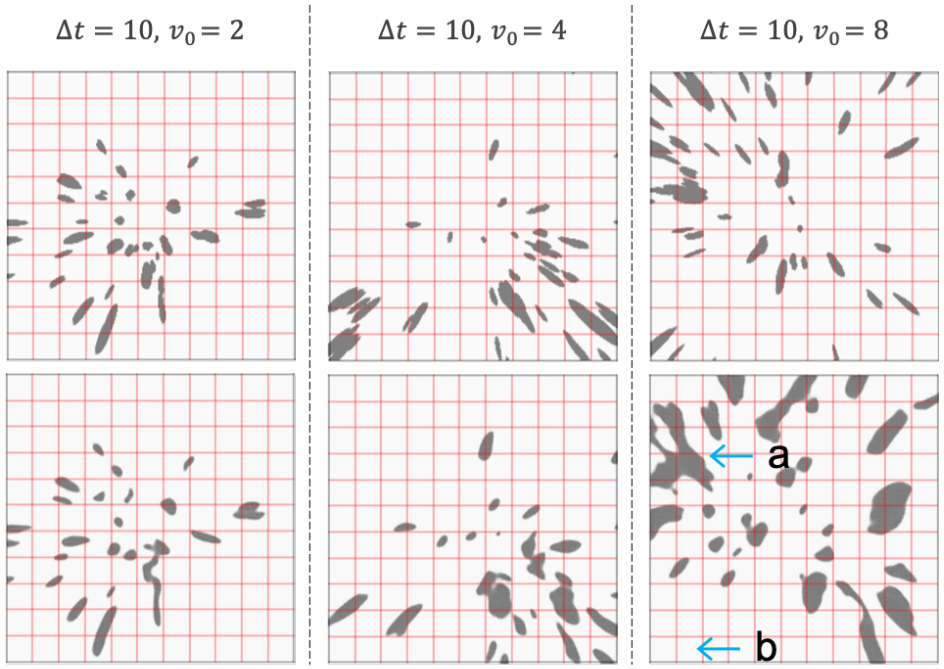



Figure 19: Last frame of the predicted sequence for variants of the *Moving* test set with higher wind speed. (a) Merging effect. (b) *Hidden* clouds. Top row shows ground truth, bottom row model prediction.

## 5.6. Extraction Stage

The two networks of the extraction stage were trained on the *Depth* dataset also at an image resolution of $256px \times 256px$. The batch size was 8 (*MonoDepth2*) and 4 (*GeoNet*) respectively. The training procedure for *GeoNet* was 60 epochs at a learning rate of $2e^{-4}$ and another 60 epochs at a rate of $2e^{-5}$. *MonoDepth2* showed a much slower convergence possibly due to the chosen batch size. It was trained in three rounds each 60 epochs long at learning rates of $1e^{-4}$, $1e^{-5}$ and $5e^{-6}$. Both networks employed data augmentation by randomly flipping single or both images axis and modifying brightness and saturation. The remaining training parameters from the original publication were left unchanged.

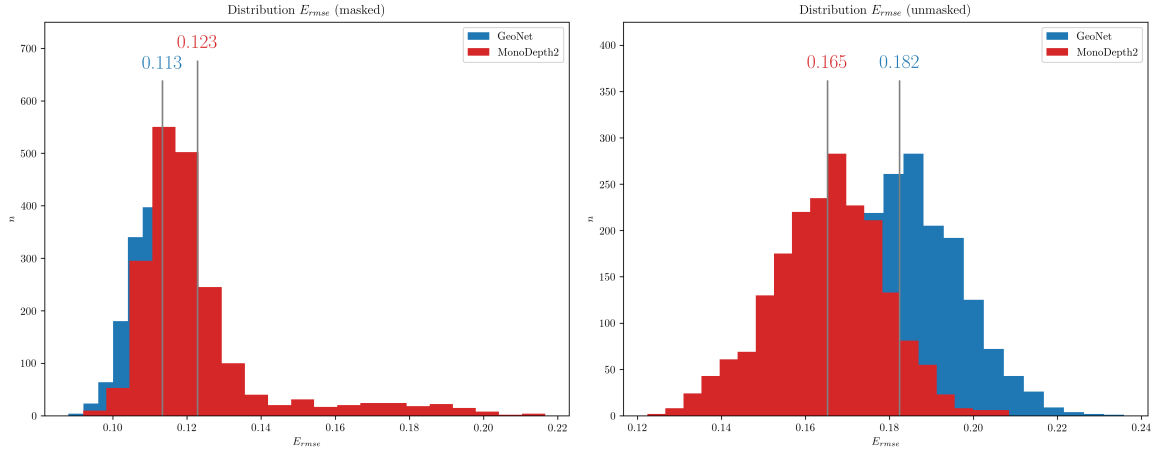



Figure 20: $E_{\text{rmse}}$ of the two architectures on first frame of the *Depth* test set once for masked background and once for unmasked background. The mean $E_{\text{rmse}}$ is denoted at the top of each histogram.

The networks were both trained in a self-supervised manner. Therefore, the output of these network do not share the same normalization and most certainly also not the one of the simulated ground truth data. It can be observed that *GeoNet* outputs a value between 7 to 9 for infinite depth while *MonoDepth2* outputs 0 in such case. The simulation encodes infinite depth as 0 (compare Figure 21). The behavior of *GeoNet* is the result of two effects. Pixel of infinite depth (sky) have no texture or color difference and a reprojection of such pixel onto another sky pixel will not cause any loss in image similarity. To recall, the network is trained so that the estimated depth map first of all minimizes the structural similarity index between the target image and reprojected source frames. Consequently, the network learns to predict a depth for sky pixel which results in reprojection onto another sky pixel. But this alone does not explain that the network outputs a depth at all for such pixel. The second effect is caused by the depth smoothness constrain which minimizes the gradient between pixel in the depth map. There are regions in the image namely the areas close to clouds in which a reprojection based on incorrect depth causes a similarity loss. Thus, the network learns to

predict a certain depth for such pixel. In order to minimize image gradients, adjacent pixel receive a similar depth which balances the similarity loss due to reprojection and smoothness loss. Because of the mentioned reason the similarity loss is of decreasing significance in low textured regions and therefore causes the network to smooth out the depth prediction over such areas. This also explains the slight variations in depth among the maps for sky pixel as the network must average over all present cloud depth predictions (see Figure 21, second row). In contrast, *MonoDepth2* predicts 0 at least for some sky pixel although it is trained with the same loss and reprojection function as *GeoNet*. This is a result of the pixel masking mechanism which purges the loss for stationary pixel. This inhibits the smoothness loss to contaminate the supervisory signal for low texture regions so that the network correctly learns to predict 0 for sky pixel. The role of stationary pixel masking for the model performance is discussed later in this section.

Why does a non-zero value for infinite depth does not affect the network of the subsequent projection stage? In theory, the position of a pixel with infinite depth matches the position of sky pixel in the image. Moreover, the 2D filter of a CNN is applied on all channels of the input and results are summed up vertically. Since the predicted depth map will be stacked channel-wise with the camera image, the assumption is that the constant value of infinite depth predictions apply to all sky pixel and therefore alter the magnitude of all filter activation in the same way. In order to reflect this assumption in the subsequent evaluation, a mask is computed to exclude pixel of infinite depth from the evaluation. The mask covers all pixel which have a depth of 0 in the ground truth depth map (*masked* case). Any further computations apply to the remaining pixel if not specified differently. Next the differing normalization needs to be harmonized. The ground truth is normalized by a constant value of $25,500$ while the model predictions are normalized based on the intrinsic camera parameters used in the reprojection function. By the equation given in (5.6), a sequence of data points $x = (x_1, x_2, ..., x_n)$ is normalized from 0 to 1. Afterwards each ground truth depth map $D$ and predicted depth map $D'$ is zero-centered by subtracting its mean $\mu_D$ and $\mu_{D'}$ respectively. Lastly, the depth maps are normalized a second time by equation (5.6) at which the global minimum/maximum among ground truth and prediction is used instead.

$$z_i = \frac{x_i - min(x)}{max(x) - min(x)} \tag{5.6}$$

Figure 20 displays the masked and unmasked performance of the models on the *Depth* dataset. Surprisingly, *GeoNet* slightly outperforms in the more relevant case of masked depth maps although *MonoDepth2* initially was considered the superior model. An average $E_{\text{rmse}}$ of 0.113 means that the predicted depth of a single pixel differs by 11.3% from the correct depth on average. Even in the unmasked comparison, *MonoDepth2* does not achieve significant better quality e.g. due to its masking mechanism of stationary pixel. By looking at qualitative performance in Figure 21, the predicted depth maps of *MonoDepth2* appear much more noisy even thought the total loss function includes the depth smoothness constrain $\mathcal{L}_{\text{ds}}$. As mentioned before, a very likely explanation is the loss purging due to the stationary pixel masking criterion. Generally, one would expect a rather smooth mask covering most of the stationary sky pixel and some low textured clouds pixel. Instead, the masks look almost like white noise indicating a failure of the masking criterion. A modification of the criterion or even removal of the masking mechanism may be beneficial to the model performance and is left to future work. Hence, the remainder of the pipeline is performed with *GeoNet* as feature extractor of this stage.

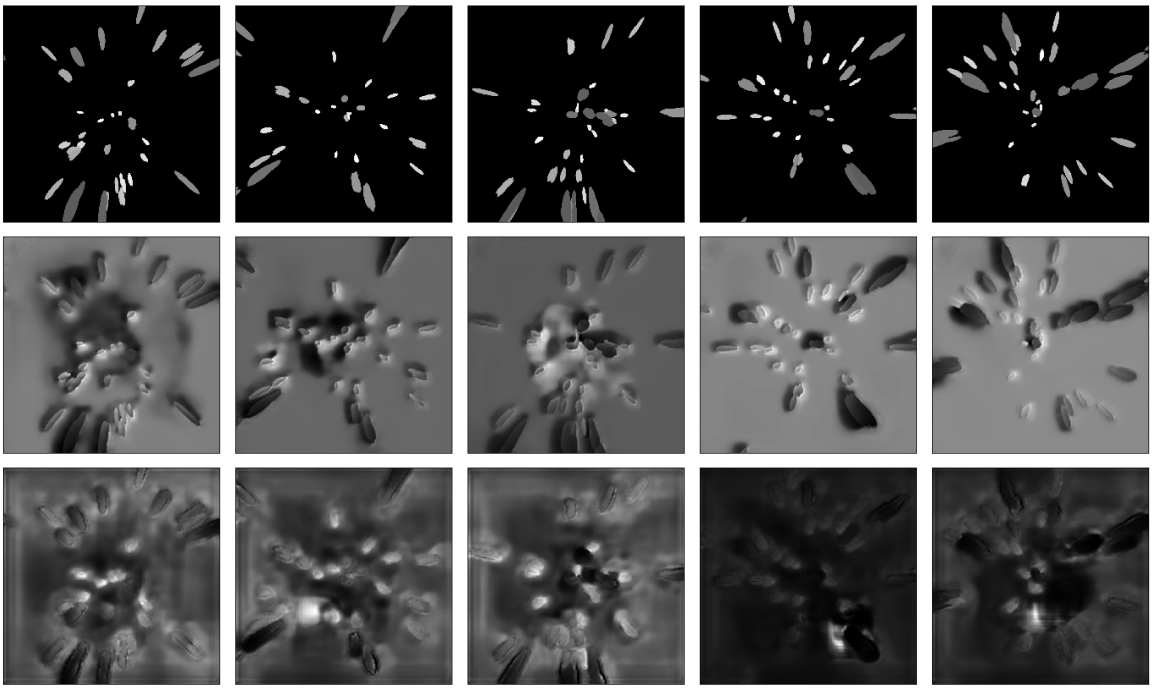


Figure 21: Qualitative comparison of depth maps: ground truth (first row), prediction *GeoNet* (second row) and prediction *MonoDepth2* (third row). Brighter means higher and black means infinite or no depth.

Regarding the predictions of *GeoNet*, depth maps are generally smooth and roughly capture the correct height profile of the clouds. In some cases the depth maps show artefacts around the center and/or halos around far distant clouds (see Figure 21, first

two from left in second row). Another notable aspect is that *GeoNet* is able to predict differing height profiles of overlapping clouds from just a single image. This is especially surprising as the authors of *GeoNet* did not take specific means to account for this. Also, two clouds at different altitudes which overlap from the ground perspective, do not appear very differently compared to a single, larger cloud. However, this should be treated with care as it grounds on simple, appearance-based image features which are very specific for the simulated dataset and may be very different in e.g. a real world dataset.

## 5.7. Projection Stage

For the evaluation of this stage, multiple case studies were performed. The first study evaluates segmentation and projection capabilities for clouds at a constant height (*Flat* dataset). The second block investigates on performance for clouds with different height profile and the benefit of prior extracted depth features (*Depth* dataset). Lastly, the same aspect is researched for the first frame of the *Moving* dataset. Each training was performed in two rounds of 60 epochs each at a learning rate of $1e^{-5}$ and $5e^{-6}$ respectively. A significant speed up of training was achieved with a batch size of 16 and batch normalization enabled.

First to mention is the positive effect of image rectification due to camera calibration. The network achieves .1352 mean $E_{\text{rmse}}$ in the rectified *Flat*/*Flat* case (Table 3) while this error increases to .2746 in the non-rectified case. Secondly, the network is able to learn a static height profile and underperforms as expected if presented to clouds with a different height (*Flat*/*Depth* case). The predicted shadow is slightly offset the correct position as depicted in Figure 22.

In the *Depth* dataset train case, the network surprisingly does not gain much performance by being trained with or without ground truth depth features. However, estimating depth features has a minor effect as opposed to not introducing such depth features. Looking at the error rates of the *Flat* and *Depth* training set, it seems that there is a general misfit of the model performance for clouds with static height and clouds at diverging height. Although the quantitative results differ only slightly, the qualitative results show a completely different picture. Predicted shadows in the *Flat*/*Flat* case are clear and accurate while in all *Depth*/*Depth* cases such are rather blurred. A likely explanation is data imbalance as the *Flat* dataset contains on average 10.31% occulted pixels per shadow map while the *Depth* dataset contains 6.32%. The ratio of occulted pixel in shadow maps of the *Moving* dataset however reaches 15.07% which is also reflected by more certain shadow predictions. Nevertheless, the error rate in the *Depth*/*Depth*/GN case share the same magnitude as the *Flat*/*Flat* case and attest for a minor benefit of estimated depth features.

For the last set of cases the error rate drastically increases which is a result of the higher number of shadows present in the dataset. In contrast to the previous case,

| **Train** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| *Flat* | | | *Depth* | | | *Moving* | | |
| **Test** | **DC** | $\boldsymbol{E}_{\text{rmse}}$ | **Test** | **DC** | $\boldsymbol{E}_{\text{rmse}}$ | **Test** | **DC** | $\boldsymbol{E}_{\text{rmse}}$ |
| *Flat* | - | .1352 | *Depth* | GT | .1497 | *Moving* | GT | .4759 |
| *Flat* | - | .2746* | *Depth* | - | .1607 | *Moving* | - | .4088 |
| *Depth* | - | .2331 | *Depth* | GN | .1555 | *Moving* | GN | .4103 |

Table 3: Results of different training/testing setups. The second row denotes the dataset used for training (in the *Moving* set only the first frame is used). The DC column describes the depth channel used: (GT) ground truth, (GN) *GeoNet* prediction and (-) no depth. (*) was trained and tested without prior image rectification.

providing ground truth depth features shows the worst performance. Figure 22 shows that shadow position and shape are amiss in this case. A notable but probably rather insignificant source of error is depth map discretization. The simulation encodes such maps as greyscale images with 255 color levels at a resolution of $100m$ per level (see Section 5.1). Given the dimensions of the simulated shadow map and an exemplary extreme value of 81° for $\theta_{\text{sun}}$, the maximum deviation in the shadow map amounts to $315.69m$ or $7.89px$. While this partly accounts for the offset in the *Moving*/*Moving*/GT case, the offset in remaining tests indicates a general bias. Upon closer inspection there exist two failure modes, one in which the shape of a shadow was correctly estimated but its position is offset and the second mode where both, general shape and position, is wrong. The reason behind this observations could not entirely be clarified. One hypothesis is insufficient evidence in the data for each solar position. The network effectively learns from solar positions and sky conditions which cause shadows. Although solar positions are equally represented in the data, it is not guaranteed for the shadow map to actually contain shadows. The network then performs accordingly to the potential of a solar position to yield such low quality training instances. One may argue that this also applies for other datasets but the number of clouds is much lower in these which reduces the variance in the image features. Generally, the performance of the last stage for a more versatile dataset stays behind the expectations.

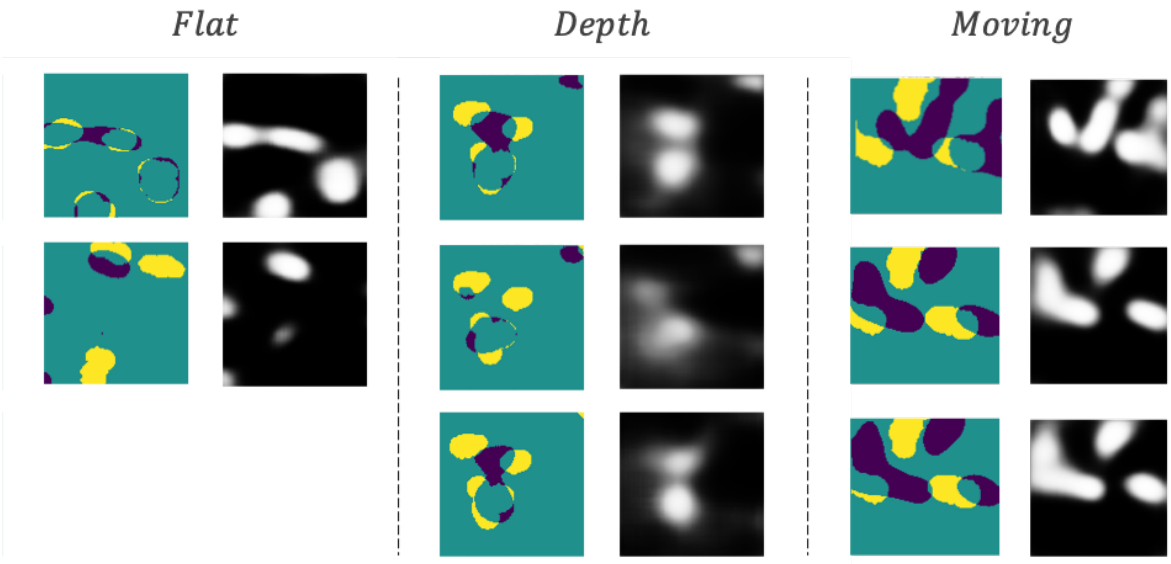



Figure 22: Qualitative comparison of prediction per case in the same order as in Table 3 (except un-rectified case). Left, binarized residual of ground truth (yellow) and prediction (purple). Right, predicted shadow map.

The evaluation of the entire pipeline is performed (trained/tested) on the *Moving* dataset. The first stage predicts 10 future frames from a preliminary sequence of 10 frames. The second stage predicts depth features for these frames and the last stage predicts the corresponding shadow maps. The persistent model (see Section 5.4) serves again as baseline. Figure 23 shows average $E_{\text{rmse}}$ and $FS$ per frame in which the model seems to outperform the baseline after the second frame. The baseline maintains a constantly higher error after the sixth frame whereas the model seems to reduce its error over time. This is the result of two effects. First, with increasing time more and more clouds left the circumsolar region resulting in significantly less shadows for later frames (see Figure 23, decreasing ground truth occlusion ratio). Similar to the behavior described in Section 5.5, the persistent model takes the shadow map of the last seen frame as prediction and accumulates error over time until it eventually converges. The increase in persistent model $FS$ is therefore just a consequence of the baseline getting worse. Second, the model constantly predicts almost no shadow as indicated by a low occlusion ratio of the forecast model in Figure 23. The model is actually slightly worse than a constant no shadow prediction. Speaking in classification terms, the model is not very sensitive with a high number of false negatives. Since the projection model was previously able to predict dense shadow maps for the *Moving* dataset either with true or predicted depth maps, the root cause must lie in the synthesized input images of the extraction stage. This indicates a dependency of these stages on certain image features which significantly change if generated by the forecasting model.

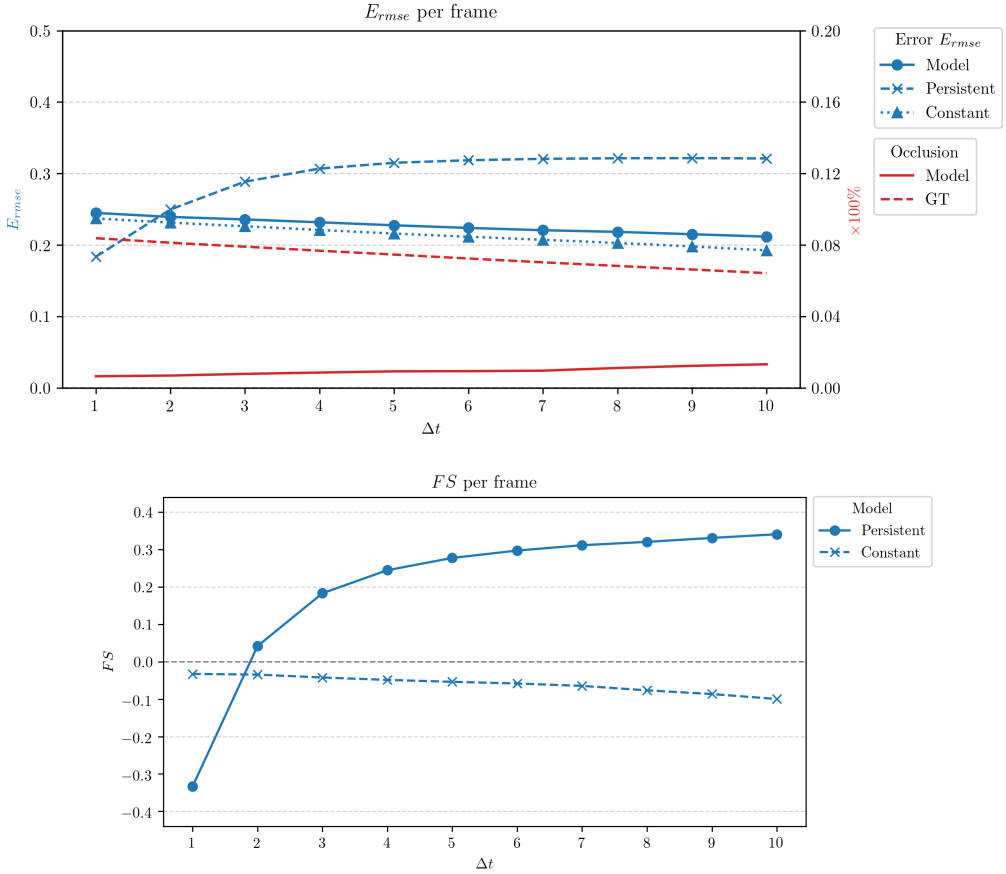



Figure 23: Frame-wise average of $E_{\text{rmse}}$ (lower is better) and $FS$ (higher is better) for the entire pipeline compared to the baseline. Additionally, the average shadow map occlusion ratio is given for prediction and ground truth. A detailed list of values is given in Table A.1.

# 6. Discussion

Following up on the insights of the previous chapter, this part elaborates possible improvements for future works. At first, the discussion sheds light on each stage of the pipeline. Subsequently, also fundamental advances to the architecture are examined.

The model of the forecasting stage outperformed the baseline for a trained, moderate cloud velocity. Qualitative results show that the model is capable to track heterogenous cloud movements within a sequence of images according to the wind profile power law at a constant surface wind speed and interpolate those into the future. It is also able to predict size and shape of occulted and partly visible clouds at the edge of the frame. For unseen velocities the model systematically underestimates cloud movements and thus is limited in the general case. The idea of using multiple, specialized networks as countermeasure was discussed. Here, future work should have a special focus on if the network can maintain its performance also for higher cloud velocities. In a broader perspective, the largest fraction of the error was caused by a merging of nearby clouds to a smoothed, single unit. A potential countermeasure is to include a $L1$ loss term to penalize smooth image gradients stronger. Another countermeasure would be an entire change of the loss function towards the structural similarity loss presented in Section 4.2 which is a more state-of-the-art approach for image synthesis. Lastly, a word on how the model can be trained for higher image resolutions despite the vast resource requirements. Instead of resizing images by standard interpolation techniques and possibly loosing information, one can use a single encoder to downsample inputs and a single decoder to upsample predictions. This image feature extractor is jointly trained with the forecasting network. Losses are summed over all scales of the prediction to train the encoder-decoder. Multi-scale loss is a common technique and is applied in [44, 59, 75] to name a few.

In the extraction stage two improvements can be made. First, the aim was to work with a single camera in the field. This does not necessarily apply to training. The method of reconstruction based monocular depth estimation can also be done for stereo image pairs. Here, the source frames origin from a stereo-camera pair while the target frame still origins from a single camera. This method is very comparable to a human who estimates distances with one eye covered. While the authors of *GeoNet* did not investigate this approach, the authors of *MonoDepth2* denote an improvement for their model. The method seamlessly integrates into the existing training framework and can therefore easily be transferred to the *GeoNet* model. The second improvement addresses the stationary pixel masking mechanism in *MonoDepth2*. This mechanism often failed and caused unintended masking which inhibits the network to learn. A modification of the masking criterion is left to future research.

As mentioned, the last stage offers the greatest potential for improvements. Despite the described architectural changes regarding the encoder of the model, further preprocessing might be a simpler enhancement. Missing evidence of certain solar positions

is the central hypothesis for the lack of performance. Since each predicted frame is processed individually, the most relevant area always lies around the circumsolar region. Consequently, the network does not need to see a whole 360° image to estimate shadows. The field of view can be limited to a smaller streak covering only the solar trace on the zenith axis. In addition, to this reduction of dimensionality, the input variance can be further diminished. One mean is to train many networks for different solar positions. This way, each network is presented to a more or less static solar position and only needs to generalize for this. The encoders of the networks can even share weights to speed up training of cloud segmentation. A second extension is the use of spatial transformer networks. This singular CNN module allows to jointly train a sampling layer for transformation-invariant feature representation. The authors report significant performance gains at almost no training overhead for image classification [76]. In the projection network, this advancement may further balance optical variance due to fish-eye distortion or introduced over-compensation by image rectification.

From a more general perspective, separated training of depth feature extractor and projection model did not provide a significant performance gain compared to training without depth features. Although, the deeper reasons for this contrary outcome have not been fully understood, it opens up the discussion for an alternative architecture. While the proposal fosters an end-to-end neural learning approach, certain aspect can also be solved analytically. The general planar projection model is easily applied as soon as the 3D coordinate of an object is known. So far this coordinate was somehow encoded in the inner, latent sub-space unit of an encoder-decoder network. Instead, a network can be trained to predict such coordinate explicitly while the projection is computed analytically. This architecture does not need any additional data if used with e.g. the reconstruction-based training framework of the second stage. The predicted coordinates of cloud pixel are used to compute a shadow map. The supervisory signal is given by the error between this prediction and the ground truth shadow map. In this framework, segmentation and depth estimation can be jointly performed by a single network e.g. *GeoNet*, *MonoDepth2*, the design of Tatarchenko et al. [55] or by separate networks i.e. segmentation and depth. Possible NN-based candidates for multi-class segmentation task are given in Section 3. A general challenge in this approach is the generalization capabilities of the networks as the supervisory signal is only present for areas in the image which can cause shadows.

Following up on the idea of merging the last two stages, one can also think of a hybrid approach. Here, forecasting is performed by the network of the first stage while shadow prediction is done analytically. The end-to-end test showed that extraction and projection stage are much more sensitive to the input image being real or synthesized than expected. A hybrid architecture might lower this dependency. Such model will require explicit cloud segmentation either by conventional means i.e. thresholding or a learning approach which in this case would require additional ground truth data. Nonetheless and thanks to the image-based forecasting, standard techniques for 3D reconstruction e.g. general photogrammetry as used by Blanc et al. [13] or specialized

cloud photogrammetry as presented by Crispel et al. [77] can be applied. The use of novel reconstruction techniques is very promising and future work should focus on improved cloud segmentation in this prospect.

# 7. Conclusion

In this work, an image-based model for spatially comprehensive solar irradiance nowcasting was proposed. In contrast to existing work, an end-to-end neural network approach was used which also closes the gap of such a model in literature. The approach consists of three stages: forecasting, extraction and projection where each stage comprises a specialized neural network architecture. The overall model performance is mixed and many advancements to improve are discussed. Generally, the forecasting and extraction stage performed well in their task while most of the overall error origins in the projection stage. NN-based approaches are a promising technique to conquer common shortcomings of conventional methods in solar irradiance nowcasting. It also benefits from a very active research community which continuously improves the state-of-the-art. Certainly, there exist multiple possible designs to tackle the general task at hand and some were discussed earlier. The proposed three-staged layout of the problem forms a first narrative and by that contributes also general design decisions i.e. pre-processing for such models. Future work should follow up on this framework implementing stated advancements while also discussing different problem layouts. Lastly to mention, future work should establish standardized datasets for the task making the problem more accessible to the research community and allowing resilient inter-model evaluation.

# A. Appendix

| | | $\boldsymbol{E}_{\text{rmse}}$ | | $\boldsymbol{FS}$ |
|---|---|---|---|---|
| **Model** | | *M* | *BS* | - |
| $\boldsymbol{\Delta t}$ | *1* | .2449 | .1836 | -.3337 |
| | *2* | .2393 | .2497 | .0417 |
| | *3* | .2358 | .2886 | .1831 |
| | *4* | .2318 | .3069 | .2448 |
| | *5* | .2276 | .3150 | .2775 |
| | *6* | .2239 | .3186 | .2973 |
| | *7* | .2207 | .3206 | .3116 |
| | *8* | .2184 | .3214 | .3205 |
| | *9* | .2151 | .3215 | .3311 |
| | *10* | .2117 | .3212 | .3409 |

Table A.1: List of frame-wise mean error of overall model performance ($M$) compared to baseline persistent model ($BS$).

| | | $\boldsymbol{E}_{\text{mse}}$ | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | | $M$ $(v_0 = 2)$ | | $BS$ $(v_0 = 2)$ | | $C$ $(v_0 = 2)$ | | $M$ $(v_0 = 4)$ | | $BS$ $(v_0 = 4)$ | |
| **Metric** | | $\mu$ | $\sigma^2$ | $\mu$ | $\sigma^2$ | $\mu$ | $\sigma^2$ | $\mu$ | $\sigma^2$ | $\mu$ | $\sigma^2$ |
| $\boldsymbol{\Delta t}$ | *1* | 0.170 | 0.001 | 1.420 | 0.064 | 3.239 | 0.353 | 0.950 | 0.064 | 4.080 | 0.372 |
| | *2* | 0.223 | 0.002 | 2.812 | 0.244 | 3.243 | 0.355 | 1.775 | 0.222 | 6.908 | 1.031 |
| | *3* | 0.319 | 0.005 | 3.970 | 0.484 | 3.247 | 0.357 | 2.645 | 0.437 | 8.057 | 1.383 |
| | *4* | 0.461 | 0.014 | 4.752 | 0.693 | 3.251 | 0.358 | 3.504 | 0.678 | 8.524 | 1.521 |
| | *5* | 0.647 | 0.034 | 5.228 | 0.841 | 3.256 | 0.359 | 4.324 | 0.920 | 8.742 | 1.576 |
| | *6* | 0.873 | 0.070 | 5.511 | 0.934 | 3.259 | 0.361 | 5.085 | 1.149 | 8.842 | 1.592 |
| | *7* | 1.135 | 0.128 | 5.686 | 0.991 | 3.263 | 0.362 | 5.774 | 1.361 | 8.882 | 1.582 |
| | *8* | 1.426 | 0.213 | 5.801 | 1.024 | 3.267 | 0.363 | 6.384 | 1.550 | 8.886 | 1.545 |
| | *9* | 1.744 | 0.331 | 5.879 | 1.044 | 3.270 | 0.364 | 6.911 | 1.715 | 8.875 | 1.522 |
| | *10* | 2.081 | 0.481 | 5.935 | 1.055 | 3.273 | 0.365 | 7.355 | 1.852 | 8.854 | 1.503 |

Table A.2: List of frame-wise mean error of the first stage model performance ($M$) compared to baseline persistent model ($BS$) and constant model ($C$).

| $C$ $(v_0 = 4)$ | | $M$ $(v_0 = 8)$ | | $BS$ $(v_0 = 8)$ | | $C$ $(v_0 = 8)$ | | $M$ $(v_0 \sim U(2,6))$ | |
|---|---|---|---|---|---|---|---|---|---|
| $\mu$ | $\sigma^2$ | $\mu$ | $\sigma^2$ | $\mu$ | $\sigma^2$ | $\mu$ | $\sigma^2$ | $\mu$ | $\sigma^2$ |
| 5.043 | 0.632 | 5.087 | 0.803 | 6.808 | 1.121 | 5.427 | 0.757 | 0.780 | 0.384 |
| 5.053 | 0.635 | 7.556 | 1.355 | 8.568 | 1.601 | 5.317 | 0.732 | 1.418 | 1.301 |
| 5.059 | 0.638 | 8.688 | 1.540 | 8.906 | 1.646 | 5.170 | 0.702 | 2.007 | 2.290 |
| 5.062 | 0.643 | 9.279 | 1.587 | 8.900 | 1.607 | 5.002 | 0.673 | 2.549 | 3.148 |
| 5.061 | 0.650 | 9.593 | 1.611 | 8.809 | 1.559 | 4.819 | 0.650 | 3.048 | 3.806 |
| 5.054 | 0.655 | 9.750 | 1.629 | 8.690 | 1.475 | 4.628 | 0.632 | 3.509 | 4.249 |
| 5.039 | 0.658 | 9.829 | 1.636 | 8.570 | 1.395 | 4.436 | 0.611 | 3.930 | 4.489 |
| 5.017 | 0.656 | 9.857 | 1.642 | 8.445 | 1.331 | 4.242 | 0.582 | 4.317 | 4.554 |
| 4.983 | 0.650 | 9.867 | 1.681 | 8.322 | 1.284 | 4.046 | 0.550 | 4.667 | 4.482 |
| 4.941 | 0.640 | 9.854 | 1.741 | 8.205 | 1.239 | 3.854 | 0.524 | 4.982 | 4.314 |

| $BS$ ($v_0 \sim U(2,6)$) | | $C$ ($v_0 \sim U(2,6)$) | |
|---|---|---|---|
| $\mu$ | $\sigma^2$ | $\mu$ | $\sigma^2$ |
| 5.427 | 0.757 | 3.370 | 0.497 |
| 5.317 | 0.732 | 3.381 | 0.501 |
| 5.170 | 0.702 | 3.389 | 0.504 |
| 5.002 | 0.673 | 3.393 | 0.505 |
| 4.819 | 0.650 | 3.392 | 0.504 |
| 4.628 | 0.632 | 3.387 | 0.501 |
| 4.436 | 0.611 | 3.375 | 0.498 |
| 4.242 | 0.582 | 3.357 | 0.495 |
| 4.046 | 0.550 | 3.334 | 0.496 |
| 3.854 | 0.524 | 3.306 | 0.499 |

| | | $\boldsymbol{FS}$ | | | | | |
|---|---|---|---|---|---|---|---|
| **Model** | | $BS$ ($v_0 = 2$) | $C$ ($v_0 = 2$) | $BS$ ($v_0 = 4$) | $C$ ($v_0 = 4$) | $BS$ ($v_0 = 8$) | $C$ ($v_0 = 8$) |
| $\boldsymbol{\Delta t}$ | *1* | 0.88 | 0.95 | 0.77 | 0.81 | 0.25 | 0.06 |
| | *2* | 0.92 | 0.93 | 0.74 | 0.65 | 0.12 | -0.42 |
| | *3* | 0.92 | 0.90 | 0.67 | 0.48 | 0.02 | -0.68 |
| | *4* | 0.90 | 0.86 | 0.59 | 0.31 | -0.04 | -0.86 |
| | *5* | 0.88 | 0.80 | 0.51 | 0.15 | -0.09 | -0.99 |
| | *6* | 0.84 | 0.73 | 0.42 | -0.01 | -0.12 | -1.11 |
| | *7* | 0.80 | 0.65 | 0.35 | -0.15 | -0.15 | -1.22 |
| | *8* | 0.75 | 0.56 | 0.28 | -0.27 | -0.17 | -1.32 |
| | *9* | 0.70 | 0.47 | 0.22 | -0.39 | -0.19 | -1.44 |
| | *10* | 0.65 | 0.36 | 0.17 | -0.49 | -0.20 | -1.56 |

Table A.3: List of frame-wise forecast skill of the first stage model performance compared to baseline persistent model ($BS$) and constant model ($C$).

| $BS$ ($v_0 \sim U(2,6)$) | $C$ ($v_0 \sim U(2,6)$) |
|---|---|
| 0.71 | 0.77 |
| 0.69 | 0.58 |
| 0.63 | 0.41 |
| 0.56 | 0.25 |
| 0.49 | 0.10 |
| 0.42 | -0.04 |
| 0.36 | -0.16 |
| 0.30 | -0.29 |
| 0.24 | -0.40 |
| 0.19 | -0.51 |

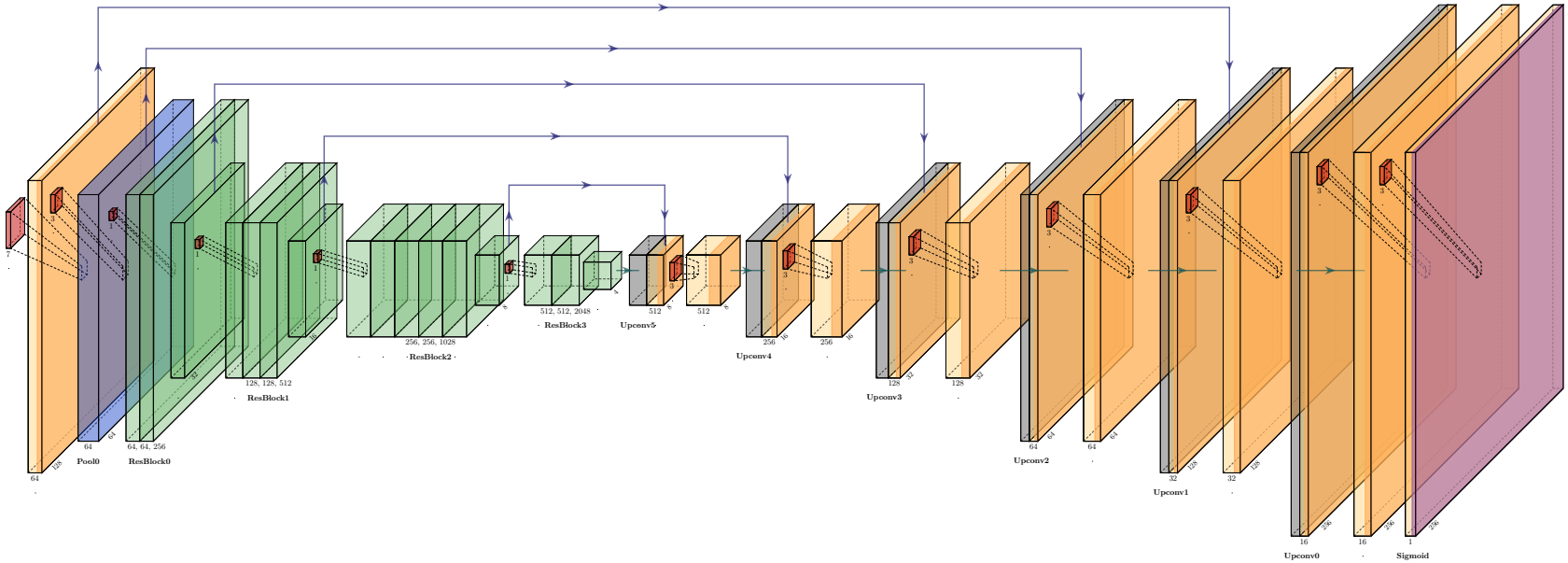



Figure A.1: Architecture of the network used for depth estimation in *GeoNet*. The first convolution layer (orange) uses a filter stride of 2. The max pooling layer (blue) halves the input dimensions followed by a sequence of blocks of residual layers (green, see Figure A.2 for details). The filter specification applies to all layers of a residual block. The upconvolution layer are a combination of an upsampling by nearest neighbor (grey) and a convolutional layer with a filter size of 3 and stride of 1. A second convolutional layer is applied on the channel-wise concatenation of the upconvolution and the outputs of previous residual blocks at the same scale. All layer use *ReLu* activation.

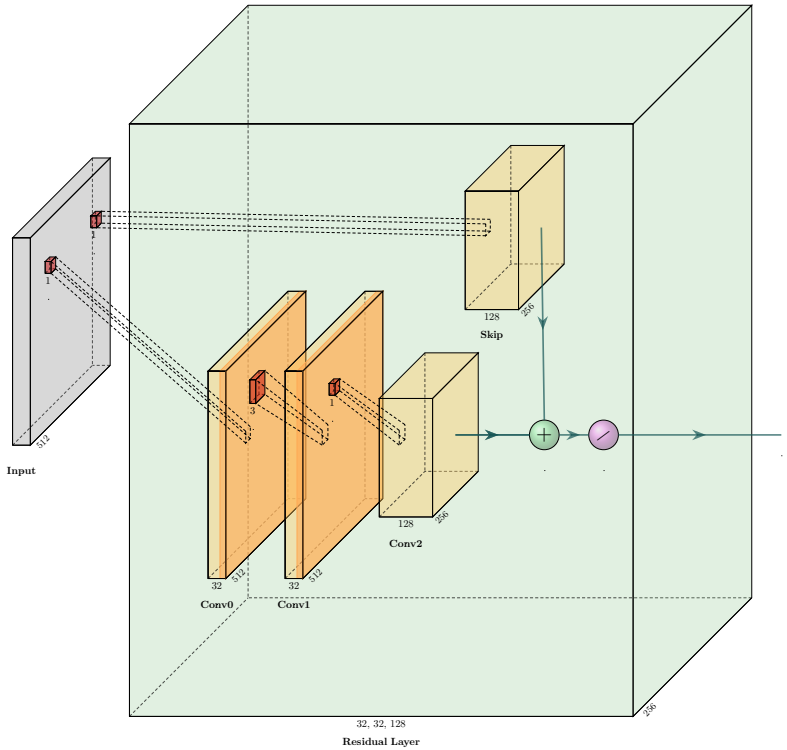



Figure A.2: A residual layer in *GeoNet* consists of three main convolutions and a skip convolution which directly links the input to the output. The Skip layer and the Conv2 layer are applied with stride 2 if the dimension of the output is halve the dimensions of the input. Otherwise a residual layer does not change the input dimensions. Conv2 and Skip layer have the same number of filters and do not apply any activation function. The sum of the outputs of these layers is passed to a *ReLu* function.

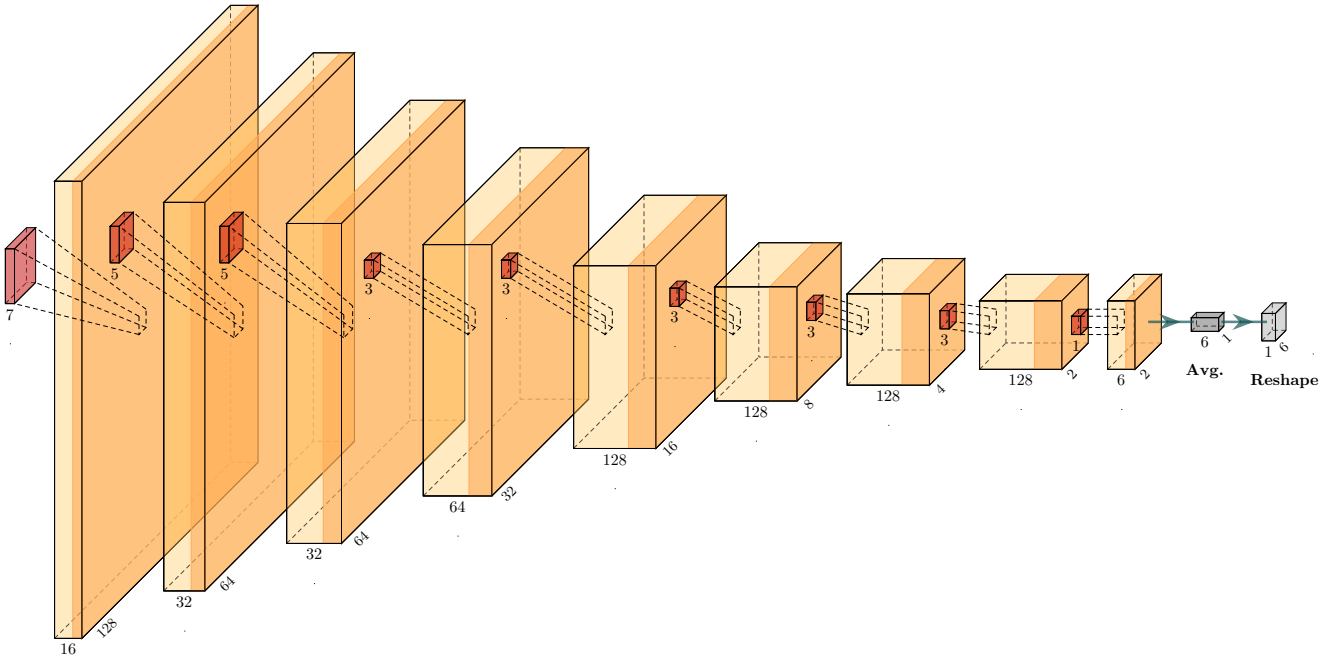



Figure A.3: Architecture of the network used for pose estimation in *GeoNet*. Each convolution has a filter stride of 2 except and uses *ReLu* activation except for the last convolution which runs with stride 1 and no activation function. The output is a vector with 6 elements describing translation and rotation of the viewpoint between two adjacent frames.

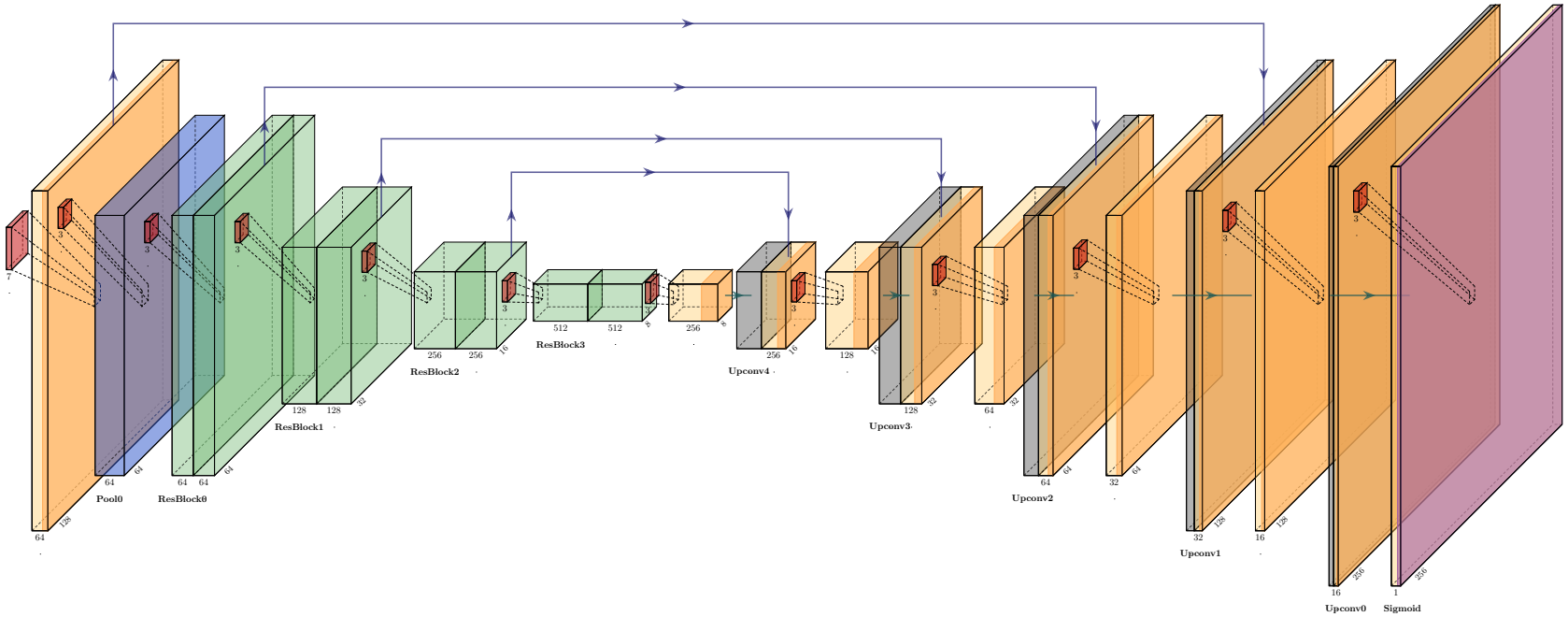
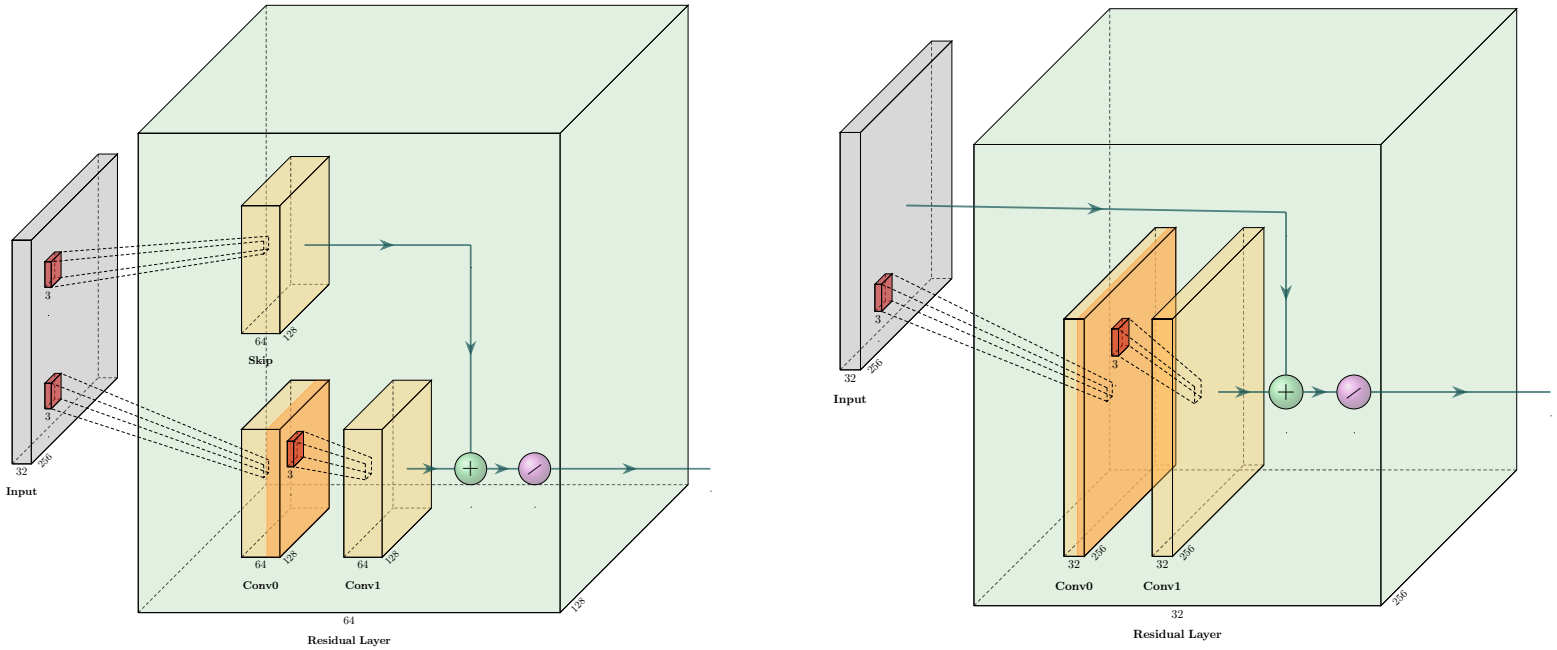



Figure A.4: Architecture of the network used for depth estimation in *MonoDepth2*. It is a lightweight variant compared to *GeoNet*. The first convolution layer uses a filter stride of 2. The max pooling layer (blue) halves the input dimensions followed by a sequence of blocks of residual layers with filter stride of 2 fo ResBlock1 to ResBlock3 (green, see Figure A.5 for details). The upconvolution layer are a combination of an upsampling by nearest neighbor (grey) and a convolutional layer with a filter size of 3 and stride of 1. A second convolutional layer is applied on the channel-wise concatenation of the upconvolution and the outputs of previous residual blocks at the same scale. All layer use *ReLu* activation.




(a) (b)

Figure A.5: *MonoDepth2* comprises two residual layer types of which (a) applies to the first layer of ResBlock1 to ResBlock3 and (b) applies to all remaining layers. (a) Layer with skip convolution and a filter stride of 2. Skip and Conv1 are without activation function while the sum of both outputs is passed to a *ReLu* function. (b) Layer with direct skip connection. In this type each convolution preserves the input dimensions.

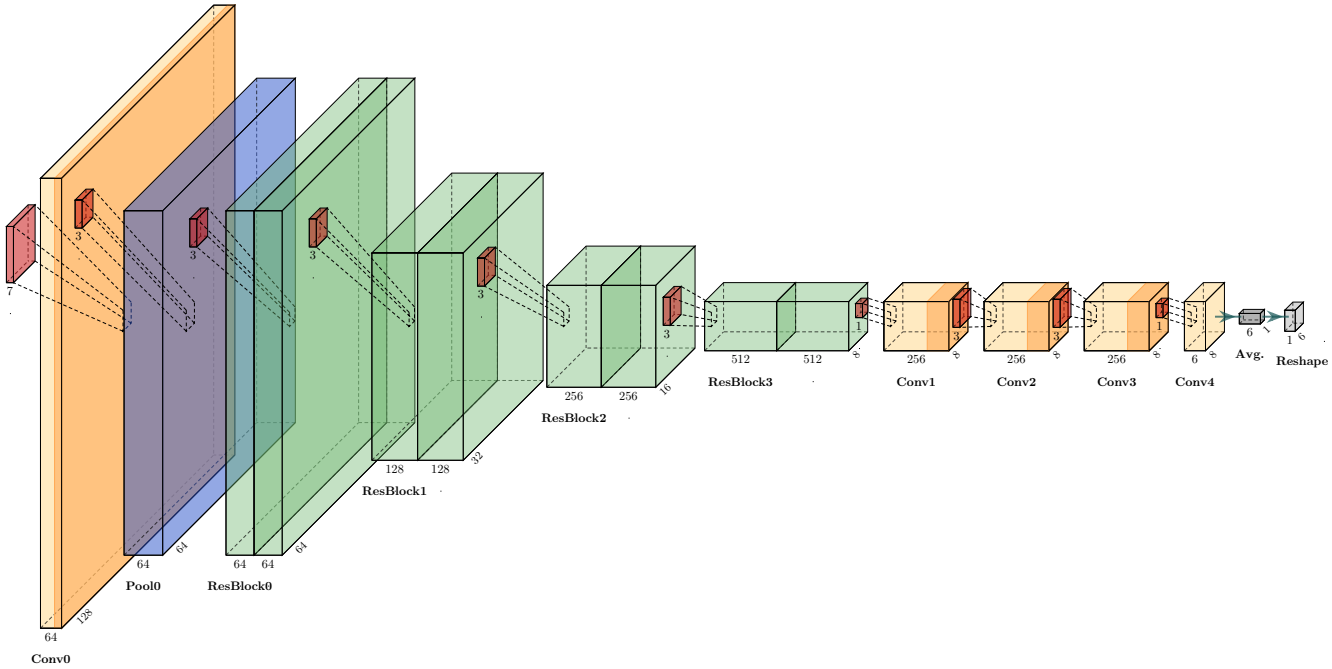



Figure A.6: Architecture of the network used for pose estimation in *MonoDepth2*. The encoder is the same as in the depth network (see Figure A.4). The output is a vector with 6 elements describing translation and rotation of the viewpoint between two adjacent frames.

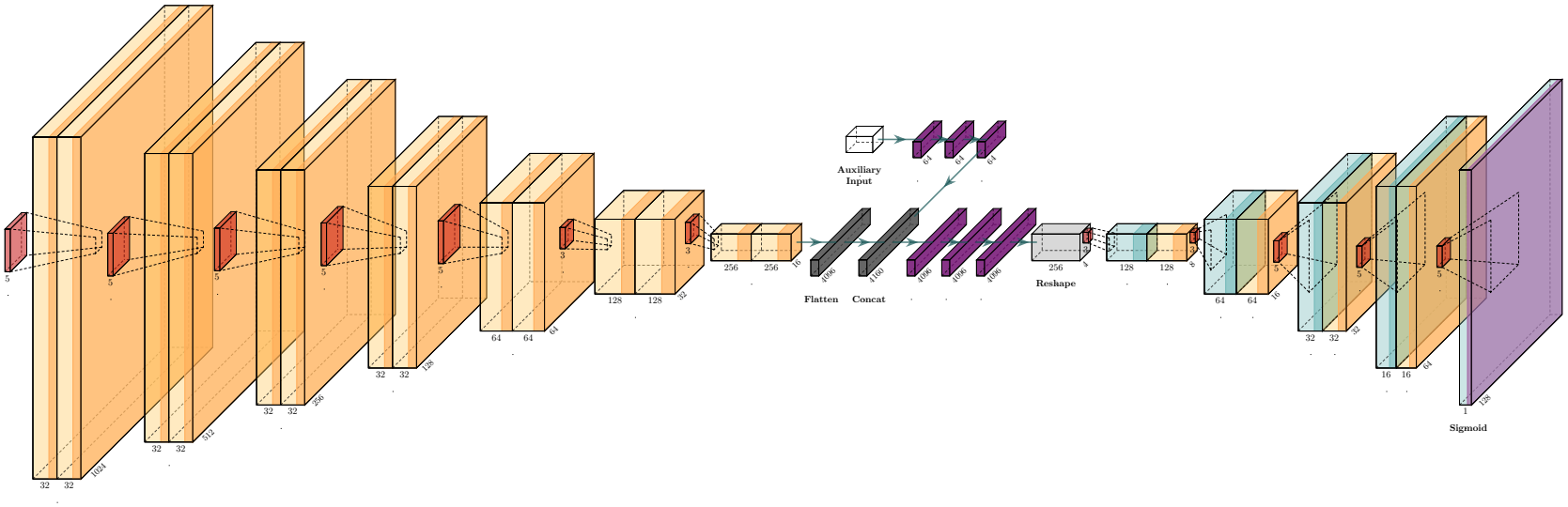



Figure A.7: Network architecture of the projection stage. A convolutional unit consist of two convolutional layers of which the first has a filter stride of 2 and the second of 1 at the same filter size. The inner unit comprises three fully connected layer (purple) and an auxiliary input for the environmental parameters. A deconvolution unit consist of a deconvolutional layer (teal) with filter stride 2 and a convolutional layer with stride 1. The last layer is a single deconvolutional layer with *sigmoid* activation. Fully connected layer use *tanh* activation while all remaining layer use leaky *ReLu* with 0.2 leakage.

# Nomenclature

| | |
|---|---|
| $h$ | Base height of cloud $(m)$ |
| $h_{min}$ | Minimum base height of cloud $(m)$ |
| $h_{max}$ | Maximum base height of cloud $(m)$ |
| $h_0$ | Surface height $(m)$ |
| $D_t$ | Depth map to image at time step $t$ |
| $D'_t$ | Predicted depth map to image at time step $t$ |
| $E_{\text{mse}}$ | Mean squared error |
| $E_{\text{rmse}}$ | Root mean squared error |
| $FS$ | Forecast skill |
| $I_t$ | Image frame at time $t$ |
| $I'_t$ | Predicted image frame at time $t$ |
| $\kappa$ | Cloud size scaling factor |
| $\mathcal{L}_{L2}$ | $L2$ loss |
| $\mathcal{L}_{\text{pm}}$ | Photo-metric loss |
| $\mathcal{L}_{\text{ds}}$ | Depth smoothness loss |
| $N_C$ | Number of clouds in one frame |
| $N_I$ | Number of simulated images |
| $\Omega$ | Environmental parameters |
| $\Delta\varphi_I$ | Delta of image rotation angle |
| $\varphi_{\text{sun}}$ | Azimuth angle sun $(°)$ |
| $\varphi_{\text{wind}}$ | Azimuth angle wind $(°)$ |
| $R$ | Radius for cloud center placement from camera $(m)$ |
| $R_{max}$ | Maximum radius for cloud center placement $(m)$ |
| $R_{min}$ | Minimum radius for cloud center placement $(m)$ |
| $S_t$ | Shadow map at time step $t$ |
| $S'_t$ | Predicted shadow map at time step $t$ |
| $SSIM$ | Structural similarity index |
| $T$ | Total number of time steps |
| $\Delta t$ | Nowcasting lead time steps |
| $\theta_{\text{sun}}$ | Zenith angle sun $(°)$ |
| $v_0$ | Surface wind speed at surface height $h_0$ $(m/s)$ |

# Abbreviations

| | |
|---|---|
| 2D | two-dimensional |
| 3D | three-dimensional |
| Ac | Altocumulus |
| As | Altostratus |
| Cb | Cumulunimbus |
| Cc | Cirrocumulus |
| Ci | Cirrus |
| CNN | Convolutional neural network |
| Cs | Cirrostratus |
| CSP | Concentrated Solar Power |
| Cu | Cumulus |
| DHI | Diffuse Horizontal Irradiance |
| DNI | Direct Normal Irradiance |
| e.g. | exempli gratia |
| GHI | Global Horizontal Irradiance |
| GHU | Gradient Highway Unit |
| i.e. | id est |
| LSTM | Long Short-term Memory |
| NN | Neural Network |
| Ns | Nimbostratus |
| NWP | Numerical Weather Prediction |
| PV | Photovoltaic |
| RNN | Recurrent Neural Network |
| Sc | Stratocumulus |
| St | Stratus |
| TSI | Total Sky Imagery |

# References


[1] UNFCCC. *Status of Treaties - 7.d Paris Agreement*. accessed 28-October-2019. https://treaties.un.org/Pages/ViewDetails.aspx?src=TREATY&mtdsg_no=XXVII-7-d&chapter=27&clang=_en, 2015.

[2] REN21. *Renewables 2017 Global Status Report*. https://www.ren21.net/wp-content/uploads/2019/05/GSR2017_Full-Report_English.pdf: REN21, 2017.

[3] REN21. *Renewables 2018 Global Status Report*. https://www.ren21.net/wp-content/uploads/2019/08/Full-Report-2018.pdf: REN21, 2018.

[4] REN21. *Renewables 2019 Global Status Report*. https://www.ren21.net/wp-content/uploads/2019/05/gsr_2019_full_report_en.pdf: REN21, 2019.

[5] Askari Mohammad Bagher, Mirzaei Mahmoud Abadi Vahid, and Mirhabibi Mohsen. "Types of solar cells and application". In: *American Journal of Optics and Photonics* 3.5 (2015), pp. 94–113.

[6] B. Hoffschmidt et al. "3.06 - High Concentration Solar Collectors". In: *Comprehensive Renewable Energy*. Ed. by Ali Sayigh. Oxford: Elsevier, 2012, pp. 165–209. ISBN: 978-0-08-087873-7. DOI: https://doi.org/10.1016/B978-0-08-087872-0.00306-1. URL: http://www.sciencedirect.com/science/article/pii/B9780080878720003061.

[7] Jan Kleissl. *Solar Energy Forecasting and Resource Assessment*. Academic Press, 2013.

[8] Cody A Hill et al. "Battery energy storage for enabling integration of distributed solar power generation". In: *IEEE Transactions on smart grid* 3.2 (2012), pp. 850–857.

[9] Maimouna Diagne et al. "Review of solar irradiance forecasting methods and a proposition for small-scale insular grids". In: *Renewable and Sustainable Energy Reviews* 27 (2013), pp. 65–76.

[10] Benjamin Kroposki et al. "Achieving a 100% renewable grid: Operating electric power systems with extremely high levels of variable renewable energy". In: *IEEE Power and Energy Magazine* 15.2 (2017), pp. 61–73.

[11] Chi Wai Chow et al. "Intra-hour forecasting with a total sky imager at the UC San Diego solar energy testbed". In: *Solar Energy* 85.11 (2011), pp. 2881–2893.

[12] Handa Yang et al. "Solar irradiance forecasting using a ground-based sky imager developed at UC San Diego". In: *Solar Energy* 103 (2014), pp. 502–524.

[13] Philippe Blanc et al. "Short-term forecasting of high resolution local DNI maps with multiple fish-eye cameras in stereoscopic mode". In: *AIP conference Proceedings*. Vol. 1850. 1. AIP Publishing. 2017, p. 140004.

[14] Z22. *Parabolic at Harper Lake in California*. accessed 03-March-2020. https://commons.wikimedia.org/wiki/File:Parabolic_trough_at_Harper_Lake_in_California.jpg, 2013.

[15] afloresm. *PS10 Solar Power Tower*. accessed 03-March-2020. https://commons.wikimedia.org/wiki/File:PS10_solar_power_tower.jpg, 2007.

[16] Ceinturion. *Solar Power Plant Serpa*. accessed 03-March-2020. https://commons.wikimedia.org/wiki/File:SolarPowerPlantSerpa.jpg, 2006.

[17] Rebecca Lindsey. *Climate and Earth's Energy Budget*. accessed 03-March-2020. https://earthobservatory.nasa.gov/features/EnergyBalance, 2009.

[18] National Renewable Energy Laboratory (NREL). *Solar Resource Glossary*. accessed 03-March-2020. https://www.nrel.gov/grid/solar-resource/solar-glossary.html.

[19] World Metorological Organisation. *Definitions of clouds*. accessed 30-October-2019. https://cloudatlas.wmo.int/definitions-of-clouds.html, 2017.

[20] World Metorological Organisation. *Genera of clouds*. accessed 30-October-2019. https://cloudatlas.wmo.int/clouds-definitions.html, 2017.

[21] Dorota Matuszko. "Influence of the extent and genera of cloud cover on solar radiation intensity". In: *International Journal of Climatology* 32.15 (2012), pp. 2403–2414.

[22] Jürgen Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural networks* 61 (2015), pp. 85–117.

[23] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[24] Chris Olah. *Understanding LSTM Networks*. accessed 16-March-2020. https://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015.

[25] Thomas Schmidt et al. "Evaluating the spatio-temporal performance of sky-imager-based solar irradiance analysis and forecasts". In: *Atmospheric chemistry and physics* 16.5 (2016), pp. 3399–3412.

[26] Ricardo Marquez and Carlos FM Coimbra. "Intra-hour DNI forecasting based on cloud tracking image analysis". In: *Solar Energy* 91 (2013), pp. 327–336.

[27] Rémi Chauvin et al. "Cloud detection methodology based on a sky-imaging system". In: *Energy Procedia* 69 (2015), pp. 1970–1980.

[28] Yinghao Chu et al. "A smart image-based cloud detection system for intrahour solar irradiance forecasts". In: *Journal of Atmospheric and Oceanic Technology* 31.9 (2014), pp. 1995–2007.

[29] Andreas Kazantzidis et al. "Cloud detection and classification with the use of whole-sky ground-based images". In: *Atmospheric Research* 113 (2012), pp. 80–88.

[30] Qingyong Li, Weitao Lu, and Jun Yang. “A hybrid thresholding algorithm for cloud detection on ground-based color images”. In: *Journal of atmospheric and oceanic technology* 28.10 (2011), pp. 1286–1296.

[31] Nobuyuki Otsu. “A threshold selection method from gray-level histograms”. In: *IEEE transactions on systems, man, and cybernetics* 9.1 (1979), pp. 62–66.

[32] MS Ghonima et al. “A method for cloud detection and opacity classification based on ground based sky imagery”. In: *Atmospheric Measurement Techniques* 5.11 (2012), pp. 2881–2892.

[33] Hao Huang et al. “Cloud motion estimation for short term solar irradiation prediction”. In: *2013 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. IEEE. 2013, pp. 696–701.

[34] Zhenzhou Peng et al. “3D cloud detection and tracking system for solar forecast using multiple sky imagers”. In: *Solar Energy* 118 (2015), pp. 496–519.

[35] Pierre Ineichen. “Comparison of eight clear sky broadband models against 16 independent data banks”. In: *Solar Energy* 80.4 (2006), pp. 468–478.

[36] Mehdi Aakroum et al. “Deep Learning for Inferring the Surface Solar Irradiance from Sky Imagery”. In: *2017 International Renewable and Sustainable Energy Conference (IRSEC)*. IEEE. 2017, pp. 1–4.

[37] Yuchi Sun, Gergely Szűcs, and Adam R Brandt. “Solar PV output prediction from video streams using convolutional neural networks”. In: *Energy & Environmental Science* 11.7 (2018), pp. 1811–1818.

[38] Anto Ryu et al. “Preliminary Analysis of Short-term Solar Irradiance Forecasting by using Total-sky Imager and Convolutional Neural Network”. In: *2019 IEEE PES GTD Grand International Conference and Exposition Asia (GTD Asia)*. IEEE. 2019, pp. 627–631.

[39] Dan Tulpan et al. “Detection of clouds in sky/cloud and aerial images using moment based texture segmentation”. In: *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE. 2017, pp. 1124–1133.

[40] Soumyabrata Dev et al. “CloudSegNet: A Deep Network for Nychthemeron Cloud Image Segmentation”. In: *IEEE Geoscience and Remote Sensing Letters* (2019).

[41] Soumyabrata Dev et al. “Multi-label cloud segmentation using a deep network”. In: *arXiv preprint arXiv:1903.06562* (2019).

[42] Jerome Revaud et al. “Epicflow: Edge-preserving interpolation of correspondences for optical flow”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1164–1172.

[43] Eddy Ilg et al. “Flownet 2.0: Evolution of optical flow estimation with deep networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2462–2470.

[44] Zhichao Yin and Jianping Shi. "Geonet: Unsupervised learning of dense depth, optical flow and camera pose". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1983–1992.

[45] Xu Jia et al. "Dynamic filter networks". In: *Advances in Neural Information Processing Systems*. 2016, pp. 667–675.

[46] Ruben Villegas et al. "Learning to generate long-term future via hierarchical prediction". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 3560–3569.

[47] Viorica Patraucean, Ankur Handa, and Roberto Cipolla. "Spatio-temporal video autoencoder with differentiable memory". In: *arXiv preprint arXiv:1511.06309* (2015).

[48] Shi Xingjian et al. "Convolutional LSTM network: A machine learning approach for precipitation nowcasting". In: *Advances in neural information processing systems*. 2015, pp. 802–810.

[49] William Lotter, Gabriel Kreiman, and David Cox. "Deep predictive coding networks for video prediction and unsupervised learning". In: *arXiv preprint arXiv:1605.08104* (2016).

[50] Nal Kalchbrenner et al. "Video pixel networks". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 1771–1779.

[51] Yunbo Wang et al. "Predrnn++: Towards a resolution of the deep-in-time dilemma in spatiotemporal predictive learning". In: *arXiv preprint arXiv:1804.06300* (2018).

[52] Jiajun Wu et al. "Single image 3d interpreter network". In: *European Conference on Computer Vision*. Springer. 2016, pp. 365–382.

[53] Jimei Yang et al. "Weakly-supervised disentangling with recurrent transformations for 3d view synthesis". In: *Advances in Neural Information Processing Systems*. 2015, pp. 1099–1107.

[54] Xinchen Yan et al. "Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision". In: *Advances in Neural Information Processing Systems*. 2016, pp. 1696–1704.

[55] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. "Multi-view 3d models from single images with a convolutional network". In: *European Conference on Computer Vision*. Springer. 2016, pp. 322–337.

[56] Tinghui Zhou et al. "View synthesis by appearance flow". In: *European conference on computer vision*. Springer. 2016, pp. 286–301.

[57] Eunbyung Park et al. "Transformation-grounded image generation network for novel 3d view synthesis". In: *Proceedings of the ieee conference on computer vision and pattern recognition*. 2017, pp. 3500–3509.

[58] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. “On the difficulty of training recurrent neural networks”. In: *International conference on machine learning*. 2013, pp. 1310–1318.

[59] Clément Godard et al. “Digging into Self-Supervised Monocular Depth Prediction”. In: *The International Conference on Computer Vision (ICCV)*. Oct. 2019.

[60] Andreas Geiger et al. “Vision meets Robotics: The KITTI Dataset”. In: *International Journal of Robotics Research (IJRR)* (2013).

[61] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[62] Zhou Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.

[63] Andrej Karpathy et al. “Large-scale video classification with convolutional neural networks”. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2014, pp. 1725–1732.

[64] Ag2gaeh. *Kugelkoordinaten: Definition*. accessed 06-Feburary-2020. `https://commons.wikimedia.org/wiki/File:Kugelkoord-def.svg`, 2015.

[65] John S Irwin. “A theoretical variation of the wind profile power-law exponent as a function of surface roughness and stability”. In: *Atmospheric Environment (1967)* 13.1 (1979), pp. 191–194.

[66] Ltd. Entaniya Co. *Super Wide Fisheye Lens for Board camera Machine Vision P0.5/M12*. accessed 06-Feburary-2020. `https://products.entaniya.co.jp/en/products/m12-fisheye/`, 2015.

[67] Sigma Corporation. *Sigma 4.5mm f2.8 EX DC HSM*. accessed 06-Feburary-2020. `https://www.sigma-global.com/en/lenses/others/wide/45_28/`, 2007.

[68] Ciarán Hughes et al. “Accuracy of fish-eye lens models”. In: *Applied optics* 49.17 (2010), pp. 3338–3347.

[69] Taco S Cohen et al. “Spherical CNNs”. In: *arXiv preprint arXiv:1801.10130* (2018).

[70] Benjamin Coors, Alexandru Paul Condurache, and Andreas Geiger. “Spherenet: Learning spherical representations for detection and classification in omnidirectional images”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 518–533.

[71] Juyang Weng, Paul Cohen, and Marc Herniou. “Camera calibration with distortion models and accuracy evaluation”. In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 10 (1992), pp. 965–980.

[72] Zhengyou Zhang. “A flexible new technique for camera calibration”. In: *IEEE Transactions on pattern analysis and machine intelligence* 22.11 (2000), pp. 1330–1334.

[73] OpenCV. *OpenCV 3.4 Documentation*. accessed 10-Feburary-2020. https://docs.opencv.org/3.4/db/d58/group__calib3d__fisheye.html, 2019.

[74] Ricardo Marquez and Carlos FM Coimbra. "Proposed metric for evaluation of solar forecasting models". In: *Journal of solar energy engineering* 135.1 (2013), p. 011016.

[75] Michael Mathieu, Camille Couprie, and Yann LeCun. "Deep multi-scale video prediction beyond mean square error". In: *arXiv preprint arXiv:1511.05440* (2015).

[76] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. "Spatial transformer networks". In: *Advances in neural information processing systems*. 2015, pp. 2017–2025.

[77] P. Crispel and G. Roberts. "All-sky photogrammetry techniques to georeference a cloud field". In: *Atmospheric Measurement Techniques* 11.1 (2018), pp. 593–609. DOI: 10.5194/amt-11-593-2018. URL: https://www.atmos-meas-tech.net/11/593/2018/.