Explainable Weather Forecasting with Deep Spatial Autoencoders

Master's Thesis Project Part - 1

submitted by

Aviral Jain

(18AG3AI08)

under the guidance of

Dr. Adway Mitra (**Name of the supervisor**)

Centre for Excellence in Artificial Intelligence (CAI)

Indian Institute of Technology Kharagpur

Kharagpur, West Bengal, India - 721302

September - November 2022

Centre for Excellence in Artificial Intelligence (CAI)

Indian Institute of Technology Kharagpur

CERTIFICATE

This is to certify that the project report entitled "**Explainable Weather Forecasting with Deep Spatial Autoencoders**" submitted by Aviral Jain (Roll No. 18AG3AI08) to Indian Institute of Technology Kharagpur towards partial fulfillment of requirements for the award of the degree of B.Tech + M.Tech (Dual degree) is a record of bona fide work carried out by him under my supervision and guidance during Autumn Semester, 2022-23.

Date: 27/11/2022

Place: Kharagpur

Prof. Adway Mitra

(CAI)

Declaration

I certify that

(a) The work contained in this thesis is original and has been done by me under the guidance of my supervisor Dr. Adway Mitra.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have followed the guidelines provided by the institute in preparing the thesis.

(d) I have confirmed the norms and conditions given in the Ethical Code of Conduct of the Institute.

(e) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them citing them in the text of the thesis and giving their details in the References section.

Date: 27/11/2022 Aviral Jain

Place: Kharagpur (Student)

Table of Contents

Declaration

Table of Contents

List of abbreviations

1. Introduction
    1.1 General
    1.2 Justification
    1.3 Objectives

2. Literature Review

3. Theory
    3.1 Convolutional Neural Networks
    3.2 Autoencoders
    3.3 Spatial Transformation Network

4. Dataset and Methods
    4.1 Dataset
    4.2 Methodology
    4.2.1 Encoder Block
    4.2.2 Decoder Block
    4.2.4 Model Description (U-NET)
    4.2.5 Model Description (U-STN)
    4.2.6 Training DDWP Models

5. Results and Discussion

6. Plan of Work
    First Semester
    Second Semester

7. References

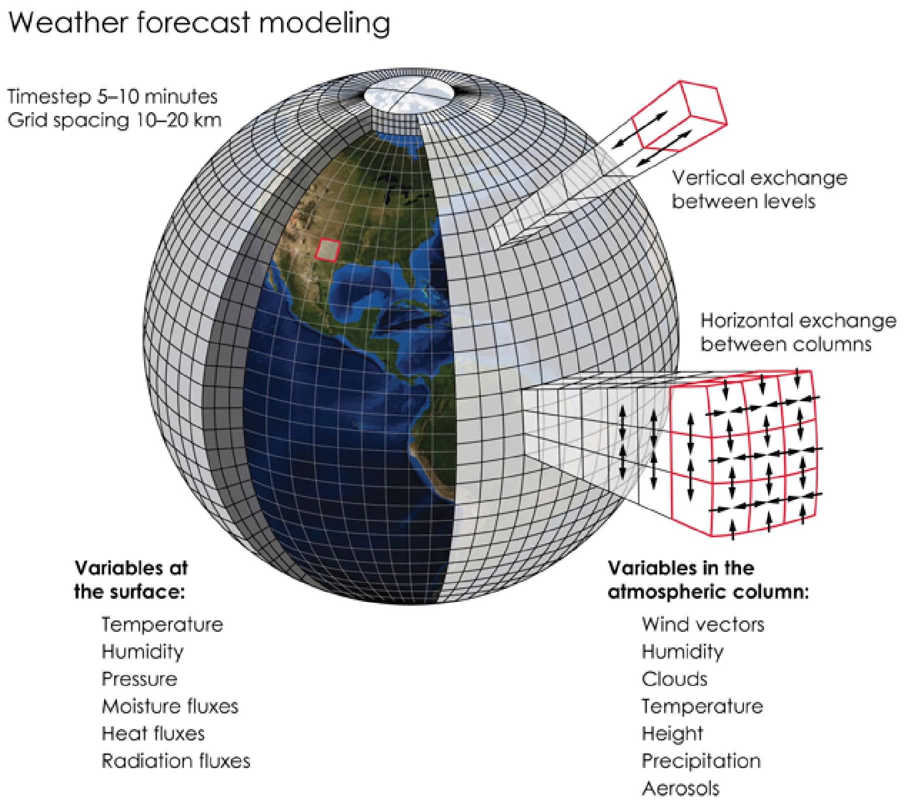
List of abbreviations

Term	Description
Autoregressive models	A model that iteratively predicts states of a system at new time steps by using the predicted state at the previous time step as input
CNN	Convolutional neural network: a type of neural network in which features are learned through successive convolutions and own sampling of the input as it maps to the output
DDWP	Data-driven weather prediction: a framework in which a data-driven model is trained on historical weather data and predicts future weather without solving physical equations of the atmosphere
U-NET (Encoder-Decoder)	A neural network in which the input is encoded into a low-dimensional representation (encoding) and then decoded back into high dimensional (often the same dimension as the input) space from the encoding
Equivariance	A property of a function that allows the output (of the function) to change appropriately in response to a transformation in the input
NWP	Numerical Weather Prediction
RNN	Recurrent neural network: a type of neural network in which information moves both forward and backward through the network
STN	Spatial transformer network: a neural network in which an affine transformation and subsequent interpolation allow the network to be equivariant
U-STN	U-NET with a spatial transformer connected to the latent space of the network

1. Introduction

1.1 General

Weather forecasting, is an important and indispensable procedure in people's daily lives, which evaluates the alteration happening in the current condition of the atmosphere. The development of precise numerical representations of the governing dynamical equations behind climate has been at the forefront of weather prediction science ever since the advent of the first completely programmable electronic digital computers at the end of WW2. Such numerical representations have now been improved to enable climate simulations based on fundamental principles. Since that time, numerical weather prediction (NWP) and climate modeling have dominated meteorological forecasting to the point where, just ten years ago, predictions using statistical empirical models are starting to seem somewhat out-of-date. Deep learning in particular has emerged as a powerful tool for prediction in many fields because of data-driven methodologies. Now, whether data-driven techniques can potentially be used to forecast global weather patterns days in advance is a question to ask.



1.2 Justification

Recent years have seen increased interest in using machine learning's (ML) and deep learning's quick algorithmic developments, particularly data-driven modeling, to enhance simulations and forecasts of nonlinear dynamical systems. Much of the success of deep learning is based on the ability of neural networks to recognize patterns in high-dimensional spaces. Fully data-driven weather prediction (DDWP) models that are trained using variables that describe the large-scale circulation collected from numerical models or reanalysis products have already demonstrated encouraging outcomes in a few experiments. These models make use of ML techniques like convolutional neural networks (CNNs) and/or recurrent neural networks (RNNs), which are trained on state variables that represent the spatiotemporal variability's historical development and then learn to predict the states of the future. There are many ways to incorporate certain physical characteristics into neural networks, such as physics-informed ML when simulating weather and climate. Some of the ways incorporate using specially created loss functions to enforce important conservation rules, appropriate symmetries, or portions (or even all) of the governing equations. For instance, conventional CNN architectures enforce translational and rotational symmetries, which may not necessarily exist in large-scale circulation. In fact, recent research in the ML community has shown that preserving a more general property called "equivariance" can improve the performance of CNNs. Equivariance-preserving neural network architectures learn the existence of (or lack thereof) symmetries in the data rather than enforcing them a priori and better tracking the relative spatial relationship of features.

1.3 Objectives

To provide proof of concept for the DDWP model, we use several climate variables (namely 500 hPa geopotential height (Z500), precipitation data, 2m temperature, 850hPa temperature) from the ECMWF Reanalysis 5 (ERA5) dataset. The DDWP model is trained on hourly samples of these variables from the year 1979 to the year 2017. The spatiotemporal evolution of these variables is then forecasted from precise initial conditions using the DDWP model. We build a spatial-transformer-based U-NET that can capture rotational and scaling features in the latent space for DDWP modeling and show the advantages of this architecture over a vanilla encoder–decoder U-NET.

2. Literature Review

[1] **Analog Forecasting of Extreme-Causing Weather Patterns Using Deep Learning** (Ashesh Chattopadhyay, Ebrahim Nabizadeh, Pedram Hassanzadeh) tell us about how their DDWP models trained on the General Circulation model (GCM) have proven to be successful in predicting extreme temperature events. They use a novel deep learning model Capsule Neural Networks (CapsNets) which are trained on mid-tropospheric large-scale circulation patterns (Z500). The trained networks predict the occurrence/region of cold or heat waves, only using Z500, 1–5 days ahead.

[2] **The ERA5 global reanalysis** (Hans Hersbach, Bill Bell, and others) introduced a new weather reanalysis data that replaces the ERA-Interim reanalysis (spanning 1979 onwards) that started in 2006. ERA5 benefits from a decade of developments in model physics, core dynamics, and data assimilation.

[3] **WeatherBench: A Benchmark Data Set for Data-Driven Weather Forecasting** (Stephan Rasp, Peter D. Dueben, Sebastian Scher, Jonathan A. Weyn, Soukayna Mouatadid, Nils Thuerey) presents a benchmark data set for data-driven medium-range weather forecasting (specifically 3–5 days). According to their research, the discretized equations governing the atmosphere serve as the foundation for conventional weather models. While they excel at many things, they nonetheless fall short on others. Deep learning techniques, which are data-driven methods, directly learn from the best available observations and may result in more accurate forecasts. In order to advance data-driven weather prediction and promote cross-disciplinary collaboration, they propose a benchmark task: predicting pressure and temperature globally, typically 3 to 5 days in advance.

[4] **Spatial Transformer Networks** (Max Jaderberg, Karen Simonyan, Andrew Zisserman, Koray Kavukcuoglu) introduce the Spatial Transformer network, which explicitly allows the spatial manipulation of data within the network. This module can be added to current convolutional architectures to enable neural networks to dynamically alter feature maps' spatial relationships, conditional on the feature map itself, without changing the optimization procedure or adding additional training supervision. They demonstrate how the employment of spatial transformers causes models to acquire invariance to translation, scale, rotation, and more generic warping, improving the performance on various benchmarks and for various types of transformations.

[5] **Can Machines Learn to Predict Weather? Using Deep Learning to Predict Gridded 500-hPa Geopotential Height From Historical Weather Data** (Jonathan A. Weyn, Dale R. Durran, and Rich Caruana) develop elementary weather prediction models using deep convolutional neural networks (CNNs) trained on past weather data to forecast one or two fundamental meteorological fields on a Northern Hemisphere grid with no explicit knowledge about physical processes. Their study summarises that even though simple models do not perform better than an operational weather model, machine learning warrants further exploration as a weather forecasting tool; in particular, the potential efficiency of CNNs make them attractive for ensemble forecasting.

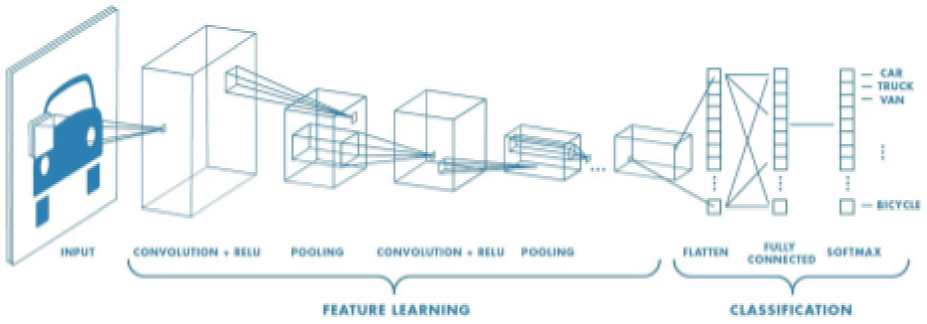
[6] **Sub-seasonal forecasting with a large ensemble of deep learning weather prediction models** (Weyn, J. A., Durran, D. R., Caruana, R., and Cresswell-Clay, N.) present an ensemble prediction system using a Deep Learning Weather Prediction (DLWP) model that recursively predicts key atmospheric variables with six-hour time resolution. They have developed a computationally efficient approach that can train ensembles of 100+ sizes which can forecast predictions of up to six weeks.

[7] **Towards physics-inspired data-driven weather forecasting** (A. Chattopadhyay, M. Mustafa, P. Hassanzadeh, E. Bach, and K. Kashinath) present a physics-inspired Data-Driven Weather Prediction model. Their method primarily consists of three parts: (1) a deep spatial transformer added to U-NET's space to capture rotation and scaling transformation in the latent space for spatiotemporal data (2) a data-assimilation (DA) algorithm to ingest noisy observations and improve the initial conditions for following forecasts and (3) a multi-time-step algorithm, which combines forecasts from DDWP models with various time steps through DA, increasing the accuracy of forecasts at short intervals.

3. Theory

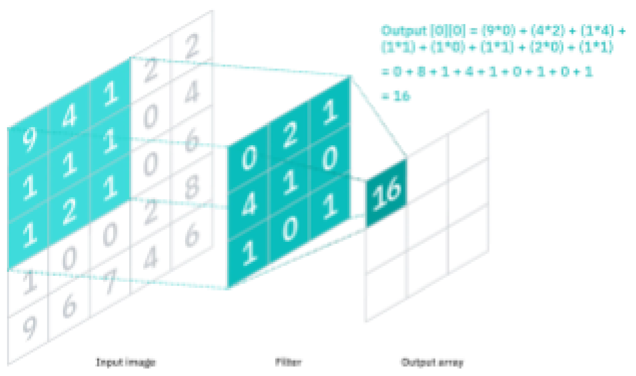
3.1 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a Deep Learning method that can take in an input image, give various elements and objects in the image importance (learnable weights and biases), and be able to distinguish between them. Comparatively speaking, a CNN requires substantially less pre-processing than other classification techniques. CNN's have the capacity to learn these filters and properties, whereas in primitive techniques filters are hand-engineered. Through the use of pertinent filters, a CNN may effectively capture the spatial and temporal dependencies in a picture. Because there are fewer factors to consider and the weights can be reused, the architecture provides a better fitting to the picture dataset. In other words, the network can be trained to better comprehend the level of complexity in the image.



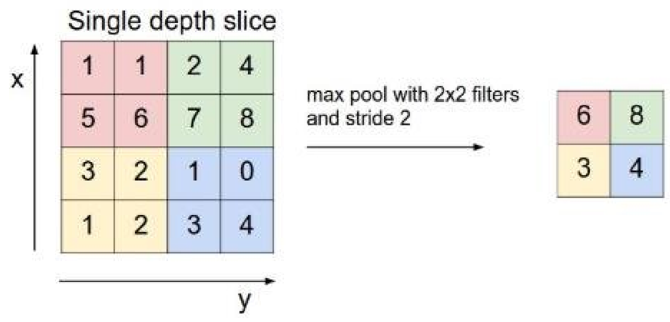
A typical Convolutional Neural Network architecture
It usually consists of convolutional, pooling, activation, and fully connected layers.

A Convolutional layer is responsible for extracting the features such as edges, from the input image. It uses several filters (kernels) to perform the convolution operation.


Output [0][0] = (9*0) + (4*2) + (1*4) + (1*1) + (1*0) + (1*1) + (2*0) + (1*1)
= 0 + 8 + 1 + 4 + 1 + 0 + 1 + 0 + 1
= 16
Input image
Filter
Output array

A simple Convolutional operation

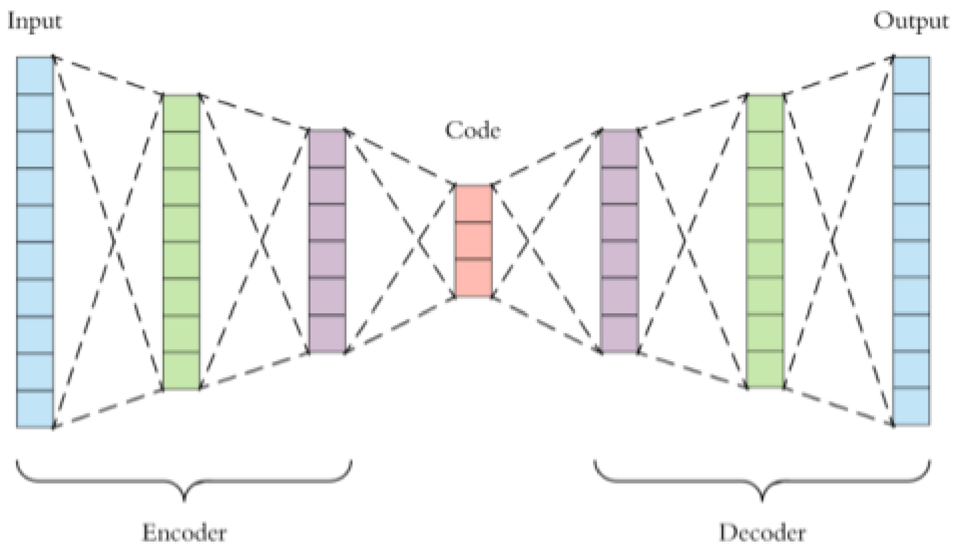
Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training the model.



After going through convolutional and pooling layers, we have successfully enabled the model to understand the features. Moving on, we flatten the final output and feed it to a regular Neural Network for learning purposes.

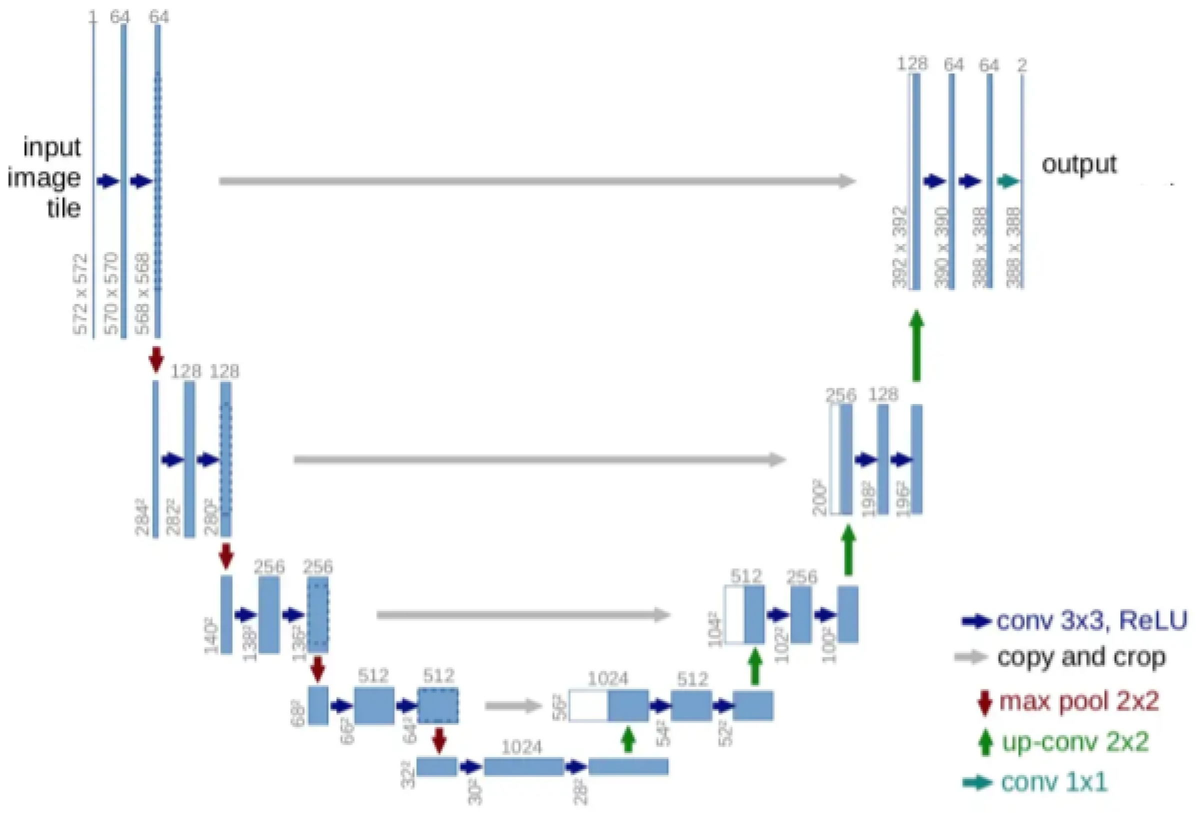
3.2 Autoencoders

Autoencoders are a specific type of feedforward neural network where the input is the same as the output. They compress the input into a lower-dimensional code and then reconstruct the output from this representation. The code is also called the latent-space representation or the bottleneck.



A typical diagram for an autoencoder

The bottleneck constrains the amount of information that can traverse the full network, forcing a learned compression of the input data. Because neural networks are capable of learning nonlinear relationships, this can be thought of as a more powerful (nonlinear) generalization of PCA. Whereas PCA attempts to discover a lower dimensional hyperplane that describes the original data, autoencoders are capable of learning nonlinear manifolds (a manifold is defined in simple terms as a continuous, non-intersecting surface).



U-NET Architecture, U-NET is one example of autoencoders

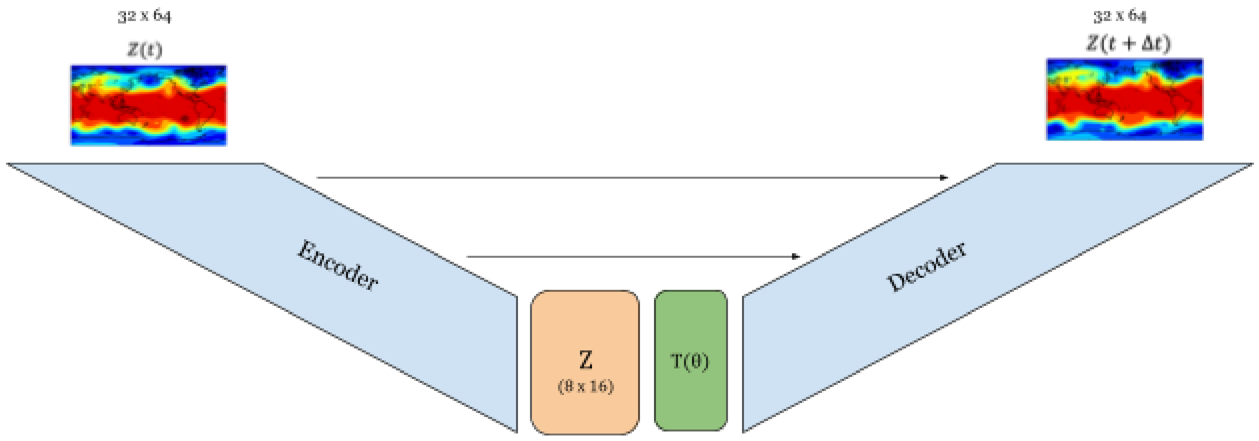
The loss function for an autoencoder is defined as follows

$$L(\theta, \phi) = \sum_{i=1}^{N} \left|\left|x_i - D_\theta(E_\phi(x_i))\right|\right|_2^2$$

where E is the encoder with parameters ϕ, D is the decoder with parameters θ, x_i is the input, and $||.||_2$ stands for L2 norm.

3.3 Spatial Transformation Network

This type of network captures rotation and scaling transformations between the input to the latent space and the decoded output, owing to the spatial transformer module implemented through the affine transformation, $T(\theta)$, along with the differentiable bilinear interpolation kernel.



A schematic of the architecture of the UNET based STN.

The spatial transformer applies an affine transformation $T(\theta)$ to the reduced coordinate system $(x_i^o,\ y_i^o)$ of the latent space to obtain a new transformed coordinate system $(x_i^s,\ y_i^s)$:

$$[x_i^s,\ y_i^s] \ = \ T(\theta)\,[x_i^o,\ y_i^o\ \ 1] \qquad (1)$$

where

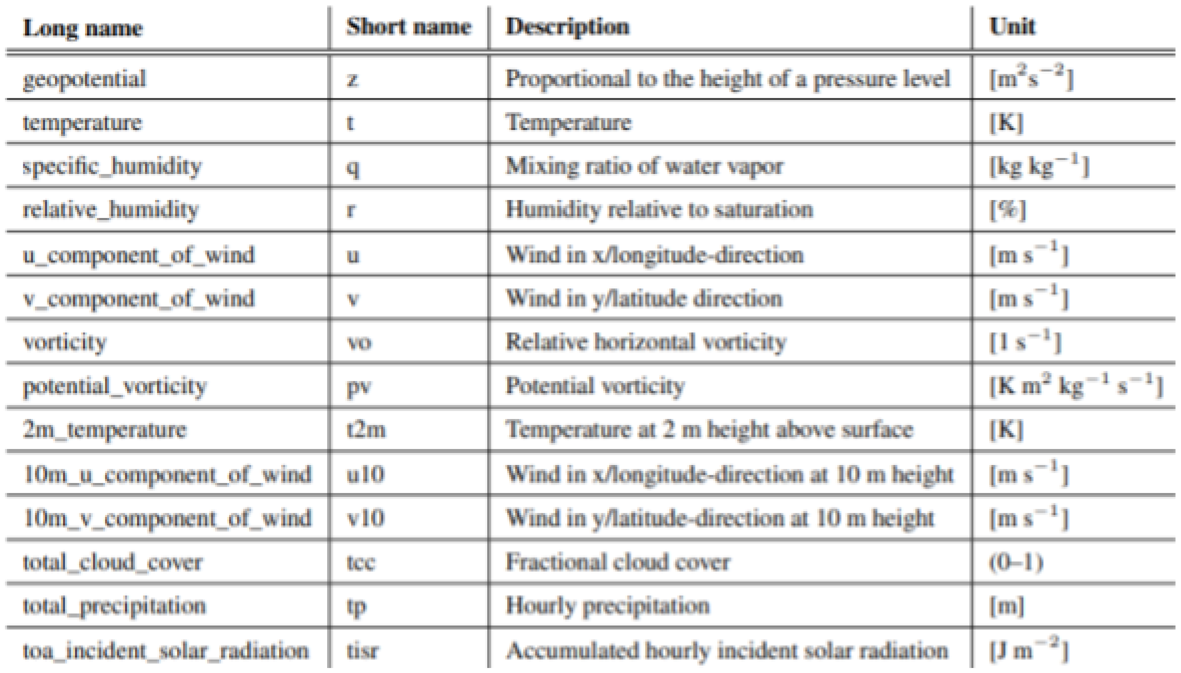
$$T(\theta) \ = \ \left[[\theta_{11}, \theta_{21}],\ [\theta_{12}, \theta_{22}],\ [\theta_{13}, \theta_{23}]\right] \qquad (2)$$

The parameters θ are predicted for each sample. A differentiable sampling kernel (a bilinear interpolation kernel in this case) is then used to transform $Z^{8\times16}$ which is on the old coordinate system $(x_i^o,\ y_i^o)$, into $\bar{z}^{8\times16}$ which is on the new coordinate system $(x_i^s,\ y_i^s)$.

4. Dataset and Methods

4.1 Dataset

We use the ERA5 Reanalysis dataset provided by WeatherBench, which provides the best guess of the atmospheric state at any point in time by combining a forecast model with the available observations. The dataset from the WeatherBench repository contains a global sample at every hour, from the year 1979 to the year 2018. These samples are downsampled to a rectangular longitude-latitude (x, y) grid of $(32, 64)$ to facilitate training. The dataset contains information about a number of climate variables like wind, cloud cover, temperature, humidity, geopotential height, precipitation, etc.



Long name	Short name	Description	Unit
geopotential	z	Proportional to the height of a pressure level	$[m^2 s^{-2}]$
temperature	t	Temperature	[K]
specific_humidity	q	Mixing ratio of water vapor	$[kg\ kg^{-1}]$
relative_humidity	r	Humidity relative to saturation	[%]
u_component_of_wind	u	Wind in x/longitude-direction	$[m\ s^{-1}]$
v_component_of_wind	v	Wind in y/latitude direction	$[m\ s^{-1}]$
vorticity	vo	Relative horizontal vorticity	$[1\ s^{-1}]$
potential_vorticity	pv	Potential vorticity	$[K\ m^2\ kg^{-1}\ s^{-1}]$
2m_temperature	t2m	Temperature at 2 m height above surface	[K]
10m_u_component_of_wind	u10	Wind in x/longitude-direction at 10 m height	$[m\ s^{-1}]$
10m_v_component_of_wind	v10	Wind in y/latitude-direction at 10 m height	$[m\ s^{-1}]$
total_cloud_cover	tcc	Fractional cloud cover	(0–1)
total_precipitation	tp	Hourly precipitation	[m]
toa_incident_solar_radiation	tisr	Accumulated hourly incident solar radiation	$[J\ m^{-2}]$

Description of the Weatherbench ERA5 dataset

We pick the following **variables** for our study: geopotential (z), total precipitation (tp), temperature (t), and 2m temperature (t2m). Here, each climate variable consists of data from 1979 to 2015 (315,360 samples) for training, 2016–2017 (17,520 samples) for validation, and 2018 (8,760 samples) for testing.

4.2 Methodology

4.2.1 Encoder Block

The network takes in an input snapshot, $Z(t)^{32\times64}$, as the initial condition and projects it onto a low-dimensional encoding space via a U-NET convolutional encoding block. The convolutions inside the encoder block account for Earth's longitudinal periodicity by performing circular convolutions on each feature map inside the encoder block. The encoded feature map, which is the output of the encoding block is sent to the spatial transformer module in the case of U-STN models.

4.2.2 Decoder Block

The decoding blocks bring the latent space $\bar{z}^{8\times16}$ back into the original dimension and coordinate system at time $t + \Delta t$, thus outputting $Z(t + \Delta t)^{32\times64}$. The concatenation of the encoder and decoder convolution outputs allows the architecture to learn the features in the small-scale dynamics of the variables better.

Finally, the objective loss function to be minimized is

$$L = \frac{1}{N+1} \times \sum_{t=0}^{t=N\Delta t} ||Z(t + \Delta t) - STN(Z(t), \lambda)||_2^2$$

where N is the number of training samples, $t = 0$ is the start time of the training set, and λrepresents the parameters of the network that are to be trained (in this case, the weights, biases, and θ of U-STNx).

In both encoding and decoding blocks, ReLU activation function is used. The number of convolutional kernels (32 in each layer), size of each kernel (5×5), Gaussian initialization, and the learning rate $\alpha = 3 \times 10^{-4}$ is chosen.

The model architecture for both U-NET and U-STN are described in detail as follows:

4.2.4 Model Description (U-NET)

```
______________________________________________________________________________
 Layer (type)                      Output Shape          Param #     Connected to
================================================================================================
 input_1 (InputLayer)          [(None, 32, 64, 1)]          0           []

 conv2d (Conv2D)               (None, 32, 64, 32)           832         ['input_1[0][0]']

 conv2d_1 (Conv2D)             (None, 32, 64, 32)           25632       ['conv2d[0][0]']

 max_pooling2d (MaxPooling2D)  (None, 16, 32, 32)           0           ['conv2d_1[0][0]']

 conv2d_2 (Conv2D)             (None, 16, 32, 32)           25632       ['max_pooling2d[0][0]']

 conv2d_3 (Conv2D)             (None, 16, 32, 32)           25632       ['conv2d_2[0][0]']

 max_pooling2d_1 (MaxPooling2D)  (None, 8, 16, 32)          0           ['conv2d_3[0][0]']

 conv2d_4 (Conv2D)             (None, 8, 16, 32)            25632       ['max_pooling2d_1[0][0]']

 conv2d_5 (Conv2D)             (None, 8, 16, 32)            25632       ['conv2d_4[0][0]']

 conv2d_6 (Conv2D)             (None, 8, 16, 32)            25632       ['conv2d_5[0][0]']

 conv2d_7 (Conv2D)             (None, 8, 16, 32)            25632       ['conv2d_6[0][0]']

 up_sampling2d (UpSampling2D)  (None, 16, 32, 32)           0           ['conv2d_7[0][0]']

 conv2d_8 (Conv2D)             (None, 16, 32, 32)           4128        ['up_sampling2d[0][0]']

 concatenate (Concatenate)     (None, 16, 32, 64)           0           ['conv2d_8[0][0]', 'conv2d_3[0][0]']

 conv2d_9 (Conv2D)             (None, 16, 32, 32)           51232       ['concatenate[0][0]']

 conv2d_10 (Conv2D)            (None, 16, 32, 32)           25632       ['conv2d_9[0][0]']

 up_sampling2d_1 (UpSampling2D)  (None, 32, 64, 32)         0           ['conv2d_10[0][0]']

 conv2d_11 (Conv2D)            (None, 32, 64, 32)           4128        ['up_sampling2d_1[0][0]']

 concatenate_1 (Concatenate)   (None, 16, 32, 64)           0          ['conv2d_11[0][0]', 'conv2d_1[0][0]']

 conv2d_12 (Conv2D)            (None, 32, 64, 32)           51232       ['concatenate_1[0][0]']

 conv2d_13 (Conv2D)            (None, 32, 64, 32)           25632       ['conv2d_12[0][0]']

 conv2d_14 (Conv2D)            (None, 32, 64, 1)            801         ['conv2d_13[0][0]']

=========================================================================================
Total params: 343,041

Trainable params: 343,041

Non-trainable params: 0
```

4.2.5 Model Description (U-STN)

```
________________________________________________________________________________________________________
Layer (type)                          Output Shape          Param #      Connected to
========================================================================================================
input_2 (InputLayer)                   [(None, 32, 64, 1)]  0            []
conv2d_8 (Conv2D)                     (None, 32, 64, 32)    832          ['input_2[0][0]']
conv2d_9 (Conv2D)                     (None, 32, 64, 32)    25632        ['conv2d_8[0][0]']
max_pooling2d_2 (MaxPooling2D)        (None, 16, 32, 32)    0            ['conv2d_9[0][0]']
conv2d_10 (Conv2D)                    (None, 16, 32, 32)    25632        ['max_pooling2d_2[0][0]']
conv2d_11 (Conv2D)                    (None, 16, 32, 32)    25632        ['conv2d_10[0][0]']
max_pooling2d_3 (MaxPooling2D)         (None, 8, 16, 32)    0            ['conv2d_11[0][0]']
conv2d_12 (Conv2D)                    (None, 8, 16, 32)     25632        ['max_pooling2d_3[0][0]']
conv2d_13 (Conv2D)                    (None, 8, 16, 32)     25632        ['conv2d_12[0][0]']
conv2d_14 (Conv2D)                    (None, 8, 16, 32)     25632        ['conv2d_13[0][0]']
conv2d_15 (Conv2D)                    (None, 8, 16, 32)     25632        ['conv2d_14[0][0]']
flatten_1 (Flatten)                   (None, 4096)          0            ['conv2d_15[0][0]']
dense_4 (Dense)                       (None, 500)           2048500      ['flatten_1[0][0]']
activation_4 (Activation)             (None, 500)           0            ['dense_4[0][0]']
dense_5 (Dense)                       (None, 200)           100200       ['activation_4[0][0]']
activation_5 (Activation)             (None, 200)           0            ['dense_5[0][0]']
dense_6 (Dense)                       (None, 100)           20100        ['activation_5[0][0]']
activation_6 (Activation)             (None, 100)           0            ['dense_6[0][0]']
____________________________________________STN Layer_______________________________________________
up_sampling2d (UpSampling2D)          (None, 16, 32, 1)     0            ['tf.reshape_6[0][0]']
conv2d_16 (Conv2D)                    (None, 16, 32, 32)    160          ['up_sampling2d[0][0]']
concatenate (Concatenate)             (None, 16, 32, 64)    0            ['conv2d_16[0][0]', 'conv2d_11[0][0]']
conv2d_17 (Conv2D)                    (None, 16, 32, 32)    51232        ['concatenate[0][0]']
conv2d_18 (Conv2D)                    (None, 16, 32, 32)    25632        ['conv2d_17[0][0]']
up_sampling2d_1 (UpSampling2D)         (None, 32, 64, 32)   0            ['conv2d_18[0][0]']
conv2d_19 (Conv2D)                    (None, 32, 64, 32)    4128         ['up_sampling2d_1[0][0]']
concatenate_1 (Concatenate)           (None, 32, 64, 64)    0            ['conv2d_19[0][0]', 'conv2d_9[0][0]']
conv2d_20 (Conv2D)                    (None, 32, 64, 32)    51232        ['concatenate_1[0][0]']
conv2d_21 (Conv2D)                    (None, 32, 64, 32)    25632        ['conv2d_20[0][0]']
conv2d_22 (Conv2D)                    (None, 32, 64, 1)     801          ['conv2d_21[0][0]']
============================================================================================================
Total params: 2,513,229
Trainable params: 2,513,229
Non-trainable params: 0
```

4.2.6 Training DDWP Models

The WeatherBench dataset has hourly data for our four chosen climate variables (geopotential, total_precipitation, 2m temperature, t850). The DDWP model takes input at a particular time t ($Z(t)$ thereafter) as the input and predicts $Z(t + \Delta t)$, which is then used as the input to predict $Z(t + 2\Delta t)$, and the autoregressive process continues as needed. We use Δt as 1 hour and 12 hours. Note that from now on "x" in U-STNx (and U-NETx) indicates the Δt (in hours) that is used, for example, U-STN1 uses Δt = 1h.

The table below depicts the *time taken (mins)* to train each model on the respective variable:

variable/model	U-NET1	U-NET12	U-STN1	U-STN12
z500	72	89	74	92
t2m	61	91	66	110
t850	63	88	69	94
tp	67	92	71	-

The system specifications used to perform training are as follows:

CPU: AMD Ryzen 4600H (6 core)	GPU: Nvidia GTX 1650 (4GB)	RAM: 16GB
Tensorflow 2.10.0	Keras 2.10.0	

For both models, we use mean squared error as our loss function. We use the Adaptive Momentum (Adam) Optimizer with a learning rate $\alpha = 3 \times 10^{-4}$ for updating the model weights. Early Stopping is also added to the model as a callback function so as to prevent any overfitting and any redundant training.

5. Results and Discussion

To benchmark our models, we use the Anamoly Correlation Coefficient (ACC) between the prediction and the ground truth and compare the quality of model predictions with time. Anomaly Correlation Coefficient (ACC) is one of the most widely used measures in the verification of spatial fields. ACC is defined as follows:

$$\mathrm{ACC} \equiv \frac{\sum_{i=1}^{n} w_i \left(f_i - \overline{f}\right)(a_i - \overline{a})}{\sqrt{\sum_{i=1}^{n} w_i \left(f_i - \overline{f}\right)^2 \sum_{i=1}^{n} w_i (a_i - \overline{a})^2}}, \quad (-1 \leq \mathrm{ACC} \leq 1),$$

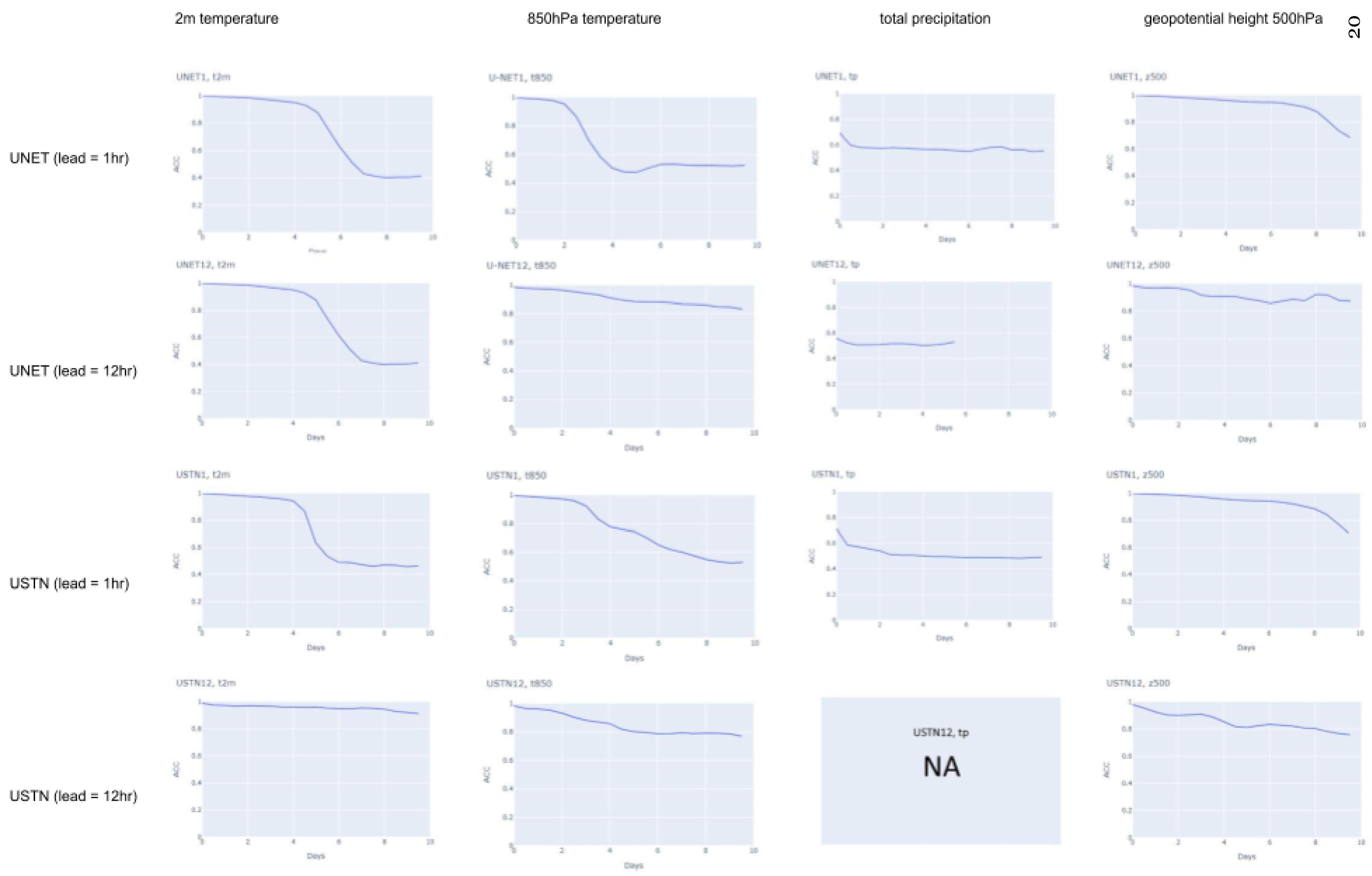
where n is the number of samples, and f_i, $\overline{f}$, a_i and $\overline{a}$ are given by the following equations:

$$f_i = F_i - C_i, \quad \overline{f} = \left(\sum_{i=1}^{n} w_i f_i\right) \Big/ \sum_{i=1}^{n} w_i,$$
$$a_i = A_i - C_i, \quad \overline{a} = \left(\sum_{i=1}^{n} w_i a_i\right) \Big/ \sum_{i=1}^{n} w_i,$$

where F_i , A_i , and C_i represent forecast, verifying value, and reference value such as climatological value, respectively. ACC lies in the range $(-1, 1)$, where 1 represents high correlation and -1 represents poor correlation.

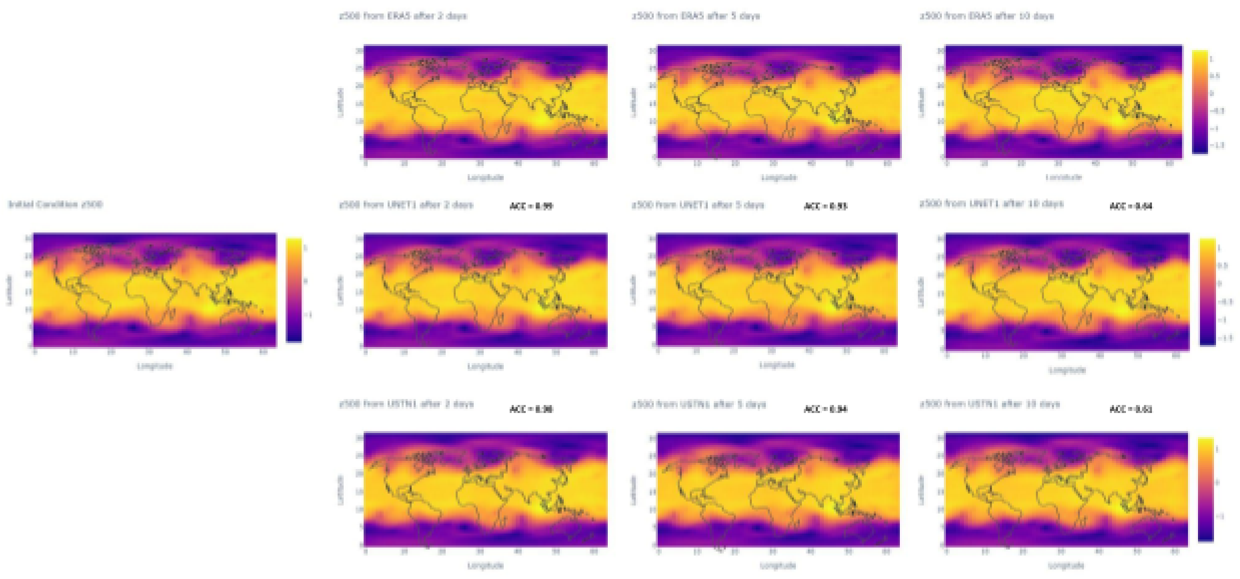
We compare the results of UNETx and USTNx (for x = 1hr and 12hr) for all four climate variables that we have chosen, i.e., geopotential height (z500), 850hPa temperature (t850), total precipitation (tp), 2m temperature (t2m).

In the 16 images shown on the next page, the x-axis represents time in days, and the y-axis represents the Anomaly Correlation Coefficient (ACC) between the corresponding model and ground truth, for the respective climate variable.


2m temperature
850hPa temperature
total precipitation
geopotential height 500hPa
UNET (lead = 1hr)
UNET (lead = 12hr)
USTN (lead = 1hr)
USTN (lead = 12hr)
UNET1, t2m
U-NET1, t850
UNET1, tp
UNET1, z500
UNET12, t2m
U-NET12, t850
UNET12, tp
UNET12, z500
USTN1, t2m
USTN1, t850
USTN1, tp
USTN1, z500
USTN12, t2m
USTN12, t850
USTN12, tp
NA
USTN12, z500
ACC
Days

Some key observations from these plots are

1. For the variable "2m temperature", we see that the ACC drops quite significantly after 4-5 days from the initial condition. But only in the case of USTN12 (lead = 12hr), the model performance is excellent even after 9-10 days.
2. For the variable "temperature 850hPa", we see that both UNET1 and USTN1 have poor ACC values after 3-4 days, whereas UNET12 and USTN12 have performed substantially better than the previous two.
3. For the variable "total precipitation", we see that neither of the models is successfully able to generate good predictions. This may imply that precipitation data has a different sub-structure than the rest of the variables.
4. Lastly, for the variable "geopotential height (500hPa)", we see that almost all of the models are able to generate sufficiently good predictions.


z500 from ERA5 after 2 days
z500 from ERA5 after 5 days
z500 from ERA5 after 10 days
Initial Condition z500
z500 from UNET1 after 2 days
ACC = 0.99
z500 from UNET1 after 5 days
ACC = 0.93
z500 from UNET1 after 10 days
ACC = 0.64
z500 from USTN1 after 2 days
ACC = 0.98
z500 from USTN1 after 5 days
ACC = 0.94
z500 from USTN1 after 10 days
ACC = 0.61
Latitude
Longitude

Examples of the spatiotemporal evolution of Z500 predicted using USTN1 and UNET1 and compared with the truth from ERA5. For the predicted patterns, the anomaly correlation coefficient (ACC) is shown above each panel

Overall, the results of Figs. 3 and 4 show the advantages of using the spatial transformer enabled U-STN in DDWP models. It is important to note that it is difficult to assert whether the transformation with $T(\theta)$ in the latent space actually leads to physically meaningful transformations in the decoded output. However, we see that the performance of the model improves with the addition of the spatial transformer module.

6. Plan of Work

First Semester

6.1	Built U-NETx and U-STNx (x=1hr, 12hr) DDWP models for predicting climate variables from weather reanalysis data.	Completed
6.2	Compared the performance of developed models in terms of Anamoly Correlation Coefficient for all four climate variables	Completed

Second Semester

6.3	Implement Data Assimilation algorithm for a combined DDWP+DA pipeline	TBD
6.4	Implement a multi-timestep framework to improve predictions using newly observed data	TBD

7. References

[1] **Analog Forecasting of Extreme-Causing Weather Patterns Using Deep Learning** (Ashesh Chattopadhyay, Ebrahim Nabizadeh, Pedram Hassanzadeh)

[2] **The ERA5 global reanalysis** (Hans Hersbach, Bill Bell, and others)

[3] **WeatherBench: A Benchmark Data Set for Data-Driven Weather Forecasting** (Stephan Rasp, Peter D. Dueben, Sebastian Scher, Jonathan A. Weyn, Soukayna Mouatadid, Nils Thuerey)

[4] **Spatial Transformer Networks** (Max Jaderberg, Karen Simonyan, Andrew Zisserman, Koray Kavukcuoglu)

[5] **Can Machines Learn to Predict Weather? Using Deep Learning to Predict Gridded 500-hPa Geopotential Height From Historical Weather Data** (Jonathan A. Weyn, Dale R. Durran, and Rich Caruana)

[6] **Sub-seasonal forecasting with a large ensemble of deep learning weather prediction models** (Weyn, J. A., Durran, D. R., Caruana, R., and Cresswell-Clay, N.)

[7] **Towards physics-inspired data-driven weather forecasting** (A. Chattopadhyay)