Deep learning based weather forecasting

Aditya Kothari Krishna Prasad Bhat

***Abstract*—This project aims to experiment with two major deep learning models to perfom the task of weather forecasting. The deep learning models experimented with are - ConvLSTM (Convolutional Long Short Term Memory) model and the classic UNet model.**

Link to codebase - https://github.com/askothar/ece5831-2023-final-project

Link to Dataset - https://tinyurl.com/ece5831projectdataset

I. Introduction

In the realm of meteorological sciences, the quest for accurate and reliable weather forecasting has been an enduring challenge. In pursuit of this objective, our project endeavors to harness the power of deep learning models to enhance the precision and efficiency of weather prediction tasks. This undertaking specifically focuses on the implementation of two distinct yet complementary deep learning architectures: the Convolutional Long Short-Term Memory (ConvLSTM) and a baseline U-Net model. Through the strategic deployment of these models, our aim is to evaluate and compare their performance in the intricate domain of weather forecasting. The ConvLSTM model, known for its ability to capture both spatial (owing to the convolutional layer) and temporal dependencies (owing to the LSTM layer), is poised to excel in capturing the dynamic nature of weather patterns. Simultaneously, the baseline U-Net model serves as a benchmark for comparison, allowing us to assess the relative strengths and weaknesses of each approach.

II. Related Work

A. Survey of existing methods

The project[1] addresses the global significance and challenges of weather forecasting, crucial for sectors like agriculture, military operations, and energy planning. Existing methods, encompassing traditional observations and advanced computer models, grapple with the inherent unpredictability of weather. Prior attempts using neural networks and mathematical models show promise but encounter limitations in explaining statistical relationships among past and future weather data. The survey of exisiting methods explores diverse methodologies, including Artificial Neural Networks, Decision Trees, Back Propagation, Recurrent Neural Networks, and Convolutional Networks. Researchers delve into the predictive power of machine learning algorithms (e.g., Naive Bayes, K-Means) and data mining techniques (e.g., chi-square). Notable studies predict specific weather parameters like rainfall, temperature, and humidity. The methodology employs data mining techniques for classification, featuring the Multivariate Adaptive Regression Splines model and other forecasting methods. The architecture adopts a comprehensive neural network approach to capture intricate weather parameter relationships. Experimental results focus on accuracy metrics, emphasizing the project's significance in understanding climate changes affecting daily life. The survey underscores the role of data mining in extracting insights from vast datasets, contributing to weather forecasting advancements, including techniques like K-Means clustering for pattern recognition.

B. Deep Learning (DL) forecasting models survey

The project [3] addresses the longstanding challenges in Earth system forecasting, emphasizing the profound impact of variabilities on daily life, including crop yields, transportation, and natural disasters. Current operational forecasting systems, notably the NOAA's rainfall nowcasting model, heavily rely on simulation-based approaches, limiting their adaptability to emerging geophysical observations and vast Earth observation data. In response, this study introduces Earthformer, a space-time Transformer for Earth system forecasting, leveraging deep learning instead of explicit physical models. The proposed Cuboid Attention, a building block for efficient space-time attention, decomposes input tensors into non-overlapping cuboids, mitigating the computational challenges associated with traditional Transformer architectures. Extensive experiments on synthetic datasets demonstrate the efficiency and effectiveness of Earthformer, showcasing its superiority in precipitation nowcasting and ENSO forecasting compared to existing baselines. The findings highlight Earthformer's potential for revolutionizing Earth system forecasting by combining the strengths of deep learning and efficient attention mechanisms. This is certainly a good approach however it uses the transformer architecture which is beyond the scope of our experiments with deep learning architectures.

III. Dataset and Data Exploration

The Storm EVent ImagRy (SEVIR) [2] dataset is a comprehensive collection of temporally and spatially aligned image sequences capturing various weather events over the contiguous US (CONUS). Acquired by the GOES-16 satellite and NEXRAD radars, the dataset encompasses five distinct image types, including visible satellite imagery, infrared satellite imagery at different wavelengths, a radar mosaic of vertically integrated liquid (VIL), and total lightning flashes detected by the geostationary lightning mapper (GLM). Each SEVIR event consists of a 4-hour sequence sampled at 5-minute intervals, covering 384 km x 384 km patches. The dataset includes

Identify applicable funding agency here. If none, delete this.

approximately 12,000 events that cover all five image types, with images stored in HDF5 files totaling 952 GB. SEVIR's spatial and temporal alignment, diverse image modalities, and coverage of severe weather events make it a valuable resource for addressing various meteorological challenges and applications in weather modeling.

Below mentioned are some VIL

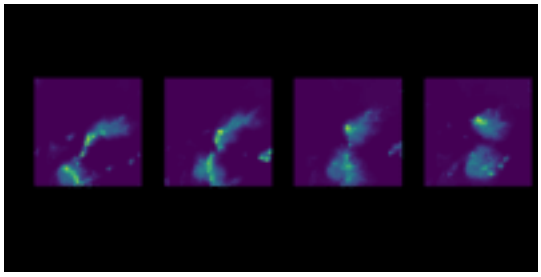

Fig. 1. Event ID: b'S696651', Sample VIL image.

The data uses LAEA projection to be visualised. The Lambert Azimuthal Equal Area (LAEA) projection (fig 2) is a map projection that preserves equal area and maintains true direction from the central point. It is commonly used for mapping polar regions due to its unique characteristic of minimizing distortion near the center of the projection. The projection is defined by specifying a central point on the map, and distances from this point are preserved accurately. However, distortions increase as one moves away from the central point, particularly towards the outer edges of the map. This projection is advantageous for applications that require accurate representation of areas, making it suitable for mapping regions near the chosen central location with minimal area distortion.

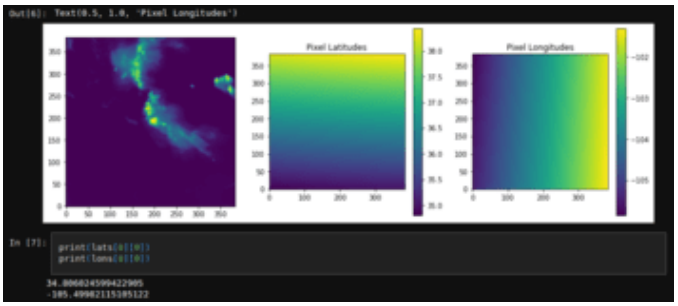

Fig. 2. LAEA Projection

IV. Model details - ConvLSTM and U-Net

We use two models in our experiments - U-Net to establish baseline and ConvLSTM to understand how spatio-temporal learning could be better than the performance of U-Net.

A. U-Net Segmentation Model

U-Net is a convolutional neural network architecture devised for semantic segmentation tasks. The U-Net structure resembles the letter "U," featuring a contracting path and an expansive path that meet at a bottleneck in the center. The contracting path comprises repeated blocks of two 3x3 convolutional layers followed by rectified linear unit (ReLU) activations and a 2x2 max-pooling operation, which progressively reduces spatial dimensions while capturing hierarchical features. This downsampling process is instrumental in extracting abstract representations from the input image. The expansive path involves upsampling the feature maps through transposed convolutions, followed by 2x2 up-sampling and a concatenation operation with the corresponding feature maps from the contracting path. Each block in the expansive path consists of two 3x3 convolutions with ReLU activations. The skip connections, formed by concatenating feature maps from the contracting path to the expansive path, facilitate the recovery of high-resolution details during segmentation.

At the center of the U-Net architecture lies the bottleneck or bridge, which consists of two 3x3 convolutions with ReLU activations, serving as the transition point between the contracting and expansive paths. The bottleneck plays a crucial role in retaining contextual information while preserving the spatial features necessary for accurate segmentation. Additionally, batch normalization is often incorporated after each convolutional layer to stabilize and accelerate training. The final layer typically involves a 1x1 convolutional layer with a softmax activation function to produce the segmentation map, assigning each pixel to a specific class. U-Net has proven to be highly effective in various image segmentation applications, particularly in medical image analysis, where it excels at delineating intricate structures within images.

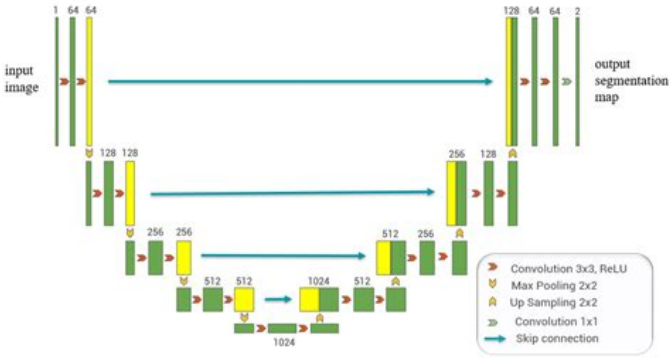

Fig. 3. U-Net Architecture

B. ConvLSTM Model

Convolutional LSTM (ConvLSTM) is an extension of the Long Short-Term Memory (LSTM) architecture, specifically designed to handle spatiotemporal data, such as video sequences or image sequences with temporal dependencies. Introduced by Xingjian Shi et al. in 2015, ConvLSTM incorporates convolutional operations into the LSTM cells, enabling the model to capture both spatial and temporal dependencies simultaneously. In a typical ConvLSTM cell, the input gate, forget gate, and output gate, which are fundamental components of traditional LSTMs, are modified to include

convolutional operations. The hidden state and cell state of the ConvLSTM are convolved with input data and input gate information, allowing the model to focus on relevant spatial information at each time step. This design makes ConvLSTM well-suited for tasks like video prediction, where understanding both spatial patterns and temporal dynamics is crucial.

The ConvLSTM structure involves a series of layers, with each layer containing multiple ConvLSTM cells. Each ConvLSTM cell, in turn, comprises convolutional layers within the gates and memory cell, facilitating the extraction of spatial features. The convolutional operations are typically 2D convolutions, considering the spatial nature of the data. The number of filters and kernel sizes in these convolutions can be adjusted based on the complexity of the task and characteristics of the input data. ConvLSTM layers can be stacked to capture hierarchical representations and intricate spatiotemporal patterns. The final layer of the ConvLSTM network is often a convolutional layer with an appropriate activation function, producing the output sequence or prediction. ConvLSTM has demonstrated success in various applications, including video prediction, action recognition, and precipitation nowcasting, leveraging its ability to model both spatial and temporal dependencies in sequential data.

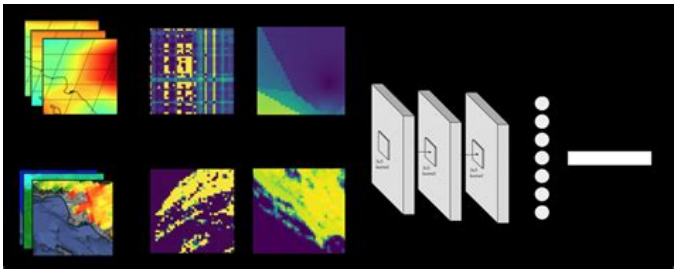

Fig. 4. ConvLSTM Architecture

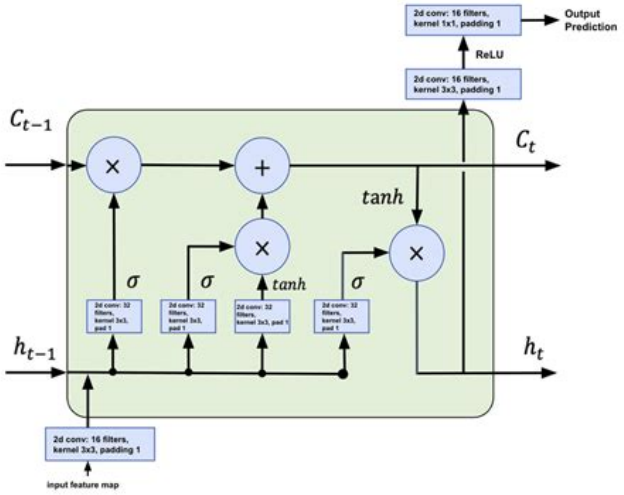

Fig. 5. ConvLSTM Detailed

V. Result and Methodology

A. Establishing a baseline with the U-Net architecture

We implement the UNet architecture for the specific task of nowcasting, which involves predicting short-term future weather conditions. The dataset is loaded from HDF5 files containing key meteorological variables, such as infrared and visible satellite images. The UNet architecture, composed of an encoder and decoder (subset of CNNs). The network's design allows it to efficiently capture and leverage both local and global contextual information in the input sequences.

In the training phase, the UNet model is optimized using the Adam optimizer and L1 loss, a regression loss function suitable for image prediction tasks. The code illustrates how the model is trained on a specified number of epochs using a DataLoader for efficient batch processing. Moreover, the script includes visualization components that showcase input data and model predictions. The color mapping and visualization techniques utilized provide insights into the spatial and temporal evolution of the predicted meteorological variables. The trained model is then saved to run and arrive at the baselines.

The flow of the program can be summarized in five key points:

1) **Data Loading:** The code begins by loading the meteorological dataset using the `SevirDataset` class, which organizes training and testing data along with corresponding metadata files.
2) **Model Definition:** The UNet architecture is defined in a class using PyTorch, comprising an encoder with convolutional layers, a decoder with transpose convolutional layers, and skip connections.
3) **Training Loop:** The model is trained using a custom training loop that iterates through batches of data from the training dataset. The Adam optimizer is employed with L1 loss for optimization.
4) **Visualization:** Matplotlib is used to visualize input images and corresponding model predictions. The code demonstrates how to display a subset of input sequences and their associated predictions.
5) **Model Saving and Animation:** The trained model is saved to a file, and the code creates a GIF animation from the model's output, showcasing the temporal evolution of predicted meteorological variables. The animation is generated using the `imageio` library, and the resulting GIF is saved as 'sample.gif'.

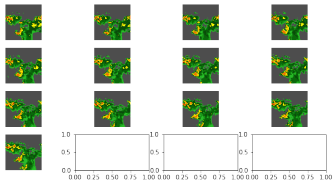

Fig. 6. Sample test data label

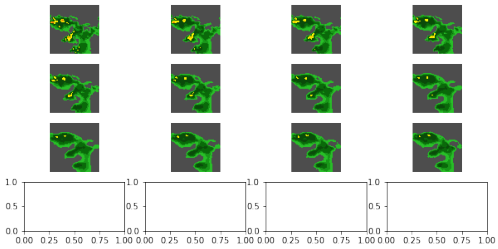

Fig. 7. Prediction on the sample test data, i.e. baseline established

B. Testing with ConvLSTM

We implement a Convolutional Long Short-Term Memory (ConvLSTM) neural network for sequence-to-sequence prediction, specifically applied to weather-related data, particularly VIL (Vertically Integrated Liquid) data. ConvLSTM architecture is designed to capture spatiotemporal dependencies in the VIL dataset, and the training loop updates the model parameters to minimize the L1 loss between predicted and ground truth frames. Finally, the code provides visualizations to assess the model's performance on the test set.

1) **Data Loading and Preprocessing:**
   - The code starts by importing necessary libraries, including PyTorch for neural network operations.
   - A custom dataset class, `SevirDataset`, is defined for loading and processing the weather dataset stored in HDF5 format. The dataset contains sequences of input and output frames related to infrared data, lightning, and VIL data.
   - The dataset is loaded using PyTorch's DataLoader, and a sample batch of input and output sequences is visualized using Matplotlib.
2) **ConvLSTM Neural Network Architecture:**
   - The ConvLSTM architecture is implemented using PyTorch. It consists of a custom ConvLSTMCell and a high-level ConvLSTM class.
   - The overall sequence-to-sequence model, `Seq2Seq`, is defined, consisting of multiple ConvLSTM layers followed by batch normalization. The final layer is a convolutional layer to produce the output sequence.
   - The model is instantiated as `net_2`, and its architecture is printed, showing the number of parameters in the network.
3) **Training Setup:**
   - The Adam optimizer is defined with a learning rate of 0.001, and the L1 loss function is used for optimization.
   - The training loop runs for 10 epochs, where for each epoch, the model is trained on batches of input and output sequences from the training dataset.
   - The training loss is printed, and the model is saved after training.
4) **Model Evaluation:**
   - The trained model is loaded, and a batch of input sequences from the test dataset is passed through the model to generate predictions.
   - The output sequences are visualized using Matplotlib to compare the predicted frames with the ground truth.
5) **Visualization and Results:** The visualizations include input and output frames during training and the predicted output frames after loading the trained model. The code employs Matplotlib to display the visualizations.

However we found that the prediction for ConvLSTM returns a null-set, leading to a very low accuracy.

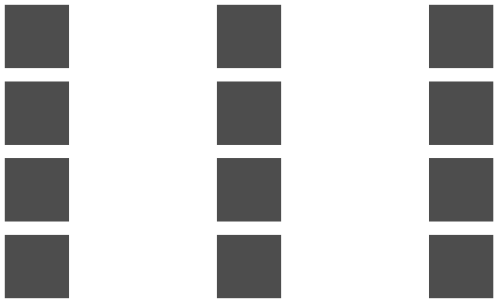

Fig. 8. Prediction for ConvLSTM returning a null set

C. Result discussion

The null results from the ConvLSTM training could stem from various issues, such as difficulties in capturing the spatiotemporal dependencies within the VIL dataset or challenges in convergence during training. To improve the ConvLSTM model's performance, several aspects can be considered. Firstly, a thorough analysis of the training loss and evaluation metrics could provide insights into where the model is struggling. Adjustments to hyperparameters, such as learning rate, may be necessary to facilitate convergence. Additionally, experimenting with different architectures or increasing the complexity of the ConvLSTM layers could help the model better capture the intricate patterns in the VIL data. Regularization techniques, such as dropout, might be employed to prevent overfitting, and the dataset itself could be scrutinized for any anomalies or issues that might affect training. Lastly, exploring alternative loss functions or combining the ConvLSTM with other architectures may offer a more robust solution for the sequence-to-sequence prediction task. Regular monitoring and fine-tuning of the training process based on performance metrics are essential in iteratively improving the model.

Scope for improvement:

1) **Hyperparameter Tuning:** Conduct a comprehensive analysis of hyperparameters, including learning rate, batch size, and the number of hidden units in the ConvLSTM cells. Adjustments to the learning rate can significantly impact convergence; for example, a high learning rate may lead to overshooting, while a low learning rate can slow down or hinder convergence.

Experimenting with different batch sizes could affect the model's generalization.

2) **Architecture Complexity:** Evaluate the impact of the ConvLSTM architecture's complexity on performance. Consider increasing the number of layers or hidden units within the ConvLSTM cells to enhance the model's capacity to capture intricate spatiotemporal patterns in the VIL dataset. However, be cautious of overfitting, and utilize techniques like dropout to regularize the model.
3) **Regularization Techniques:** Implement regularization techniques to mitigate overfitting. Dropout, a common regularization method, randomly drops units during training to prevent the network from relying too heavily on specific nodes. This could be particularly beneficial if the ConvLSTM is struggling to generalize from the training set to unseen data.
4) **Dataset Analysis:** Scrutinize the VIL dataset for anomalies or issues that might be adversely affecting training. This involves identifying outliers, ensuring data consistency, and handling missing or corrupted data. Preprocessing steps, such as normalization or data augmentation, can contribute to better model training and generalization.
5) **Loss Function Exploration:** Assess the impact of different loss functions on the ConvLSTM's training. Depending on the nature of the VIL prediction task, alternative loss functions, such as weighted loss or a custom loss function tailored to specific characteristics of VIL data, may lead to improved convergence. Additionally, investigating advanced loss functions like SSIM (Structural Similarity Index) designed for image data could be beneficial.

REFERENCES

[1] Deep Learning-Based Weather Prediction: A Survey" by Xiaoli Ren et. al., Big Data Research Journal 2023

[2] SEVIR : A Storm Event Imagery Dataset for Deep Learning Applications in Radar and Satellite Meteorology" by Mark S. Veillette, Siddharth Samsi, Christopher J. Mattioli, 34th Conference on Neural Information Processing Systems (NeurIPS 2020)

[3] Earthformer: Exploring Space-Time Transformers for Earth System Forecasting" by Zhihan Gao et. al., 36th Conference on Neural Information Processing Systems (NeurIPS 2022)