Workshop in Deep Learning
Cloud Forecasting Transformer

Leah London Arazi, Ofir Gaash

October 2023

Abstract

Weather forecasting is one of the many areas in which Deep Learning provides potential for new capabilities. In particular, it is a promising candidate for Nowcasting, short-term weather prediction, a field that requires faster computation than traditional methods. The Israeli Meteorological Service is interested in forecasting the position of clouds in the near future to aid routine operations and decision making in solar power plants. A recent work named Earthformer, presents a Transformer-based architecture for precipitation nowcasting. We adopt this architecture for cloud cover nowcasting. Our main contribution is training Earthformer on our original dataset of satellite images while using perceptual loss which enhances the model's outputs in terms of sharpness. Furthermore, we investigate our own novel loss function named FSS in order to improve the predication accuracy.

1 Introduction

Nowcasting is a field of meteorology which aims at forecasting weather for a short term of up to a few hours. Cloud cover nowcasting refers to predicting the position of clouds at a certain area in the near future. Cloud cover nowcasting provides crucial information for the management and decision making at solar power plants. The field of Renewable Energy Sources (RES) has great importance due to climate change. In Israel, solar energy has the most potential out of all natural energy resources.

While long term forecasting is achievable by the traditional use of physics-driven models, the long computation time makes them irrelevant for short term forecasting [1]. In recent years the success of deep learning and the explosive growth of Earth observation data has resulted in the adoption of deep learning models for various Earth system forecasting tasks.

Predicting cloud cover is a hard task. There is a variety of cloud types, different in their height, composition and texture. There can be more than one cloud in the same area, causing the cloud movement to look more chaotic. Clouds do not only move and change their shape, but also form and disappear, while our model claims to only detect movement. The behavior of clouds is affected by the season, the time of day and the topography of the area (for example clouds tend to stop near mountains and form near beaches).

In our project we attempt to enhance the Earthformer [2], a recent promising architecture based on Transformers, originally created for rainfall nowcasting. We hope that despite the complexity of the data, our model will provide a sufficient solution to the problem when given many examples. We examine a few loss functions in order to improve the model and train it on satellite images from the Israeli Meteorological Service (IMS). The IMS allowed us to access satellite data and provided us with domain knowledge.

From the user endpoint, our end product is a trained cloud cover prediction model: the input is a sequence of 13 satellite images taken 15 minutes apart, the output is a sequence of 12

images that represents the prediction of the next satellite images following the input sequence.

2 Related Work

2.0.1 Previous Works

Due to the spatial and temporal patterns of the relevant data, most of the early works are based on CNNs [3] [4] or RNNs [5]. One of the previous works, which we use as a baseline, suggests an architecture called PredRNN [6]. This work tries to jointly model the spatial correlations and temporal dynamics at different levels of RNNs, by extending the inner-layer transition function of memory states in LSTMs to spatiotemporal memory flow.

More recent works are based on Transformers, an emerging deep learning architecture, which despite its broad success in other domains has a limited adoption in this area. Another baseline we use is named Rainformer [7], a precipitation nowcasting framework. The work proposes two practical components: the global features extraction unit that provides robust features learning ability, and the gate fusion unit (GFU) that combines the local and the global features.

2.0.2 Earthformer

Earthformer [2] is a recent work that uses a Transformer-based architecture for precipitation nowcasting. The input data is a series of images $\{X_i\}_{i=1}^{t_{in}}, X_i \in \mathbb{R}^{H \times W \times C}$, and the output is a series of predicted images $\{\hat{Y}_i\}_{i=1}^{t_{out}}, \hat{Y}_i \in \mathbb{R}^{H \times W \times C}$ of the same dimensions. As the Transformer model relies on the concept of *attention*, there was a need to adapt the attention mechanism to fit 3-Dimensional visual input. A naive adaptation is considered computationally infeasible because of the high dimensionality of the input. The paper proposes a framework called *cuboid-attention* that allows to perform efficient space-time attention. The general idea is to decompose the input, which can be portrayed as a rectangular cuboid, into "small" non-overlapping cuboids. We apply self-attention within each cuboid. Finally, the cuboids are merged back to get the original dimensions using the inverse of the decomposition process. The cuboid attention significantly reduces the computational cost, but it only allows us to learn local relationships. To compensate for this, the paper introduces the idea of *global vectors*, inspired by the [CLS] token adopted in BERT.

3 Method

For the rest of this section we denote the input sequence by $\{X_i\}_{i=1}^{t_{in}}, X_i \in \mathbb{R}^{H \times W \times C}$, the target sequence by $\{Y_i\}_{i=1}^{t_{out}}, Y_i \in \mathbb{R}^{H \times W \times C}$ and the predicted sequence by $\{\hat{Y}_i\}_{i=1}^{t_{out}}, \hat{Y}_i \in \mathbb{R}^{H \times W \times C}$.

3.1 Datasets

We created our own novel dataset for this project. Our data consists of satellite images provided by EUMETSAT, the European operational satellite agency for monitoring weather, climate and the environment from space. There are many types of satellite images which are based on measuring a variety of wavelengths and performing different post-processing[1]. We collected two types of grayscale satellite images called VIS (Visible) and IR (Infra-Red). VIS imagery is produced by the sun's rays reflecting off of clouds. IR imagery is produced by sensing the emitted radiation coming off of clouds, which is determined by their temperature [8]. Both types have advantages and disadvantages regarding cloud detection; On the one hand, VIS

[1] https://eumetview.eumetsat.int/static-images/MSG/IMAGERY/

shows cloud texture better, however VIS images are not available during nighttime. On the other hand, IR is not limited to daytime and allows to get a general idea of the height of the clouds, yet its resolution is worse.

We collected 350GB of IR images and 140GB of VIS images. The data is organized as chronological events saved in HDF5[2] files. Each event is composed of 45 grayscale frames taken 15 minutes apart. In addition, the dataset has a catalog file in a CSV format which stores metadata on the events. The catalog lists for each event the index of the event's frames in the relevant .h5 file. Each catalog entry also includes the start time of the event and its image type ("MIDDLE_EAST_IR" or "MIDDLE_EAST_VIS"). The structure of the dataset was inspired by the SEVIR dataset [9].

We started by extracting 24-hour events. During the handling of the raw images we noticed that there are arbitrarily missing frames. Instead of dropping data, we adjusted the extraction to have smaller granularity so we would be able to partition the existing subsequent frames to smaller chunks. After trial and error we settled on an event length of 45 frames.

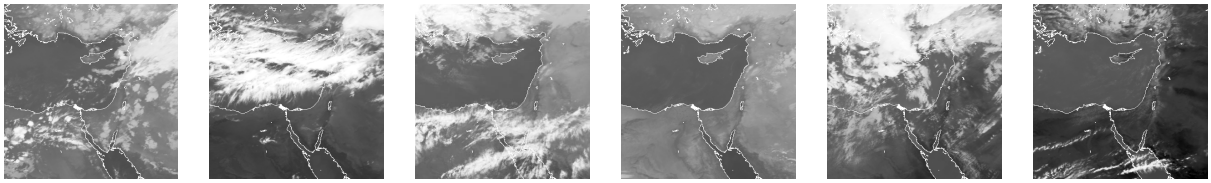

Table 1: IR Images

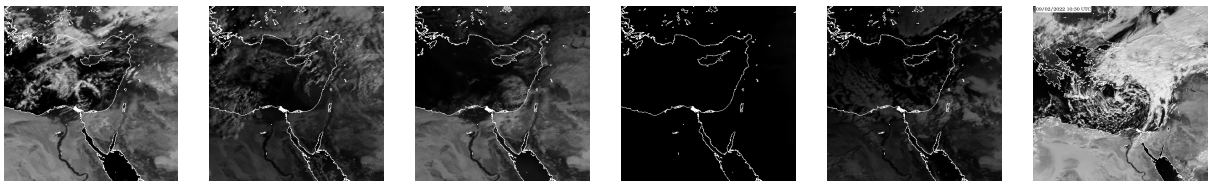

Table 2: VIS Images

We wrote a dataloader in order to allow configurable pre-processing of the images and flexibility of the sequences (for example, the number of frames and the time delta between two subsequent frames). We crop the raw images (600 x 600 pixels) to manageable images (384 x 384 pixels) in terms of compute while concentrating on the geographical area where we want to predict. Each event is divided to possibly more then one sequences. Following SEVIR dataloader, the sequences are partially overlapping. The dataloader has a parameter named "stride" that determines the size of the overlap, as described below:

$$
\begin{array}{l}
[-----------------------raw\_seq\_len-----------------------] \\
[-----seq\_len-----] \\
<--stride-->[-----seq\_len-----] \\
\qquad\qquad\qquad\quad <--stride-->[-----seq\_len-----]
\end{array}
$$

Since our input sequence length is 13 and the output sequence length is 12 ($seq\_len = 25$), we choose the stride to be 12, so a sequence's output will not overlap with the previous sequence.

[2]Hierarchical Data Formats (HDF) are used to store large amount of data for quick retrieval and analysis.

3.2 Loss Functions

Our loss function is the following:

$$\mathcal{L}(Y, \hat{Y}) = \lambda_{MSE} \cdot MSE(Y, \hat{Y}) + \lambda_{LPIPS} \cdot LPIPS(Y, \hat{Y}) + \lambda_{FSS} \cdot FSS(Y, \hat{Y}; \tau, n) \quad (1)$$

Where $\lambda_{MSE}, \lambda_{LPIPS}, \lambda_{FSS}, \tau, n$ are training hyper parameters.

3.2.1 MSE (L2)

This loss is a common pixel-to-pixel loss used in vision tasks. MSE is defined as follows:

$$MSE(Y, \hat{Y}) = \frac{1}{t_{out} \cdot H \cdot W \cdot C} \sum_{i=1}^{t_{out}} \sum_{h,c,w=1}^{H,C,W} ((\hat{Y}_i)_{h,w,c} - (Y_i)_{h,w,c})^2. \quad (2)$$

In the Earthformer model the training was solely based on optimizing the MSE loss. We noticed that using only MSE resulted in blurry predictions. In addition, MSE has no special relevance for our specific use case, since an important goal is to output an accurate cloud cover prediction.

3.2.2 Perceptual Loss (LPIPS)

The L2 loss, like other per-pixel losses, does not capture perceptual, "human-like" observed differences between images [10]. Perceptual loss leverages a pre-trained general-purpose image classification network. Most commonly, the loss is computed using the following process: first, the predicted and the target images are forward-passed through the pre-trained network. Then, the L2 distances between the activations of the hidden layers are calculated and summed.

The perceptual loss implementation we used is called LPIPS [11]. Specifically, we used LPIPS with a VGG network between $\{Y_i\}_{i=1}^{t_{out}}$ and $\{\hat{Y}_i\}_{i=1}^{t_{out}}$. The motivation behind using this loss is to generate sharper images in order to improve the quality of the prediction.

3.2.3 Fractional Skill Score (FSS)

FSS is a verification metric wildly used in weather prediction. The purpose of this verification method is to obtain a measure of how forecast skill varies depending on the spatial scale. Intuitively, FSS measures the similarity of the observed and the forecast fields[3] in a scale of size n.

Let us present the calculation of FSS as described in [12, pp. 80-82], [13]: We denote the observed field as O_r and the forecast field as M_r.

1. Convert O_r, M_r to binary fields: we choose a *threshold*, denoted as τ[4], and calculate:

$$I_O = \begin{cases} 1, & O_r \geq \tau, \\ 0, & O_r < \tau. \end{cases} \qquad I_M = \begin{cases} 1, & M_r \geq \tau, \\ 0, & M_r < \tau. \end{cases} \quad (3)$$

2. Generate fractions: for every point in the binary fields obtained in step 1, we compute the fraction of surrounding points within a given $n \times n$ square that have a value of 1. We call this square the *neighborhood* of size n of this point.

$$O_{(n)}(i,j) = \frac{1}{n^2} \sum_{k=1}^{n} \sum_{l=1}^{n} I_O\left[i + k - 1 + \frac{n-1}{2}, j + l - 1 + \frac{n-1}{2}\right], \quad (4)$$

[3] In our context the observed field is the target sequence and the forecast field is the predicted sequence.

[4] In the original paper multiple thresholds can be chosen but in our case we use only one threshold that represents if a cloud is present or not.

$$M_{(n)}(i,j) = \frac{1}{n^2}\sum_{k=1}^{n}\sum_{l=1}^{n} I_M\Big[i+k-1+\frac{n-1}{2}, j+l-1+\frac{n-1}{2}\Big]. \tag{5}$$

The equations above describe the resultant field of observed fractions for neighborhoods of size n.

3. Compute the fractional skill score: we denote the number of possible neighborhoods in the direction of the x-axis and y-axis by N_x, N_y respectively. We finally calculate:

$$FSS_{(n)} = 1 - \frac{\sum_{i=1}^{N_x}\sum_{j=1}^{N_y}(O_{(n)}(i,j) - M_{(n)}(i,j))^2}{\sum_{i=1}^{N_x}\sum_{j=1}^{N_y}(O_{(n)}(i,j))^2 + \sum_{i=1}^{N_x}\sum_{j=1}^{N_y}(M_{(n)}(i,j))^2} \tag{6}$$

Although the classic use of FSS is as a verification metric, we thought it would be interesting and hopefully helpful to use FSS as a one of our loss functions. We see a mathematical reasoning behind it; The network's goal is to predict the presence of a cloud in each pixel. Since FSS binarizes the pixels, the loss will become indifferent to inaccurate colors but more sensitive to wrong predictions[5]. In addition, we want to test different scales and investigate their effect on the perceptual quality of the predicted images.

The adaptation of FSS as a loss is not trivial since it is not naturally differentiable. Therefore, we implement an approximate version of the field binarization described in step 1. This process can be thought of as passing the input field through an element-wise threshold function. We use the differentiable *tanh* function as our threshold function. Given a smoothing factor s[6], the calculation is described as follows:

$$I_O = \frac{1}{2}\cdot tanh(s\cdot(O_r - \tau)) + \frac{1}{2} \qquad I_M = \frac{1}{2}\cdot tanh(s\cdot(M_r - \tau)) + \frac{1}{2} \tag{7}$$

Another notable implementation detail is the usage of a convolution filter in order to generate the fractions as described in step 2.

Moreover, optimizing only FSS loss required adaptations. Apparently, in this scenario the training suffers from vanishing gradients, since the loss remains constant. Our investigation shows that the binarization of the data and the relatively high threshold $\tau = 160$ for cloud detection cause the initial model's random (and relatively small) outputs to constantly be below the threshold. According to the FSS calculation we get a score of 0 in every step. Our solution to the problem was optimizing with MSE for one epoch to stabilize the outputs and then optimizing only with FSS.

3.3 Evaluation Method

Our evaluation method uses three evaluation metrics; one evaluates the forecast quality (CSI) and the others evaluate pixel-to-pixel similarity (MSE, MAE).

CSI (Critical Success Index) is a method to evaluate the accuracy of a spatial forecast [14] [15]. Given the threshold τ, we binarize both the target and forecast images, and label each pixel as a "hit" if both target and forecast are above τ, "miss" if the target is above τ and the forecast is below τ, and "false alarm" if the target is below τ and the forecast is above τ. We then calculate:

$$CSI(Y_i, \hat{Y}_i, \tau) = \frac{\#hits}{\#hits + \#misses + \#false\_alarms} \tag{8}$$

[5]Note that color accuracy does improve the mCSI metric score presented in Section 3.3. In this project we wanted to inspect whether improving accuracy in one of the mCSI's thresholds will improve the total mCSI score.

[6]This is a hyperparameter of the training.

We average the CSI over a list of thresholds denoted $T = \{145, 160, 180, 200, 220\}$. The final CSI, denoted as $mCSI(Y, \hat{Y}, T)$, is averaged over every $\tau \in T$ and the entire sequence. The thresholds were chosen manually by us based on the pixel values of clouds in IR images.

We use the classic implementation of MSE (L2) and MAE (L1) between Y and $\hat{Y}$. These metrics evaluate the model's ability to predict small details and accurate shapes of clouds. Since MSE is sensitive to outliers, we use both MSE and MAE.

We calculate the validation loss according to this formula:

$$\mathcal{L}_{val} = mCSI(Y, \hat{Y}, T) + 10 \cdot MAE(Y, \hat{Y}) + 100 \cdot MAE(Y, \hat{Y}) \tag{9}$$

3.4 Training and Inference

Each example in both image types is a sequence of 25 frames taken 15 minutes apart (13 input frames and 12 output frames). Each frame in the sequence is a grayscale image of size 384 x 384. In all of our experiments, we use the AdamW optimizer with a cosine learning rate scheduler. Our total batch size is 16 or 32 depending on the experiment. Compute limitations lead us to use gradient accumulation with micro-batch size 1 or 2.

All experiments are conducted on a local machine[7] ("nala") with a Quadro RTX 6000 GPU. All models, including the baselines, can fit in a single GPU (which has 24GB RAM). We also use a local machine for debugging ("timon") with one Nvidia GeForce RTX 3060 Ti GPU (which has 8GB RAM). Each experiment took roughly 3 days on "nala". We had 4TB of storage space for our dataset, checkpoints and logging, of which we use less than 1TB.

4 Experiments

We run most of our experiments on the IR data. For each experiment we save between 3 and 6 checkpoints with the lowest validation loss. In the experiments tables (Table 3, Table 4, Table 5) the column "epochs" represents the total number of epochs the experiment was run and not the best checkpoint's epoch. Since we had limited resources we run each experiment until there was no improvement in validation loss.

4.1 IR

IR sequences ("MIDDLE_EAST_IR") have a total of 5848 examples. We split the data to 4772 examples for the train set, 396 for the validation set and 680 for the test set. See Table 3 for details.

4.2 VIS

VIS sequences ("MIDDLE_EAST_VIS") have a total of 2342 examples. We split the data to 1826 examples for the train set, 180 for the validation set and 336 for the test set. See Table 4 for details.

4.3 Baselines

For our baselines we use Rainformer [16] and PredRNN [17].

Note that the training of PredRNN is 5 times slower than training our model, which is a significant disadvantage of PredRNN. Consequently, due to our limited resources we might have not trained PredRNN enough to achieve its full potential. See Table 5 for details.

[7] At "Theator" company, see https://theator.io/.

5 Results

We discuss the results of the IR experiments presented in Table 6.

The first experiments we conducted were *lpips_wd* and *mse_wd*. We later noticed that experiments with $wd = 0$ resulted in a more stable training so we ran them for more epochs, what naturally led to better results. Therefore, we decided to run two more experiments; *mse* and *lpips*.

From comparison between *mse* and *lpips*, we conclude that using LPIPS alone leads to a worse CSI score. However, the qualitative results in Table 10 indicate that *lpips* performs better than *mse* in terms of sharpness; the predicted images seem natural and capture the cloud movement better. Following that, we run experiments combining both LPIPS and MSE (*mse_lpips*, *mse_lpips_naive*). When determining solely by the numeric scores, we get that *mse* is the best in terms of CSI and one of the best models overall. We notice that the scores of *mse* are slightly better than *mse_lpips*, however, from inspecting the qualitative results in Table 10 it is clear that LPIPS improves the sharpness of the results.

Using FSS only is (way) worse in terms of MSE and MAE, which makes sense since it's goal is only to indicate whether a cloud is present or not while neglecting the image colors. As seen in Table 11, FSS optimization alone seem to result in static predictions where the clouds do not move, but vary a bit in their size. Between all "FSS-only" experiments we notice that $wd = 0$ and a lower scale achieves better CSI score, so we keep these hyper-parameters for the following experiments.

Combining FSS with LPIPS in fss_lpips_equal resulted in "flickering" predictions as can be seen in Table 15. When using the term "flickering" we refer to the unstable changes in cloud intensity throughout the prediction. Beside the flickering, we note that the movement of the clouds is better perceived than "FSS-only" training. In order to reduce the flickering we conduct three follow-up experiments that tackle the issue from different angles; in fss_lpips we assign a smaller weight to FSS loss, in fss_mse we test whether MSE helps with flickering, and in $fss\_scale = 1\_lpips$ we test whether larger n is the root cause of the flickering. The first follow-up experiment shows no improvement. However, the second and third experiments do eliminate the flickering. In addition, fss_mse outperforms *mse* in its MSE score and $fss\_scale = 1\_lpips$ improves *lpips* overall score without causing blurring, which can be seen in Table 12.

Since MSE helps to eliminate the flickering and LPIPS improves sharpness, we conduct the experiment fss_lpips_mse. Unfortunately we discover that fss_lpips_mse performed worse in terms of CSI than *mse* and *mse_lpips*. To further investigate this, we tested again the experiments that use FSS as a loss function by calculating the CSI score for each threshold individually as can be seen in Table 9. Interestingly, we observe that the main difference between fss_lpips_mse and *mse* or *mse_lpips* is specifically in $\tau = 220$ where fss_lpips_mse performs much worse than the other two. Table 9 also shows that fss_mse is better than *mse* in the lower thresholds but worse in higher thresholds. This shows that FSS may be able to improve the score of a specific threshold when combined with MSE, but does not manage to improve the CSI in general. When combined with LPIPS, FSS seem to cause the opposite effect - fss_lpips is worse than *lpips* in the lower thresholds but better in higher thresholds.

In conclusion, MSE loss preforms the best "metric-wise". LPIPS provides sharp and more realistic predictions. FSS seem to improve results only in specific experiments and not in general. We suspect that further experimenting with the FSS loss can reveal whether it is truly beneficial or not and hopefully explain the above phenomena.

The numerical results of the VIS experiment are presented in Table 7 and the qualitative results are presented in Table 14. Since we conducted only one experiment, we do not have detailed observations to discuss.

The scores of the baselines on our data are presented in Table 8 and the qualitative results are presented in Table 13. We clearly see that the predictions of both baselines are very blurry, especially PredRNN's. The baselines are examples of how a low MSE score is an insufficient metric to indicate the model's performance.

6 Limitation and Future Work

6.1 Limitations

We bring failure cases from different experiments. As we mention in the introduction, there are behaviors of clouds that are challenging to predict, such as the spontaneous formation and deformation of clouds. Indeed, our models do not predict successfully in some of those cases. Examples of an inability to predict clouds creation can be seen in Table 16, Table 17, Table 18, Table 19. The area of the clouds that were missed is surrounded by a red rectangle. Interestingly, we notice that the model succeeds with predicting disappearance of clouds, as can be seen in Table 20.

Furthermore, the model architecture cannot receive additional metadata as input. Hence, there are many factors that affect cloud cover that can not be directly provided as features, such as the time of day and the season, which may be difficult to learn from the input.

6.2 Lessons Learned

As this project is our first practical Deep Learning project, we had to deal with considerations and challenges regarding the training of the model that were not trivial to us:

- Creating a dataset - By creating our own dataset we ran into several challenges. First, making a large dataset available online requires significant resources (which we did not have). To overcome this, we stored the data on a hard disk and connected it directly to the training machine. Second, collecting, cleaning and organizing the data was time consuming and required trial and error. We ended up writing a code to help us create .h5 files from the raw images.

- Evaluating the model's results - In order to create metrics that measure the model's performance in a way that reflects its "goal", we had to acquire domain-specific knowledge. In our case, we were fortunate to cooperate with IMS. However, working with domain experts is something to keep in mind in our future projects.

- Weighing different metrics and losses - In our first experiments we weighted the losses intuitively, for example, to consider MSE and LPIPS equally we set $\lambda_{MSE} = 0.5, \lambda_{LPIPS} = 0.5$. Eventually, by looking at the logs, we noticed that the losses were of different orders of magnitude, so truly considering them equally means to match their orders of magnitude.

6.3 Future Work

Possible future directions regard the architecture of Earthformer. The concept of cuboid-attention is inspiring further research since the 3-Dimensional input decomposition method can be configured using three parameters: "strategy", "shift" and "cuboid size". We can think of exploring different decomposition patterns, which might even be possible using Neural Architecture Search (NAS) [18] techniques. Furthermore, Earthformer paper presents a variant of the architecture we use named Earthformer AR, which uses masked attention to make the decoding process auto-regressive. While the perceptual quality of Earthformer AR was better,

its quantitative performance results were much worse than the non-auto-regressive architecture. This leaves room for future experiments with Earthformer AR.

Future work to enhance the results of this project can be, following advice from IMS, to create a dataset comprising of pairs of sequences of both VIS and IR (and maybe even other image types) and train the model to consider both VIS and IR images in it's prediction. Furthermore, to the best of our knowledge, we are the first to use an evaluation metric from meteorology as a loss function in the area of nowcasting. Hence, we see value in further study of the FSS loss. We can try to optimize an FSS loss with more discretization, meaning, which use a list of thresholds instead of one threshold when binarizing the fields (similar to CSI). Moreover, we can test other weighted combinations of FSS with MSE and LPIPS. Another idea is exploring adaptation of another domain specific metrics as loss functions and hopefully improve the model's CSI score.

A Experiments

	Loss Coefficients			FSS parameters		Training parameters			
	λ_{MSE}	λ_{LPIPS}	λ_{FSS}	τ	n	lr	batch size	epochs	wd
mse_wd	1.0	0	0	-	-	0.0005	32	22	1.0
mse	1.0	0	0	-	-	0.00025	16	43	0
lpips_wd	0	1.0	0	-	-	0.00025	16	38	1.0
lpips	0	1.0	0	-	-	0.00025	16	43	0
mse_lpips_naive	0.5	0.5	0	-	-	0.00025	16	35	0
mse_lpips	10.0	1.0	0	-	-	0.00025	16	44	0
fss_scale=20_wd	0	0	1.0	160	20	0.0005	32	14	1.0
fss_scale=10_wd	0	0	1.0	160	10	0.0005	32	11	1.0
fss_scale=20	0	0	1.0	160	20	0.0005	32	15	0
fss_scale=10	0	0	1.0	160	10	0.0005	32	19	0
fss_scale=5	0	0	1.0	160	5	0.0005	32	21	0
fss_lpips_equal	0	0.1	1.0	160	5	0.00025	16	39	0
fss_lpips	0	0.5	0.5	160	5	0.00025	16	41	0
fss_mse	20.0	0	1.0	160	5	0.00025	16	44	0
fss_scale=1_lpips	0	0.5	0.5	160	1	0.00025	16	27	0
fss_lpips_mse	20.0	2.0	1.0	160	5	0.00025	16	43	0

Table 3: Experiments on IR dataset.

	Loss Coefficients			FSS parameters		Training parameters			
	λ_{MSE}	λ_{LPIPS}	λ_{FSS}	τ	n	lr	batch size	epochs	wd
fss_lpips_mse	20.0	2.0	1.0	160	5	0.00025	16	43	0

Table 4: Experiments on VIS dataset.

	lr	batch size	epochs	wd
PredRNN	0.00001	16	15	0
Rainformer	0.003	24	37	0

Table 5: Baseline experiments on IR dataset.

B Results

	1 - $mCSI$ ↓	MAE ↓	MSE ↓	Overall
mse_wd	0.421	0.067	0.012	0.764
mse	0.297	0.045	0.007	0.476
lpips_wd	0.374	0.066	0.011	0.713
lpips	0.354	0.072	0.029	1.320
mse_lpips_naive	0.352	0.062	0.009	0.611
mse_lpips	0.309	0.049	0.007	0.505
fss_scale=20_wd	0.620	0.184	0.071	3.191
fss_scale=10_wd	0.638	0.181	0.068	3.088
fss_scale=20	0.520	0.218	0.105	4.407
fss_scale=10	0.514	0.225	0.120	4.932
fss_scale=5	0.485	0.291	0.181	7.170
fss_lpips_equal	0.362	0.079	0.016	0.905
fss_lpips	0.348	0.069	0.014	0.819
fss_mse	0.303	0.045	0.006	0.475
fss_scale=1_lpips	0.373	0.066	0.013	0.772
fss_lpips_mse	0.335	0.051	0.007	0.524

Table 6: Experiments results of Cloudformer on IR dataset.

	1 - $mCSI$ ↓	MAE ↓	MSE ↓	Overall
fss_lpips_mse	0.395	0.059	0.011	0.693

Table 7: Experiments results of Cloudformer on VIS dataset.

	1 - $mCSI$ ↓	MAE ↓	MSE ↓	Overall
PredRNN	0.340	0.054	0.008	0.564
Rainformer	0.328	0.053	0.009	0.576

Table 8: Experiments results of baselines on IR dataset.

τ	145 ↓	160 ↓	180 ↓	200 ↓	220 ↓
mse_wd	0.331	0.347	0.390	0.457	0.584
mse	0.242	0.270	0.302	0.322	0.348
lpips_wd	0.285	0.315	0.357	0.399	0.515
lpips	0.280	0.314	0.353	0.382	0.440
mse_lpips	0.244	0.278	0.314	0.339	0.371
fss_lpips_equal	0.328	0.329	0.350	0.388	0.422
fss_lpips	0.311	0.319	0.341	0.369	0.402
fss_mse	0.236	0.269	0.305	0.333	0.374
fss_scale=1_lpips	0.292	0.322	0.363	0.403	0.486
fss_mse_lpips	0.244	0.276	0.315	0.358	0.481

Table 9: CSI comparison of different thresholds.

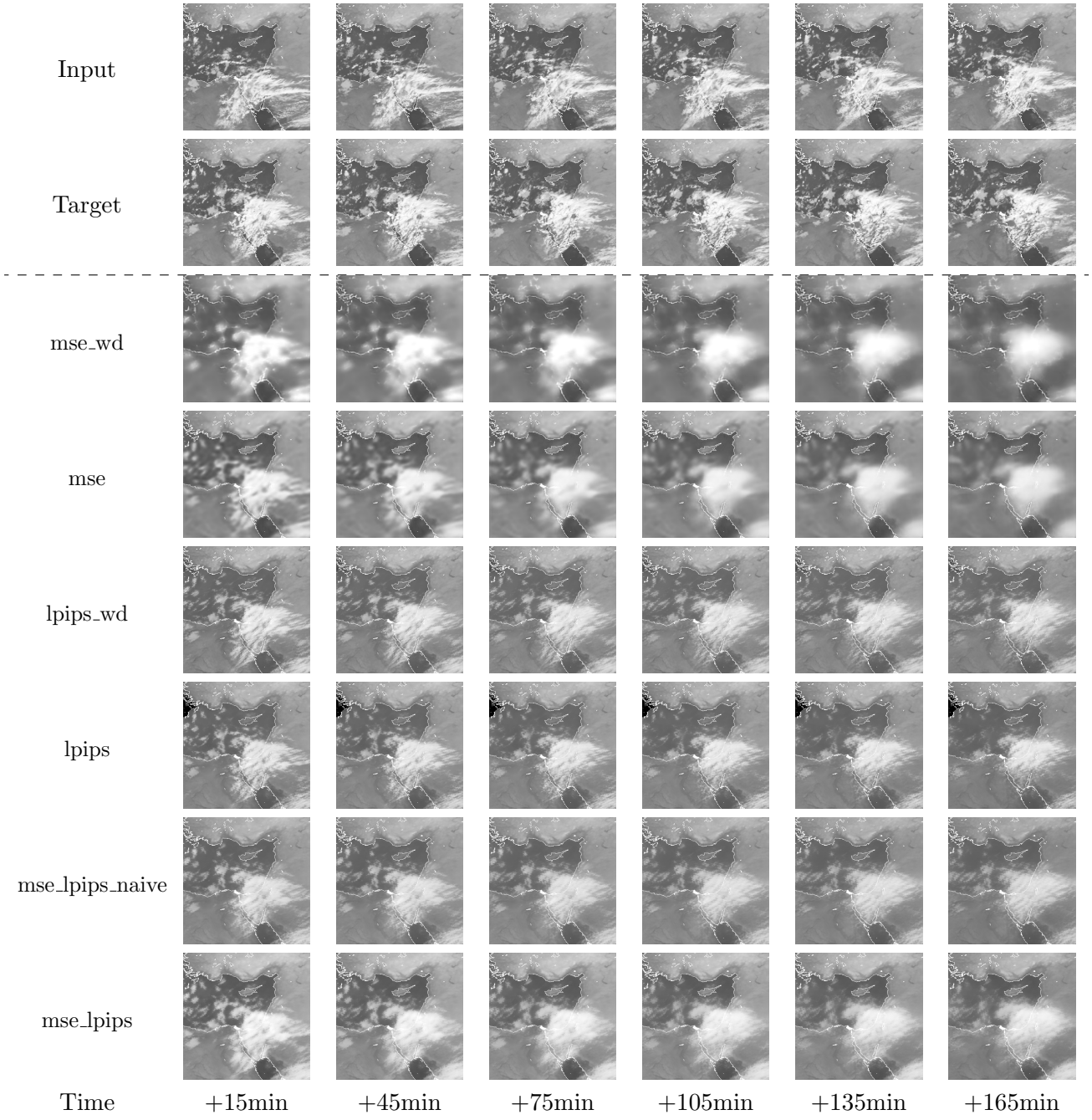

Table 10: Qualitative results on IR dataset of LPIPS.

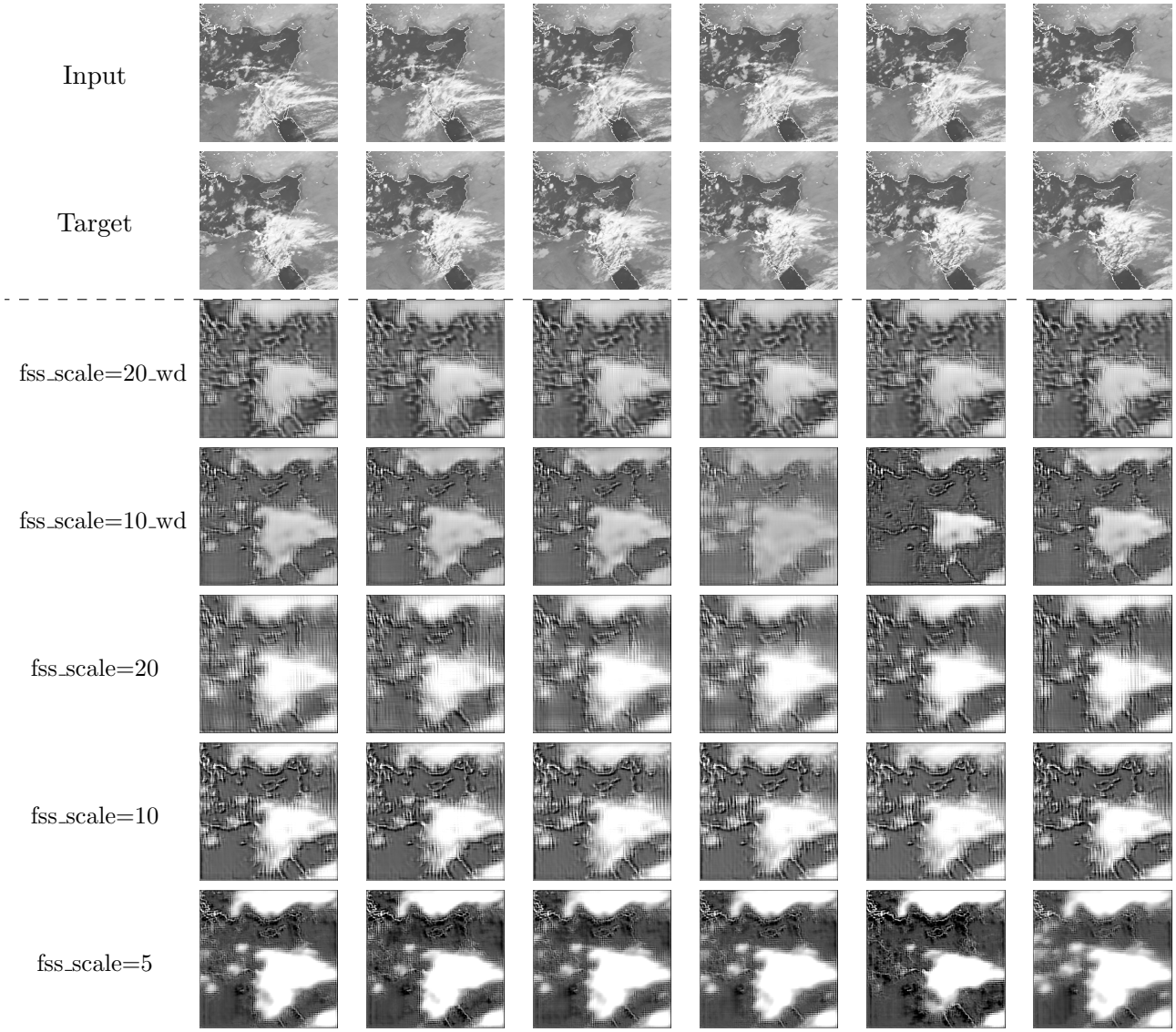

Table 11: Qualitative results on IR dataset of FSS.

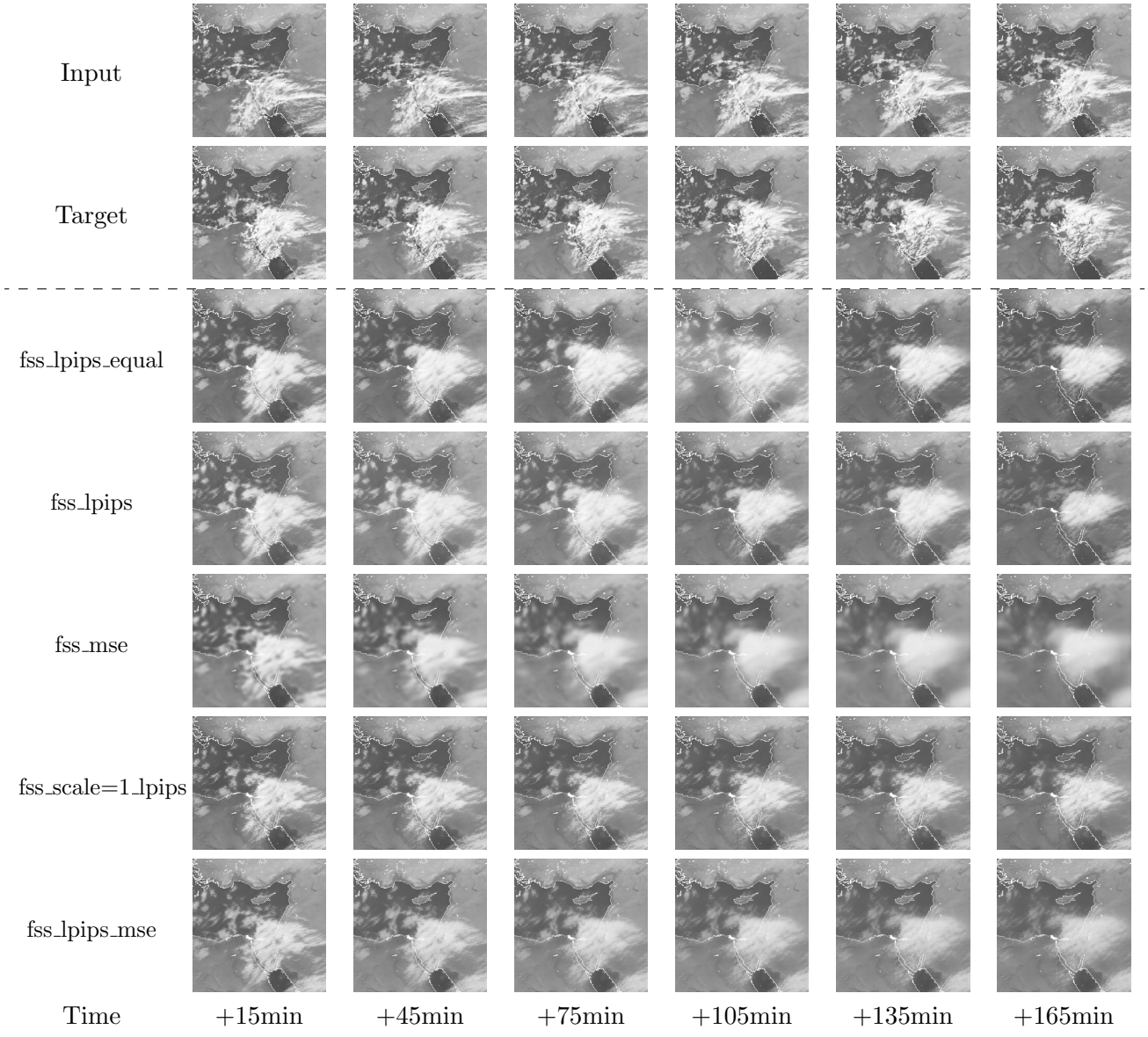

Table 12: Qualitative results on IR dataset of weighted combinations including FSS.

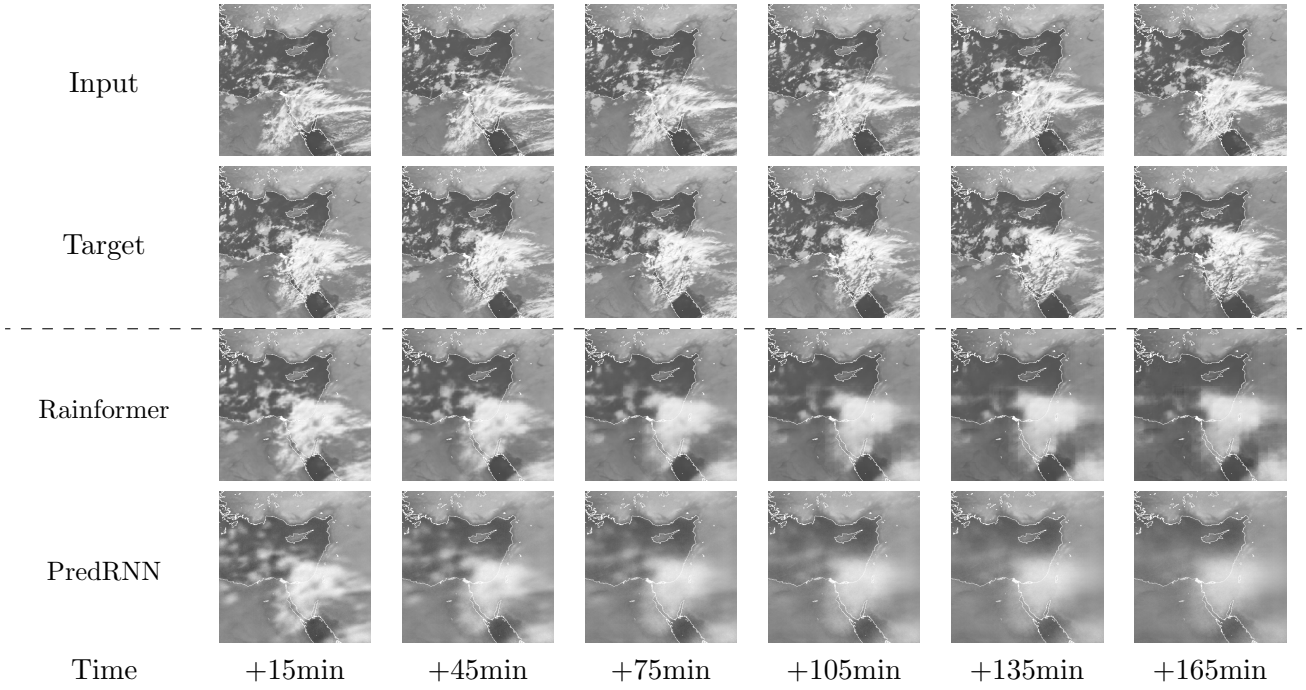

Table 13: Qualitative results on IR dataset of Baselines.

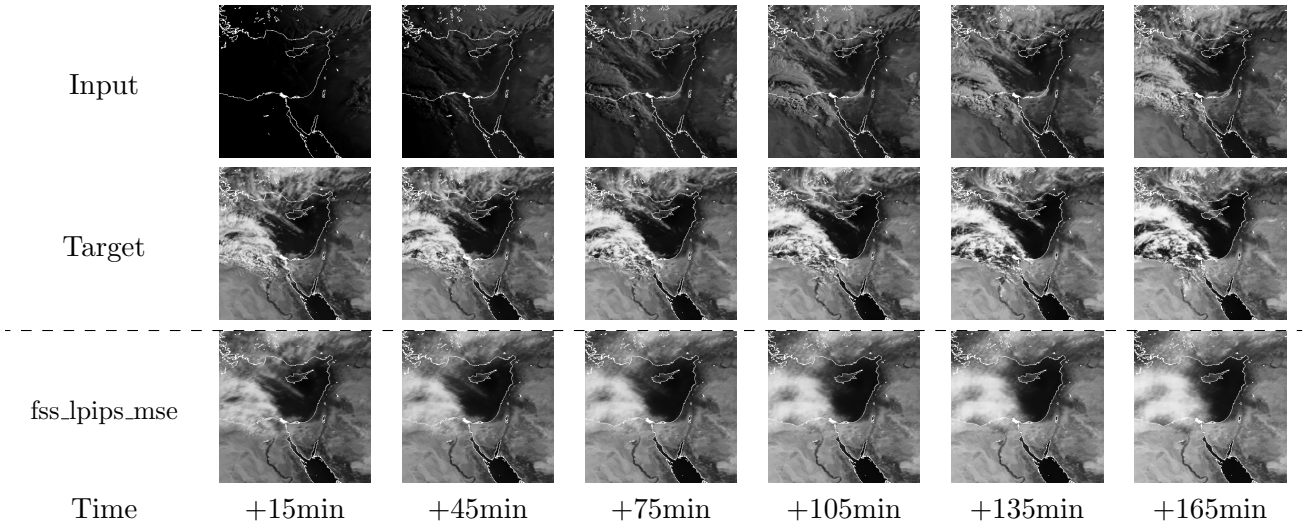

Table 14: Qualitative results on VIS dataset.

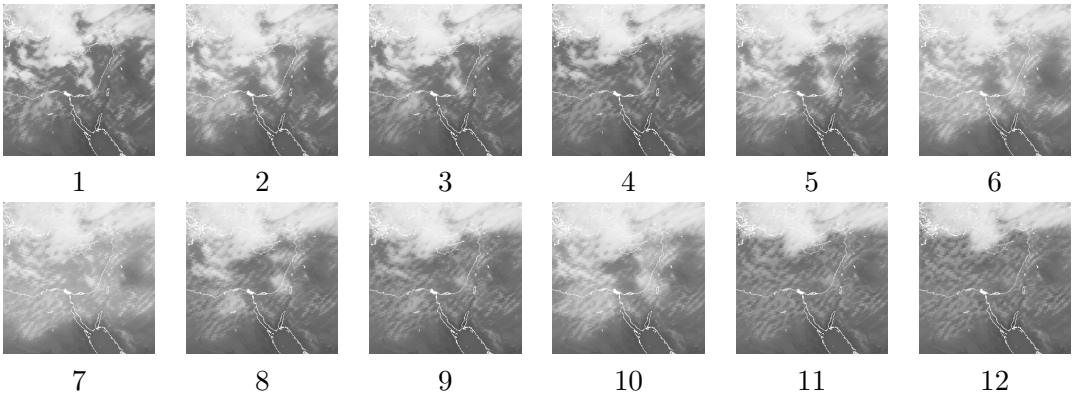

Table 15: Example of flickering when FSS loss is used with LPIPS. This example shows flickering in images 4-10. The model that is used for prediction is fss_lpips_equal.

C Limitations

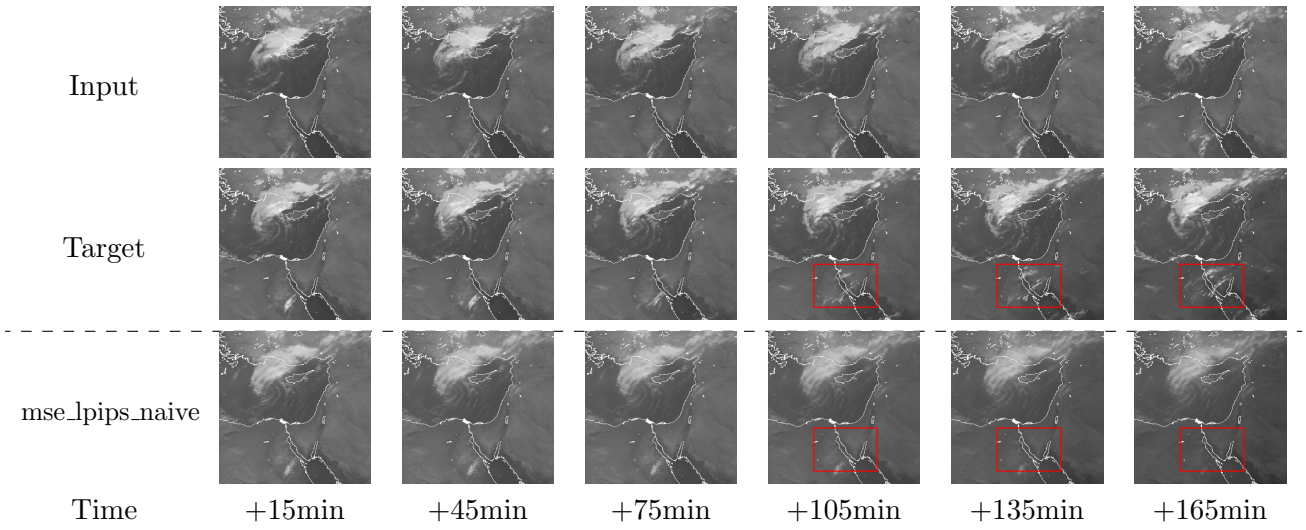

Table 16: Limitation - creation of clouds - example 1.

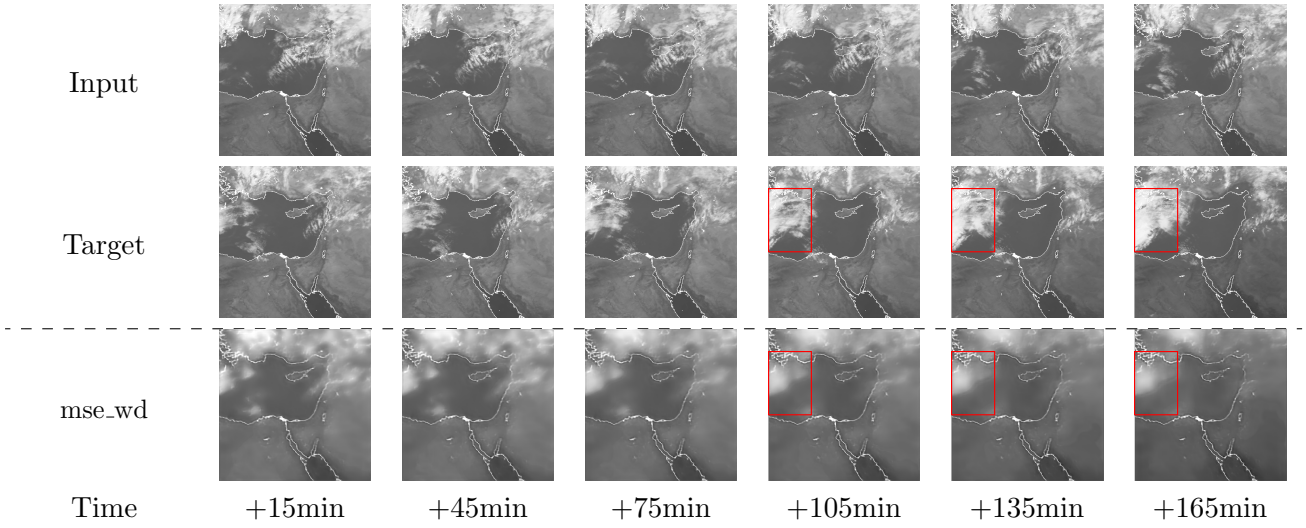

Table 17: Limitation - creation of clouds - example 2.

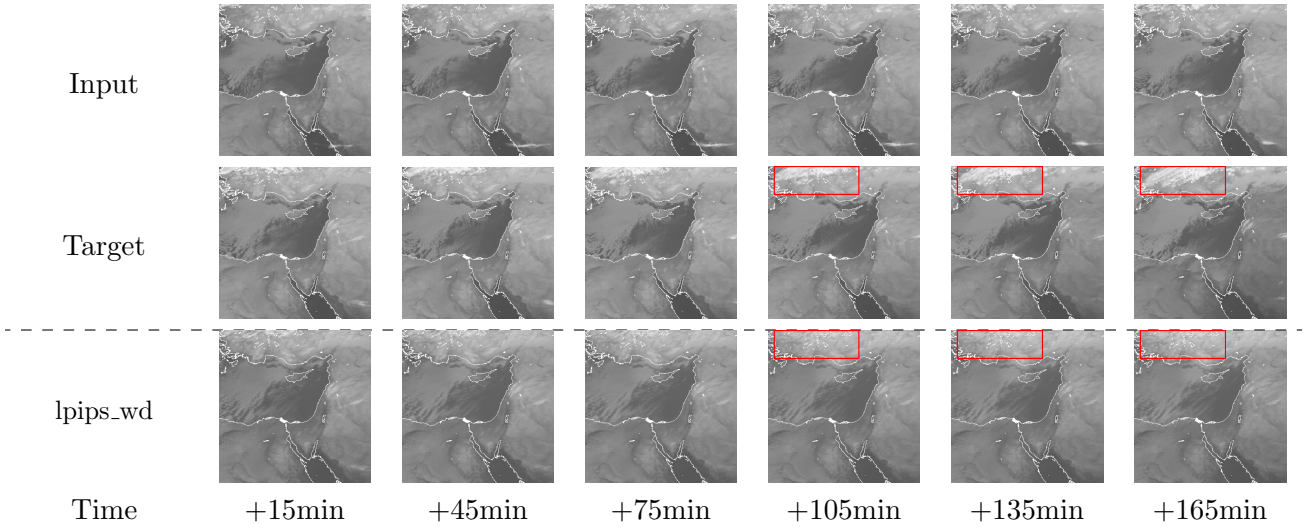

Table 18: Limitation - creation of clouds - example 3.

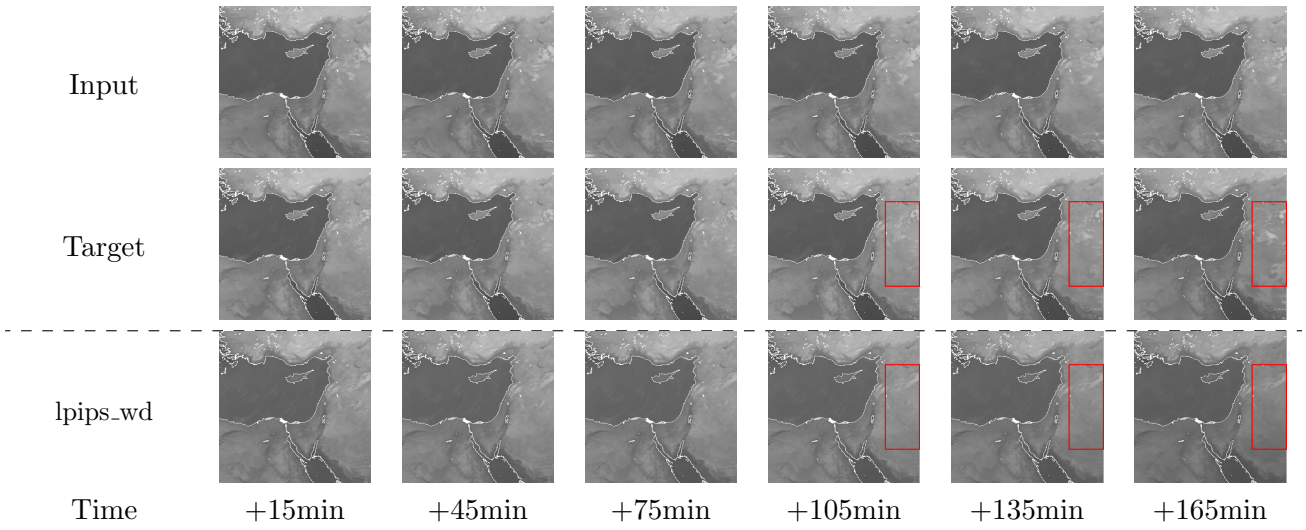

Table 19: Limitation - creation of clouds - example 4.

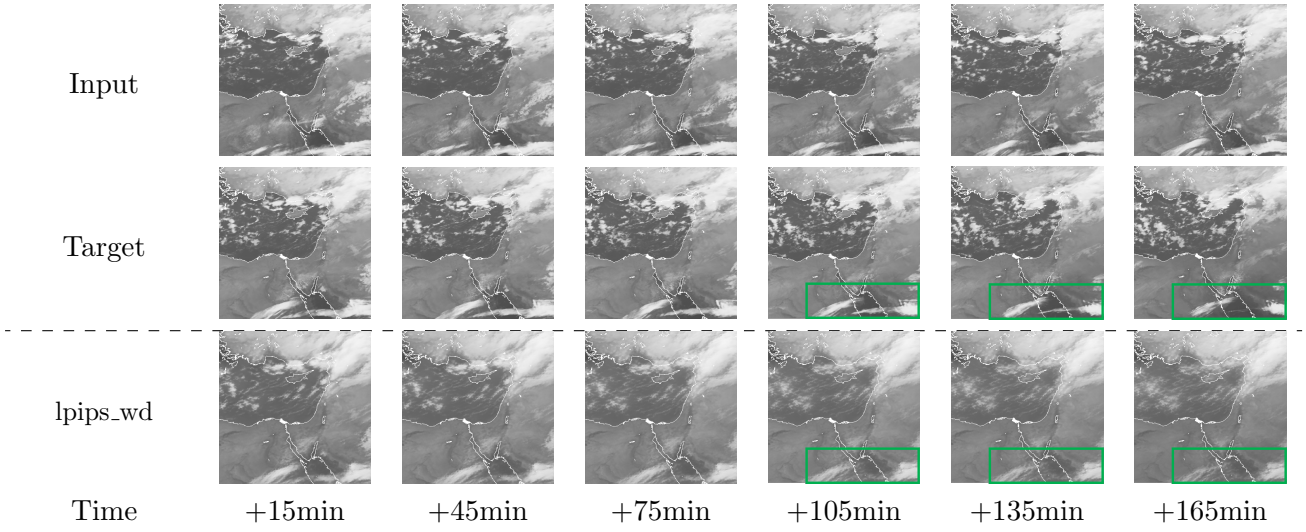

Table 20: Success - disappearance of clouds - example 1.

D Source Code

All source code of this project can be found in the following repository: https://github.com/leahandofir/cloud-forecasting-transformer, which is a fork of the Earthformer project: https://github.com/amazon-science/earth-forecasting-transformer.

References

[1] Léa Berthomier, Bruno Pradel, and Lior Perez. Cloud cover nowcasting with deep learning. *CoRR*, abs/2009.11577, 2020.

[2] Zhihan Gao, Xingjian Shi, Hao Wang, Yi Zhu, Yuyang (Bernie) Wang, Mu Li, and Dit-Yan Yeung. Earthformer: Exploring space-time transformers for earth system forecasting. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 25390–25403. Curran Associates, Inc., 2022.

[3] Xingjian SHI, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun WOO. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[4] Yuankang Ye, Feng Gao, Wei Cheng, Chang Liu, and Shaoqing Zhang. Msstnet: A multi-scale spatiotemporal prediction neural network for precipitation nowcasting. *Remote Sensing*, 15(1), 2023.

[5] Vincent Le Guen and Nicolas Thome. Disentangling physical dynamics from unknown factors for unsupervised video prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[6] Yunbo Wang, Haixu Wu, Jianjin Zhang, Zhifeng Gao, Jianmin Wang, Philip S. Yu, and Mingsheng Long. Predrnn: A recurrent neural network for spatiotemporal predictive learning, 2022.

[7] Cong Bai, Feng Sun, Jinglin Zhang, Yi Song, and Shengyong Chen. Rainformer: Features extraction balanced network for radar-based precipitation nowcasting. *IEEE Geoscience and Remote Sensing Letters*, 19:1–5, 2022.

[8] Cloud detection (ir vs. vis). https://www.theweatherprediction.com/habyhints2/512/#:~:text=Visible%20(VIS)%20satellite%20imagery%20and,radiation%20coming%20off%20of%20clouds. Accessed: 2023-08-05.

[9] Sevir dataset. https://sevir.mit.edu/sevir-dataset. Accessed: 2023-08-05.

[10] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution, 2016.

[11] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[12] Nigel M. Roberts and Humphrey W. Lean. Scale-selective verification of rainfall accumulations from high-resolution forecasts of convective events. *Monthly Weather Review*, 136(1):78 – 97, 2008.

[13] M. P. Mittermaier. A "meta" analysis of the fractions skill score: The limiting case and implications for aggregation. *Monthly Weather Review*, 149(10):3491 – 3504, 2021.

[14] Mark S. Veillette, Siddharth Samsi, and Christopher J. Mattioli. Sevir: A storm event imagery dataset for deep learning applications in radar and satellite meteorology. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[15] Sevir dataset repository. https://github.com/MIT-AI-Accelerator/sevir_challenges/blob/dev/radar_nowcasting/RadarNowcastBenchmarks.ipynb. Accessed: 2023-08-07.

[16] Rainformer github repository. https://github.com/Zjut-MultimediaPlus/Rainformer. Accessed: 2023-08-11.

[17] Predrnn github repository. https://github.com/thuml/predrnn-pytorch. Accessed: 2023-08-11.

[18] Colin White, Mahmoud Safari, Rhea Sukthanker, Binxin Ru, Thomas Elsken, Arber Zela, Debadeepta Dey, and Frank Hutter. Neural architecture search: Insights from 1000 papers, 2023.