Incorporating Prior Knowledge To Forecast Fine-Grained Cloud-Top Temperature

Min Namgung (Advisor: prof. Yao-Yi Chiang)
Department of Computer Science and Engineering, University of Minnesota, Twin Cities, MN
namgu007@umn.edu

Introduction

With a large number of meteorological satellite images, forecasting weather has become important but challenging at a fine spatial resolution by jointly modeling spatiotemporal dependencies in a large-scale data-driven method. In this poster, we propose a machine learning approach incorporating prior knowledge (e.g., cloud type) to forecast cloud-top temperatures at a fine spatial scale in the next hour. To forecast cloud-top temperature precisely, we integrate prior knowledge of each cloud type that domain experts have provided.

Feature Selection

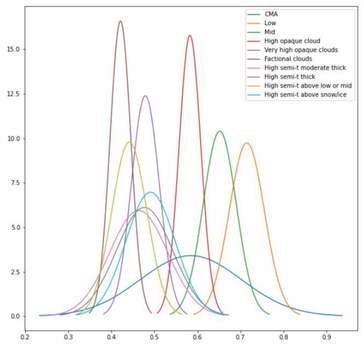

Figure 1: Gaussian Distribution of 11 Features

- Cloud temperature is closely related to cloud height and density (thickness or thinness) [Riihimaki et al . 2012]
- Preliminary experiment analyzes 11 cloud type features [Figure 1]
- The narrower standard deviation, the more related to cloud-top temperature
- Choose 5 features based upon [Figure 1]: cloud height and density, low-level, mid-level, high-opaque cloud, very high-opaque cloud, and fractional cloud

Methodology

Once we apply the image encode module to the fine-grained raster data, we apply SA-ConvLSTM to forecast cloud top temperature at each pixel region over time.

- SA-ConvLSTM:
- Additional self-attention memory module than ConvLSTM [Figure 3]
- Global spatial information on pixel images

- Major difference between SA-ConvLSTM and ConvLSTM: **feature aggregation** by taking two attention memory cells

- By aggregating features from the current hidden and previous memory state, this step covers all input positions by extracting features with long-range spatiotemporal dependencies.

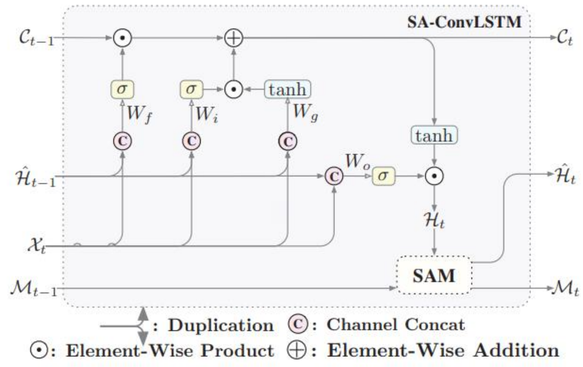

Figure 3: Self Attention- Conv LSTM
[Lin et al. 2020]

Overall Pipeline

We first implement the image encode and decode module to process the fine-scale raster image faster than the original image size processing.

- **Image encode module**: three blocks for reducing image size, and one block for linearly capturing spatial information
- **Image decode module** three transpose convolution blocks and one convolution block as a linear operation
- **Unet** (between image encode and decode): the decoder has more robust information to retrieve the original image picture

Therefore, the weather pixel images are less likely to lose spatial information among their' pixel values.

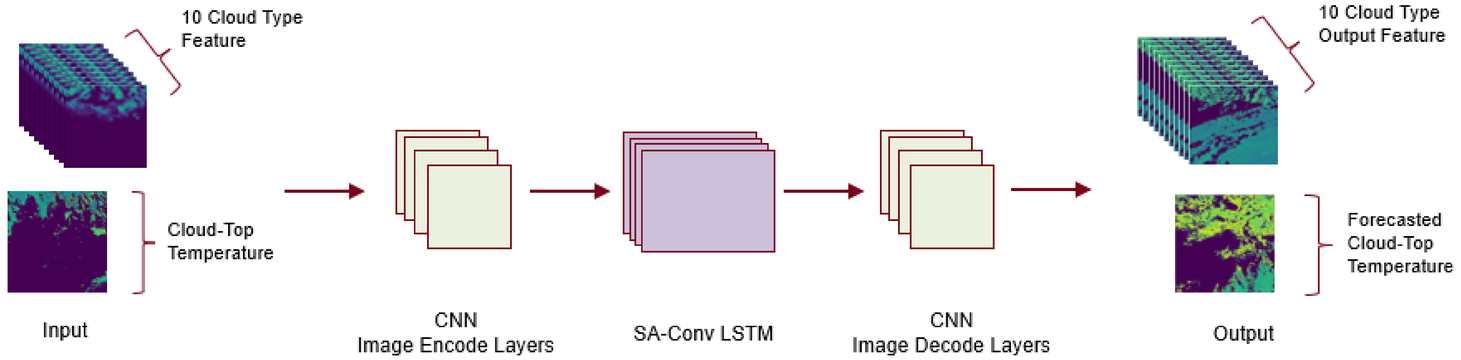

Figure 2: Overall Architecture

- Input: Feed past 1-hour cloud temperature, 10 different cloud type features
- CNN Image Encode Layers: Decrease original image size 256x256 to 32x32
- SA-Conv LSTM: Forecast next 1-hour cloud temperature
- CNN Image Decode Layers: Increase image size 32x32 to 256x256
- Output: Retrieve next 1-hour cloud temperature, 10 different cloud type features

Training Objective

$Loss_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$	Mean Squared Error(MSE) minimizes the difference between ground truth and prediction in cloud-top temperature
$Loss_{CrossEntropy} = -\sum_{i=1}^{N} y_i \log \hat{y}_i$	Cross Entropy (CE) calculates the difference between ground truth and prediction in ten different cloud type
$Loss_{KL} = y_i \log \frac{y_i}{\hat{y}_i}$	Kullback-Leibler divergence loss (KL loss) helps the model to have an ideal cloud-top temperature distribution when selected 5 cloud type features are existed

Table 1: Training Loss

$$Loss_{total} = Loss_{MSE} + Loss_{CrossEntropy} + Loss_{KL}$$

Therefore, we exploit the sum of three losses (MSE, CE, KL) to optimize the model's parameters during the training process.

Result

We compare our approach with three different training objectivity, which use (1) MSE, (2) MSE + CE, and (3) MSE + CE + KL loss. Then, we compare different objectivity using **MSE** and structural similarity index (**SSIM**) in the testing data. The smaller MSE and the higher SSIM, the better the model performance.

$$SSIM(y_i, \hat{y}_i) = \frac{(2\mu_{y_i}\mu_{\hat{y}_i} + c_1)(2\sigma_{y_i\hat{y}_i} + c_2)}{(\mu_{y_i}^2 + \mu_{\hat{y}_i}^2 + c_1)(\sigma_{y_i}^2 + \sigma_{\hat{y}_i}^2 + c_2)}$$

where μ_{y_i} is the average of y_i
$\mu_{\hat{y}_i}$ is the average of $\hat{y}_i$
$\sigma_{y_i}^2$ is the variance of y_i
$\sigma_{\hat{y}_i}^2$ is the variance of $\hat{y}_i$
$\sigma_{y_i\hat{y}_i}$ is the covariance of y_i and $\hat{y}_i$
$c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$ are two variables to stabilize the division with weak denominator;
L is the dynamic range of the pixel-values
$k_1 = 0.01$, $k_2 = 0.03$ by default.

	MSE	MSE + CE	MSE + CE + KL
Overall MSE	134.29	139.96	127.19
Overall SSIM	53.93	52.14	56.13

Table 2: Evaluation

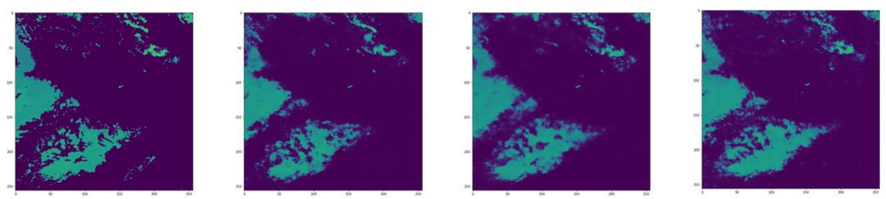

Figure 4: An example of forecasting result

- Our approach outperforms other baseline approaches
- This is the first real-world application that applied SA-ConvLSTM to capture spatial and temporal dependencies
- Our approach draws the most similar temperature distribution range with ground truth than other baseline approaches
- In the future, we forecast temperature with longer time point (next 4-hour) and test on different geo-coordinates regions