6.869 Final Project Report
Evaluating Machine Learning Methods for Nowcasting

Aniruddha Deshpande
6.869
ani0203@mit.edu

Joonhee Kim
6.869
joonheek@mit.edu

Abstract

Nowcasting is a weather forecasting problem that aims to predict weather events, such as rainfall, in a given region over short time spans. Our project addresses this challenging task from a machine learning perspective. A good predictive model will have a wide-range of useful applications, including storm management and emergency response, renewable energy generation forecasting, and electricity grid management. The ability to foresee extreme weather events is increasingly important as we adapt to a more varying climate. We select the SEVIR dataset from the MIT Lincoln Laboratory, which consists of spatially and temporally referenced radar images collected from multiple sensors across the contiguous United States. Previous studies have employed two different model architectures - U-Net and ConvLSTM - for nowcasting; however, there has not been a comparative evaluation of these two models to suggest which may be more well suited for this task. We evaluate the two architectures to compare their performances for nowcasting. We also explore the effect of varying the contribution of two types of losses - mean square error and style-content loss in the loss function - on the prediction images. In our experiments we find that the U-net model overall performs better than the ConvLSTM, but the preferred model would depend on the relative importance of avoiding false positives vs false negatives. We do not find any clear monotonic effect of varying the contribution of the different losses in the model training.

1. Introduction

Our project evaluates machine learning methods for an application in meteorology called nowcasting. Nowcasting is defined as "forecasting weather with local detail, by any method, over a period from the present to six hours ahead" [1]. Unlike traditional weather forecasting models that primarily rely on physics to model complex weather systems, nowcasting focuses on creating rapidly updated, high-resolution observations of high-impact weather events due to the short prediction timeframe. Nowcasting has the potential for significant impact and value on many real-world applications. Examples include storm management, electricity load management, and aviation, where providing alerts for emergency response is crucial.

As a machine learning objective, the nowcasting challenge is formulated as a prediction task. An input sequence of radar image frames are used to generate the next sequence of future frames in order to detect regions that may experience extreme weather in the specified window.

Deep learning methods to address the nowcasting problem has been proposed through two different network architectures; the U-Net [6, 7] and ConvLSTM [4].

In [7], a U-Net architecture was used to produce 12 future images based on an input of 13 images. In [6], a U-Net architecture that is referred to as SmaAT-UNet was implemented employing attention modules in the time domain. This improved interpretability in terms of which time instance matters more or less for prediction as well as what their relative importance is. In addition, the SmaAt-UNet model used depth-wise separable convolutions. This helped lower the number of parameters, making the overall model smaller and less memory-intensive. However, the authors of [6] use their model to predict only one future frame.

Long short-term memory networks, or LSTMs, have been frequently applied to address problems with temporal data. The fully-connected LSTM (FC-LSTM) was introduced in 2015 [5] with an encoder-forecaster network for multi-step predictions. Since the FC-LSTM does not capture spatial dependencies, Shi et al. 2015 [4] first proposed the ConvLSTM network for spatio-temporal sequence problems by implementing convolutional structures in both the input-to-state and state-to-state transitions of the FC-LSTM. Wu 2019 [9] built on top of [4] by incorporating an encoder-forecaster structure to improve the future prediction task. Wu 2019 used this model on the training MNIST data, which contains 14 frames of 64x64 images.

While the U-Net architecture is well suited for image segmentation and localization, it doesn not explicitly cap-

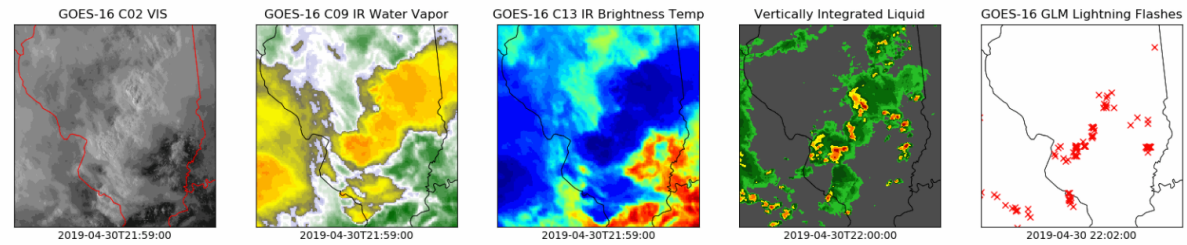


Figure 1. Sample from dataset [7]

ture temporal dependencies in sequential data like the ConvLSTM model. However, the recurrent component of the ConvLSTM may reduce the resolution of the images. In addition, the advantage of the LSTM for images sequences with little movement is not obvious.

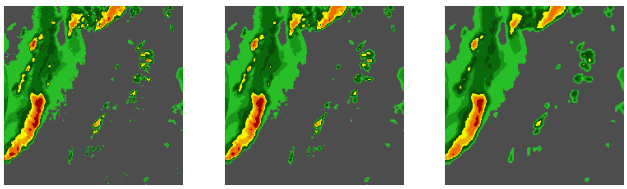
In this project, we implement and evaluate both a U-Net and ConvLSTM model in order to compare performances for nowcasting to determine which model may be better suited for the task. We follow the architecture of the SmaAT-UNet model [6], but expand its objective to the same prediction task in [7] - to predict 12 frames given an input of 13 frames. For our ConvLSTM model, we follow the structure provided by [9], but expand the resolution of the input images from 64x64 to higher resolutions of our dataset.

In addition, we evaluate different variations of loss functions to determine what combination of mean square error (MSE) loss and style content (SC) loss yields the best image outputs for nowcasting.

2. Data

This project uses the SEVIR dataset provided by the MIT Lincoln Laboratory. This dataset contains over 10,000 annotated and spatio-temporally aligned image sequences spanning 4 hours of time in 5 minute time steps. The image sequences depict weather events over the contiguous United States. Figure 1 shows the 5 different modalities: visible satellite imagery (vis), infrared satellite imagery (ir069), infrared satellite imagery (ir07), NEXRAD radar mosaic of vertically integrated liquid (vil), and intercloud and cloud to ground lightning events (lght). Following [7], we train our models (described in the following section) on the vil image type, which are of size 384 km x 384 km. We also follow the dataset authors by standardizing the data using the sample mean and variance prior to training or inputting the data into our models.

Due to the lack of computational resources we had available on Google Colab to train our models, we were unable to use the full resolution of the data. Instead, we down-sampled the images to lower resolutions (192x192 and 96x96 images). To do this, we implemented a blur and down-sample operation to the specified resolution. Figure 2 shows a visual comparison of a sample input image at the original 384x384 resolution and the two down-sampled images. We see that as the images get further down-sampled to lower resolutions, the original texture of the image gets lost.



(a) 384x384 (b) 192x192 (c) 96x96

Figure 2. Comparison of sample image at different resolutions used in this project

3. Methodology

3.1. U-Net

The SmaAT-UNet (small attention U-Net) architecture from [6] is shown in Figure 3. We adapt this network to predict 12 output frames at once as opposed to one frame as seen in the figure. This architecture differs from the one used in [7] in its uses of Depth-wise separable convolutions (DSC) and Convolution-block Attention modules (CBAM) [8].

DSCs are useful in that they reduce the number of total parameters used in the model by separating the 3D convolution into 2 steps; depth-wise convolution followed by point-wise convolution. This makes storing the model a lot less memory intensive. This would be specifically helpful in the case where weather prediction applications are to work on smaller devices such as mobiles or computers with lower computing power and memory.

CBAM serves the purpose of learning attention weights in the spatial as well as time-domain, and trying to ensure that the model focuses on more important features.

The overall architecture [6] is standard, with an encoder and decoder part. The encoder successively down-samples the images by a factor of two and doubles the number of feature maps.The decoder uses bilinear upsampling to bring

the feature maps back to the dimensions of the original input image.

The model also uses skip connections at each step so that the final inference task is performed using features at different scales. Also note that the attention module is used only through the skip connections and not while down-sampling or up-sampling the feature maps.

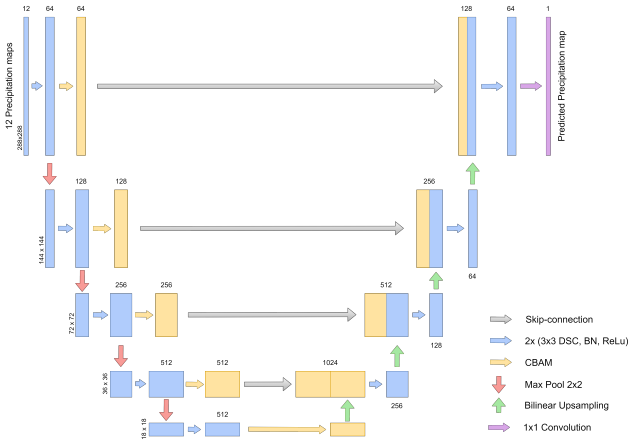


Figure 3. Small Attention U-net as described in [6]

3.2. ConvLSTM

For the ConvLSTM model, we implement the structure from Wu 2019 [9]. This architecture has convolutional structures in the input-to-state and state-to-state transitions as shown in Figure 4. For the sequence-to-sequence prediction task, we use the encoding and forecasting network like the FC-LSTM. Our model essentially consists of two ConvLSTMs: the encoder ConvLSTM reads the input sequence of frames, and the forecaster ConvLSTM takes the accumulated hidden states and memory cells to predict the future sequence. As an illustrative example, Figure 5 shows how the encoder and forecaster models interact for a 2-layer FC-LSTM. Figure 6 shows the architecture of a 2-layer ConvLSTM, the LSTM equivalent of two memory cells.

One advantage of this structure is that the model is not strictly limited in reading exactly 13 input frames and producing 12 output frames as in the U-Net. However, for comparison, we limit our evaluation to this combination of frames.

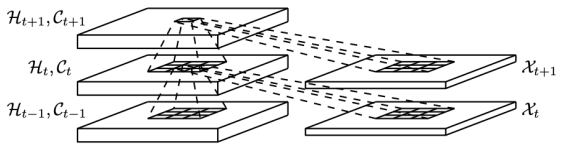


Figure 4. Inner structure of ConvLSTM [4]

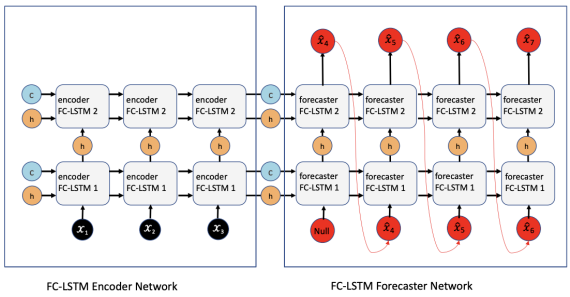


Figure 5. Example of a FC-LSTM encoding-forecasting network [9]

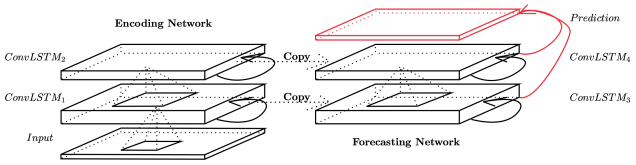


Figure 6. Example of 2-layer stacked ConvLSTM encoder-forecaster network [4]

Because ConvLSTM is not an available module in PyTorch, we implement this model using TensorFlow using the skeleton from the above mentioned references. Our encoder and forecaster networks are stacked with 3 ConvLSTM layers. The input-to-state and state-to-state kernel sizes are 5x5, the number of hidden states in the hidden layer is 64.

3.3. Loss functions

To train the models we use a combination of multiple loss functions. The simplest we use is the L_{MSE} which is simply the mean square error loss between the predicted outputs and the target outputs. However, MSE loss can lead to poor texture of predicted images [7], as well as an over-reliance on pixel-wise correspondence rather than capturing the target image's overall content. Therefore, we also employ style and content losses (L_{style} and $L_{content}$) computed from a VGG16 model trained on the ImageNet dataset as described in [2] and [3]. We use scaling factors decided by observation so that the scales of the two losses are comparable. A combination of these losses (L_{SC}) is used by the the authors of [7] to train their U-Net, and found that adding style and content losses along with MSE drastically improves quality of the images both qualitatively via visual outputs and quantitatively via performance metrics.

Our combined loss function that we train our models on is defined as:

$$L_{combined} = \alpha L_{MSE} + (1 - \alpha) L_{SC},$$

where our parameter $\alpha \in [0, 1]$ is used to decide the relative weighting between these 2 losses. We can see that for

$\alpha = 1$, the loss function is L_{MSE} and $\alpha = 0$ uses only L_{SC}.

4. Evaluation

4.1. Metrics

For nowcasting, we use common weather forecasting evaluation metrics as described in [7]. Pixels for the ground truth and predicted sequences are classified to "Hits (true positives)" "Misses (false negatives)" and "False Alarms (false positives)" for various threshold values, where pixels greater than the threshold are classified to have an event (such as precipitation) occur. The following metrics - probability of detection (POD, also known as recall), Success Ratio (SUCR, also known as precision), and critical success index (CSI, also known as intersection over union) - are defined as the following ratios:

$$\text{POD} = \frac{\#Hits}{\#Hits + \#Misses}$$

$$\text{SUCR} = \frac{\#Hits}{\#Hits + \#FalseAlarms}$$

$$\text{CSI} = \frac{\#Hits}{\#Hits + \#Misses + \#FalseAlarms}$$

Apart from the overall ratio over an entire prediction sequence, another dimension of evaluating model performance is tracking these metrics over time, or from frame to frame, as the decay in performance may differ across the two models as well as α in our loss function.

4.2. Comparing Model Architectures

The first question we investigate is the relative performance of the U-net and ConvLSTM architectures for nowcasting. The key difference between the two architectures is the following; given the 13 input frames, the U-net predicts the next 12 frames all at once. The ConvLSTM on the other hand does the predictions sequentially, ie. predicts one frame at a a time, and then uses the prediction at time t as the input to predict the frame at time t+1.

We found that training the ConvLSTM model with 192x192 images was not possible as it led to the GPU RAM that was available to us running out. Hence, we compared the performances of the 2 architectures on further downsampled 96x96 images. We used a training set of 800 image sequences, a validation set of 200 images and a test set of 500 images. Both models were trained using the loss function described in 3.3, with $\alpha = 0.5$, using the Adam optimizer with a learning rate of 0.001 and batch size of 4.

We compared the performance of the models in three ways; first by visually inspecting the outputs of the models, second by comparing the values of metrics described in 4.1 over all the predicted frames, and third by seeing how these metrics vary with the lead time, ie. how far in the future you are making the prediction.

4.3. Comparing Effect of α in Loss Function

The second research question we investigate is the relationship between performance and the relative weighting of L_{MSE} and L_{SC} in the loss function. Because we found our U-Net easier and faster to train, we conducted the analysis of varying the α term in our loss function on this model. Here, we were able to use the same number of samples in the training, validation, and test sets but at a higher resolution of 192x192 images.

We trained 5 separate U-Net models on 5 distinct loss functions, where α was varied in the set $\{0, 0.25, 0.5, 0.75, 1\}$. To reiterate, the SC loss is favored for a lower α, and the MSE loss for a higher α. These models were trained with the same parameters (Adam optimizer, learning rate of 0.001, and batch size of 4) as well as evaluated and compared by the same procedure as described above.

5. Results

5.1. Comparing Model Architectures

Metric	Threshold	Conv-LSTM	U-Net
POD	16	0.60	**0.78**
	74	**0.55**	0.46
	133	**0.39**	0.08
SUCR	16	0.83	**0.89**
	74	0.72	**0.91**
	133	0.32	**0.80**
CSI	16	0.53	**0.71**
	74	**0.46**	0.44
	133	**0.21**	0.08

Table 1. Evaluation metrics of ConvLSTM and U-Net model outputs over test set across different thresholds

Figure 7 shows a comparison of the outputs from the two models for a sample event at 15, 30, 45, and 60 minute lead times. Visually, we see that the ConvLSTM outputs show degradation in the shape, especially at longer lead times (frames most far away from the initial input sequence). The U-Net outputs appear to retain the boundaries of the target shape better, but it lacks the high intensity pixel values across all lead times. It is also interesting that even at the same α or weighting between the MSE loss and SC loss, the U-Net outputs appear to retain the original texture better than the ConvLSTM outputs, which appear smoother along the edges.

Figure 8 plots the evaluation metrics described in Section 4.1 on the entire test set over lead time for the ConvLSTM and U-Net model outputs, and Table 1 compares the

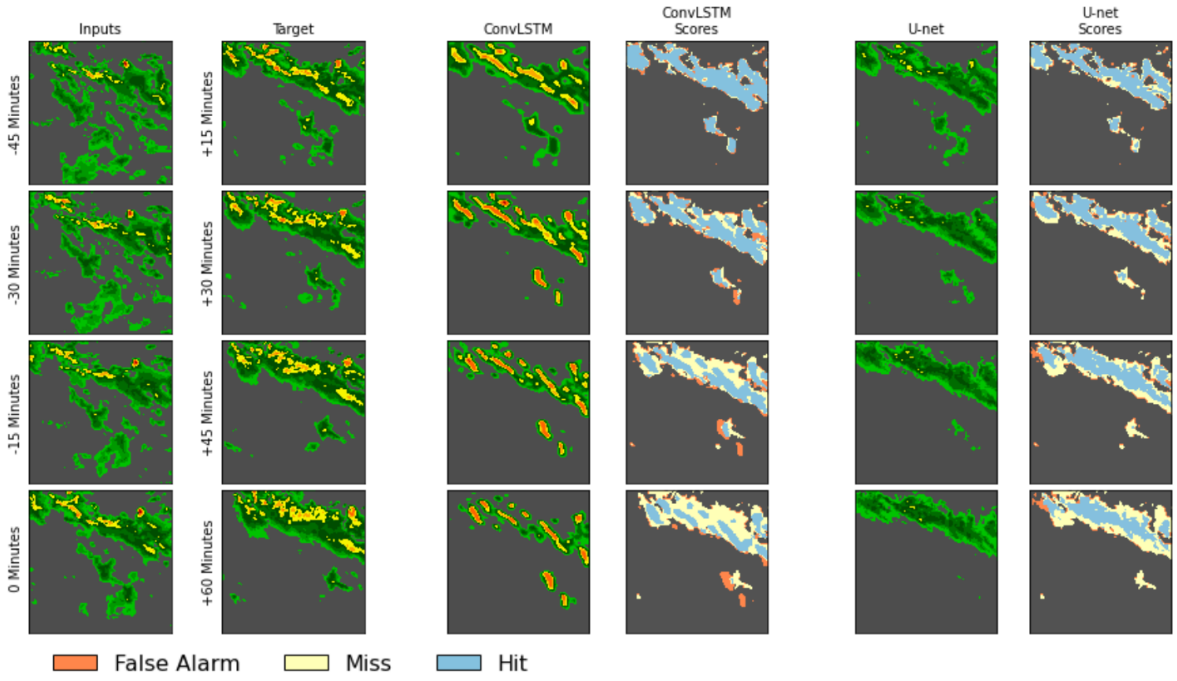


Figure 7. Outputs from ConvLSTM and U-Net models for $\alpha = 0.5$ compared to the target output

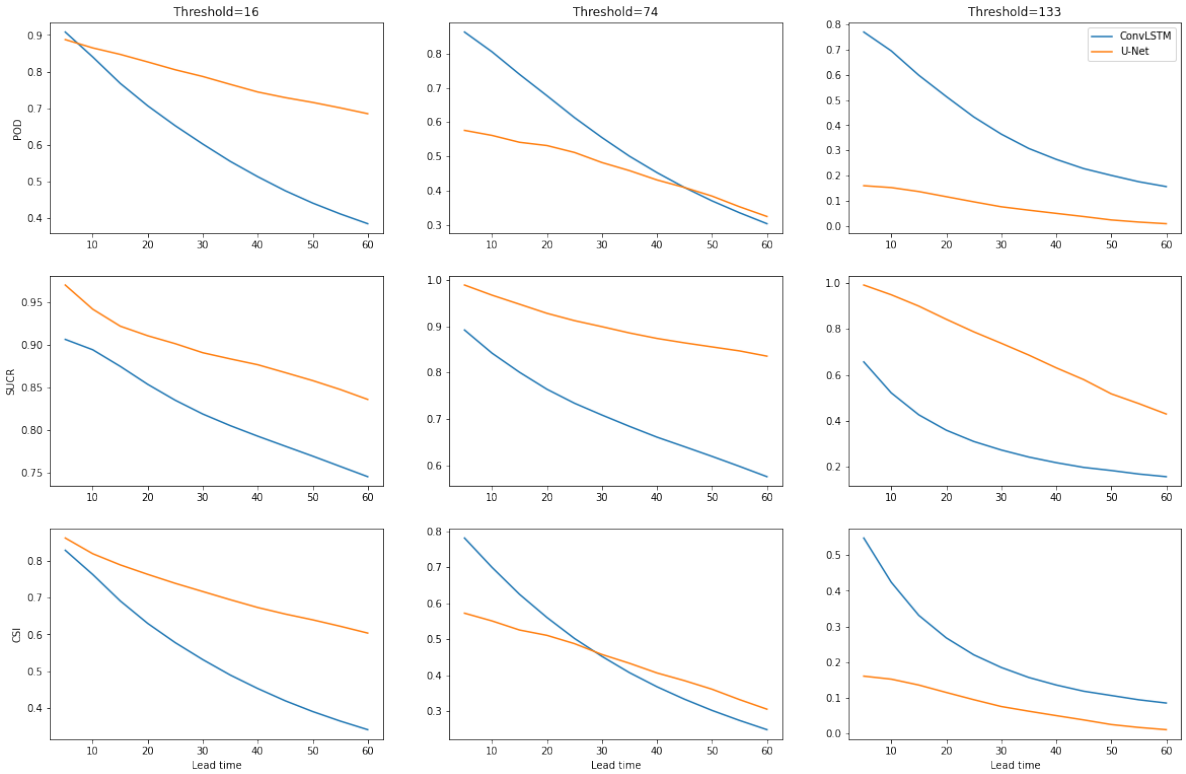


Figure 8. Evaluation metrics of ConvLSTM and U-Net model outputs for $\alpha = 0.5$ over lead time

overall metrics across the entire sequence. Bolded values in each row indicate better performance for that metric under a specified pixel threshold. Both Figure 8 and Table 1 verify the visual findings in Figure 7 that the U-Net outputs appear more consistent in shape but lack higher pixel intensities.

In Table 1, the SUCR metric shows that the U-Net outperforms the ConvLSTM for all thresholds, suggesting that ConvLSTM has a high false alarm rate. The SUCR metric under the highest threshold (133) shows the biggest difference between the two models. This can also be see in Figure 7, where the ConvLSTM outputs contain a lot of extra "red" pixels where they don't exist in the corresponding target frames. Both models perform similarly (to themselves) across the POD and CSI metrics, especially the U-Net. Since the only difference between the two metrics is that the denominator in CSI adds the "False Alarm," this suggest that the false alarm rate in the U-Net is very low. Still, the lowest scores occur for the U-Net under the POD and CSI at the highest threshold, meaning that the U-Net misses most high-intensity events.

In Figure 8, we see that for any given metric and threshold, one model tends to consistently perform better than the other across all lead times. This is really only an exception for CSI at threshold=74 (changes in model performance POD at threshold 64 and 74 are very minor, and occur at the very beginning or tail end). We find that across all thresholds, the decay or rate of change in any metric is noticeably larger for the ConvLSTM. This is also consistent with our observation in Figure 7 that the ConvLSTM output frames visibly degrade in quality much faster than the U-Net outputs as lead time increases.

5.2. Comparing Effect of α in Loss Function

Metric	Threshold	α				
		0	0.25	0.5	0.75	1
POD	16	0.79	0.81	0.80	0.79	**0.91**
	74	**0.66**	0.57	0.64	0.51	0.60
	133	**0.21**	0.14	0.18	0.05	0.10
SUCR	16	0.89	0.89	0.89	**0.90**	0.78
	74	0.84	0.90	0.88	0.93	0.90
	133	0.57	0.85	0.82	**0.95**	0.89
CSI	16	0.72	**0.74**	0.73	0.73	0.72
	74	**0.59**	0.54	0.59	0.49	0.57
	133	**0.18**	0.14	0.17	0.05	0.10

Table 2. Evaluation metrics of U-Net model outputs over test set across different α values in loss function $L_{combined}$

Figure 9 shows the frames predicted by the U-net model trained on different α values.We observed that for lower α values the predicted frames seem to pick up on the higher intensity pixels better than models trained on higher α values. This checks out as the colour variations seem intuitively closer to the idea of the image texture, and the style-content loss L_{SC} specifically controls for replicating the target image texture. It can also be seen that the outputs for higher α values appear more 'blob-like' and smooth which also serves to corroborate the above point.

We also investigate the how α affects the objective evaluation metrics of POD, SUCR and CSI in table 2 and figure 10. We do not observe any predictable monotonic effect of α on these metrics. For the smallest threshold value of 16 most of the models seem to perform almost identically, with the exception of $\alpha = 1$ which gives a higher probability of detection, but also a much lower success rate. As we increase the threshold values, the performances of the different models start diverging. The $\alpha = 0$ (only L_{SC}) model performs better for POD, but successively worse for SUCR. It's performance also does not monotonically decay with lead times as in the case of all the other models, which is a little strange. This could be an indicator that while style-content loss is useful, without some component of mean-squared loss along with it, it can give unpredicatable results. Other than this, we find that the $\alpha = 0.5$ model does pretty well in terms of POD and not as badly for the $SUCR$ metric as the only MSE model. $\alpha = 0.75$ gives the best results in terms of the SUCR metric. The behaviour of CSI seems to closely follow the POD trend. Observing the tables and figures it isn't clear which model does the best, but it will give us the ability to choose a model based on what our aim is; to maximise detection probability or success rate.

6. Discussion

One clear trend that we observed in both comparing the U-net vs ConvLSTM models and the models trained using different α values was the trade-off between POD and SUCR. This translates to a trade-off between training a model that is capable of making more detections, but at the cost of more false alarms as well. We will choose the model depending on the our specific goal and how okay we are with false alarms or misses respectively.

This trade-off appears to be linked also with the model's ability to predict the shape of the target image and the pixel intensities. Models which did one well, seemed to underperform for the other.

7. Limitations to Current Approach

The biggest limitation to our work was the lack of GPU resources. This led to us making certain choices through the course of our project.

- Down-sampling the data from the full 384x384 image resolution to lower resolutions.

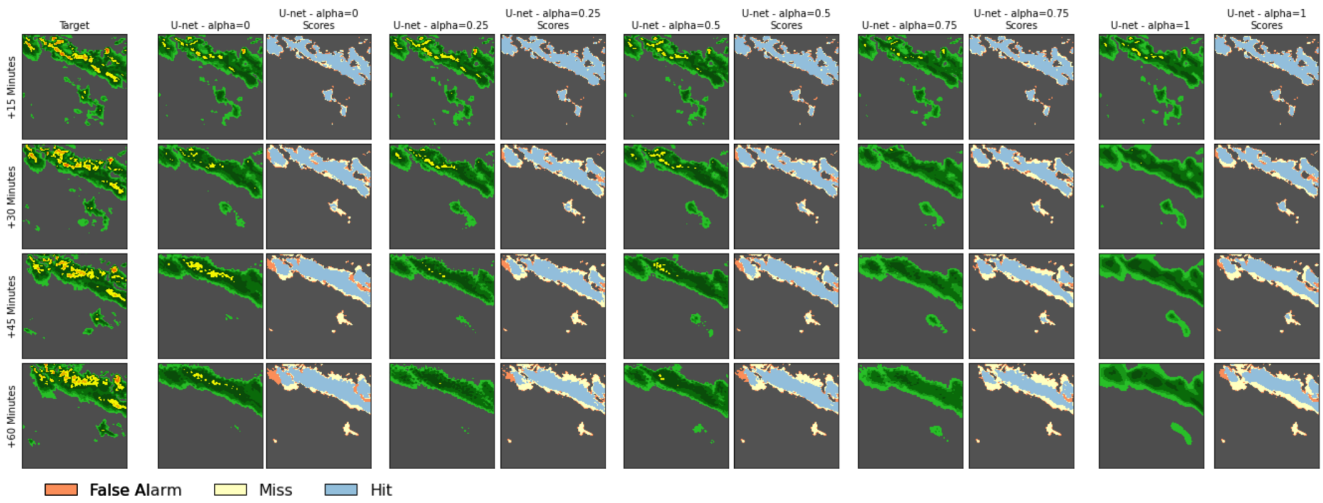


Figure 9. Outputs from U-Net models for $\alpha = 0, 0.25, 0.5, 0.75, 1$ compared to the target output

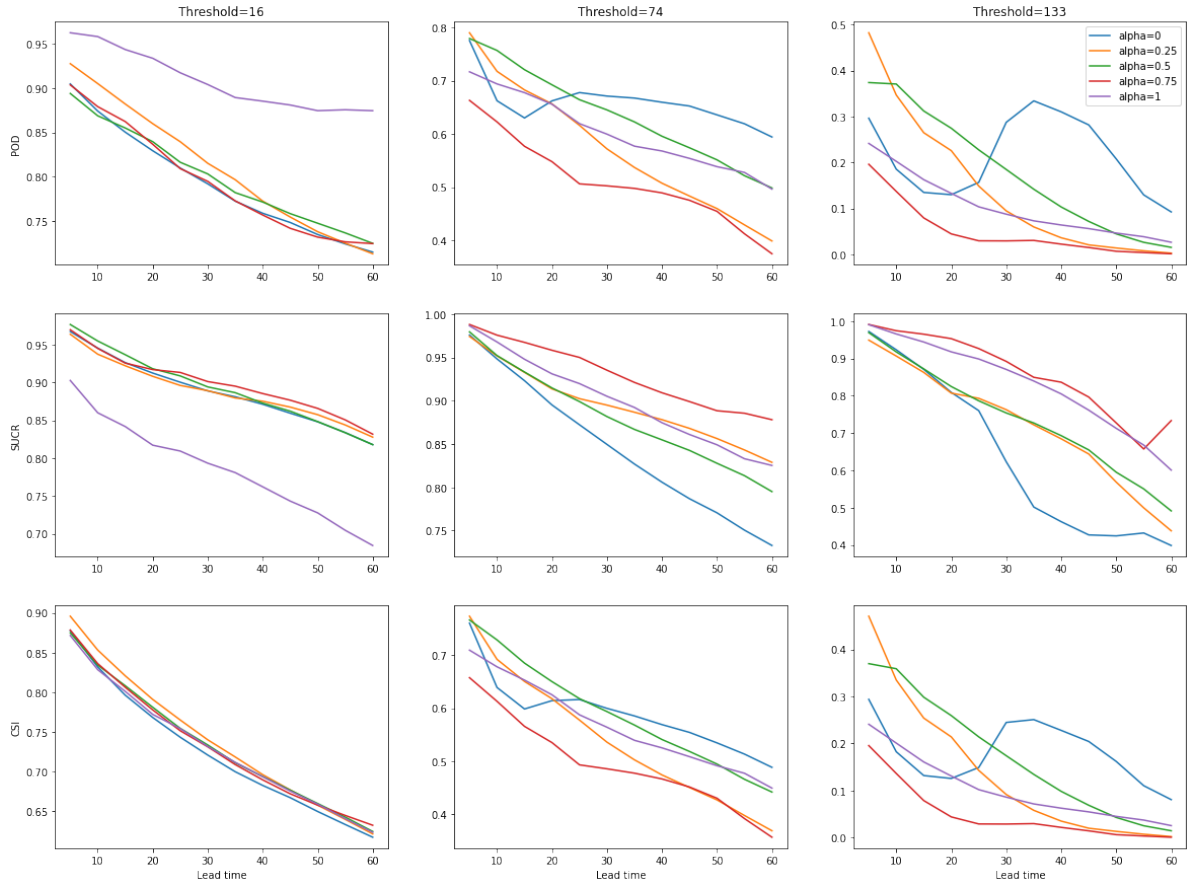


Figure 10. Evaluation metrics of U-Net model outputs for $\alpha = 0, 0.25, 0.5, 0.75, 1$ over lead time

- Using only a small subset of the available data (1000 sequences for training and validation sets) to train the models
- Performing α experiment only on U-Net model

As previously mentioned, the ConvLSTM was our limiting model as we had to use 96x96 images to train and com-

pare the two models. You can visually observe in Figure 2 the differences between this and the original resolution after performing the down-sampling and blur procedure. We believe that had the two models been able to train on the full dataset containing 10,000 sequences, at the full resolution, both models' metrics would have increased. Since the resulting predictions had distinct limitations - ConvLSTM had high false alarm rates, U-Net had high miss rates for high thresholds - we cannot say which model would benefit the most from more computation resources.

Further, we have made some observations about the effect of α (combination of MSE loss and SC loss) on the prediction performance. However, the ConvLSTM may not have had the same pattern of results as we saw on our U-Net model based on varying α between 0 and 1. This argument could be supported by the fact that for our first experiment with $\alpha = 0.5$, the texture of ConvLSTM's prediction images looked smoother (more "MSE-like") than the U-Net's prediction (Figure 7).

8. Summary and Conclusions

Our work implemented two different machine learning architectures - U-Net and ConvLSTM - which so far has not been comparatively evaluated for the nowcasting prediction task. The ConvLSTM model predicts output frames one a ta time, whereas the U-net predicts all the 12 future frames in one go.

Under the same training setting (i.e., dataset, number of epochs, batch size, etc.) and loss function weighing MSE and SC loss equally, we find that the ConvLSTM predictions deteriorate in texture and shape as step time increases. On the other hand, the U-Net predictions remain more consistent in these characteristics over time but lack information about events with high intensity pixel values. Comparing these two models, there appears to be a trade-off between misses and false alarms, or in other words between the expected "shape" of the image and the "intensity" of pixel values.

For a given model architecture (U-Net), we varied the relative weighting between MSE loss and SC loss for the training loss functions as well. We don;'t find a clear effect on performance by varying α, but we do find that the performance in the case when both losses are used is better than when each loss is used individually. We find similar trade-offs between POD and SUCR performances in the different models as well as the trade-off between predicting the shape of the output vs the high intensity pixel values. We also find that the U-net model with reduced number of parameters is able to perform pretty well given its trained on only the partial dataset, and further training it with the complete dataset and more GPU resources could lead to a performance comparable to that in [7]. Similar performance with a less memory-intensive model would be a very helpful contribution.

9. Team Member Contributions - Aniruddha Deshpande

Our work was roughly divided into the following sections. Names in brackets indicate that team member took the lead on that item.

1. Data downsampling - Implemented the blur and downsample operation that we applied on all our data that we used for training, validation and testing
2. Assessing literature for model networks and nowcasting metrics. Studied U-net architectures implemented in [7] and [6]
3. Modified the U-net architecture implemented in [6] in pytorch. Implemented training, validation routines for this model. Also set up dataloaders to effectively use the limited RAM and GPU capabilities available to us. This was what allowed us to train our U-net model on higher resolution images and a larger training set at pretty good speed
4. Understood and incorporated implementation of style-content loss and modified it so that it can be used for training the U-net model. This differed from what we did in the pset and was closer to the implementation in [3].
5. Evaluating model results - Set-up notebook to evaluate performance of our trained models. This included visualising output images, evaluating and plotting the metric from section 4.1. This was also challenging at times as the session would often crash due to RAM running out. Had to ensure that code for evaluation used memory efficiently.
6. Presentation and final report

References

[1] Schmid Franziska, Yong Wang, and Abdoulaye Harou. Definition of nowcasting. In *Nowcasting Guidelines – A Summary*. World Meteorological Organization, 2019. 1

[2] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016. 3

[3] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016. 3, 8

[4] 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, 2015. 1, 3

[5] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms, 2015. 1

[6] Kevin Trebing and Siamak Mehrkanoon. Smaat-unet: Precipitation nowcasting using a small attention-unet architecture. *CoRR*, abs/2007.04417, 2020. 1, 2, 3, 8

[7] Mark Veillette, Siddharth Samsi, and Chris Mattioli. Sevir : A storm event imagery dataset for deep learning applications in radar and satellite meteorology. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 22009–22019. Curran Associates, Inc., 2020. 1, 2, 3, 4, 8

[8] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 2

[9] Mingkuan Wu. Sequential images prediction using convolutional lstm with application in precipitation nowcasting. *University of Calgary, Calgary, AB.*, 2019. 1, 2, 3