2024-05-01

Modeling The Spatiotemporal Variations Of The Magnetic Field In Active Regions On The Sun Using Deep Neural Networks

Godwill Asare Mensah Mensah
University of Texas at El Paso

MODELING THE SPATIOTEMPORAL VARIATIONS OF THE MAGNETIC FIELD IN ACTIVE REGIONS ON THE SUN USING DEEP NEURAL NETWORKS

GODWILL AMANKWA ASARE MENSAH

Doctoral Program in Computational Science

APPROVED:

Olac Fuentes, Ph.D., Chair

Amy Wagler, Ph.D.

Felicia Manciu, Ph.D.

Stephen Crites, Ph.D.
Dean of the Graduate School

Dedication

to my

MOTHER and FATHER

with love

MODELING THE SPATIOTEMPORAL VARIATIONS OF THE MAGNETIC FIELD IN ACTIVE REGIONS ON THE SUN USING DEEP NEURAL NETWORKS

by

GODWILL AMANKWA ASARE MENSAH

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

Computational Science Program

THE UNIVERSITY OF TEXAS AT EL PASO

MAY 2024

Acknowledgements

I am thankful to the Almighty God for his mercies and goodness to this point in my education.

I would like to express my deep-felt gratitude to my advisor, Dr. Olac Fuentes of the Computer Science Department at the University of Texas at El Paso, for his advice, encouragement, enduring patience, and constant support. He was never ceasing in his belief in me. I also wish to thank the other members of my committee, Dr. Amy Wagler of the Public Health Sciences Department and Dr. Felicia Manciu of the Physics Department, both at the University of Texas at El Paso. Their suggestions, comments, and additional guidance were invaluable to the completion of this work.

Additionally, I want to thank the Computational Science Program for the support given to me in completing this thesis.

I am also grateful to my loved ones, family, and research group for their support and contribution to this work.

Abstract

Solar active regions are areas on the Sun's surface that have especially strong magnetic fields. Active regions are usually linked to a number of phenomena that can have serious detrimental consequences on technology and, in turn, human life. Examples of these phenomena include solar flares and coronal mass ejections, or CMEs. The precise prediction of solar flares and coronal mass ejections is still an open problem since the fundamental processes underpinning the formation and development of active regions are still not well understood.

One key area of research at the intersection of solar physics and artificial intelligence is deriving insights from the available datasets of solar activity that can help us understand solar active regions better. Some machine learning models have been employed to forecast solar flares from a 6-hour to 48-hour time span, thanks to advancements in artificial intelligence. Support Vector Machine (SVM) [5, 42], K-Nearest-Neighbor (KNN) [27], Extremely Randomized Trees (ERT) [36], and deep neural network [35] are some of the machine learning models that have been used in forecasting solar flares, but the results are not good. This is due to the models being trained with a specific set of active region parameters and an imbalanced dataset with few positive flare cases.

As a result, there is a need to understand space weather and the basis by which these events occur. In this study, we applied a deep learning architecture originally designed for video prediction to predict the changes happening on the Sun in continuous time by using time series Helioseismic and Magnetic Imager data captured by the Solar Dynamics Observatory (SDO) and compared it against a no-change baseline and a regression baseline.

In addition, we expanded our study to examine the changes in active regions by incorporating the 3D viewing geometry and the sun's rotation, which helped the models focus on the changes in the active regions. We proposed using log-scale normalization to normalize the data and using the Cascading Convolutional Neural Network to predict the changes in

active regions. To improve the performance of the model, we included the gradient information and the Structural Similarity Index in the training of the model by adding them as part of the loss function.

In this dissertation, we demonstrated that deep neural networks can be trained to predict changes in active regions. It is our hope that further development of this work will lead to a better understanding of various physical phenomena related to space weather.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The Sun's dynamic conditions and events in the space around Earth and in our upper atmosphere play a significant role in human life. These phenomena are known as space weather. Different kinds of space weather are caused by solar activities such as solar flares, coronal mass ejections, high-speed solar wind, and solar energetic particles. The solar magnetic field is the primary source of all these solar activities.

1.1 Active Regions

1.1.1 What Is An Active Region

An active region is an area on the Sun with an especially strong magnetic field. Solar flares and coronal mass ejections (CMEs) are known to originate from active regions on the Sun. These active regions are most frequently observed during the peak of the solar cycle (sunspot cycle) when the Sun's magnetic field is highly disturbed. Bright areas seen in X-ray and ultraviolet images of the Sun represent active regions. Intense bursts of energy are released from the powerful magnetic fields surrounding active regions, often taking the form of high-energy X-ray and UV photons. Sunspots are visible indications of active regions, but it is important to note that not all active regions have sunspots. Additionally, solar prominences and coronal loops may also appear around active regions [49].

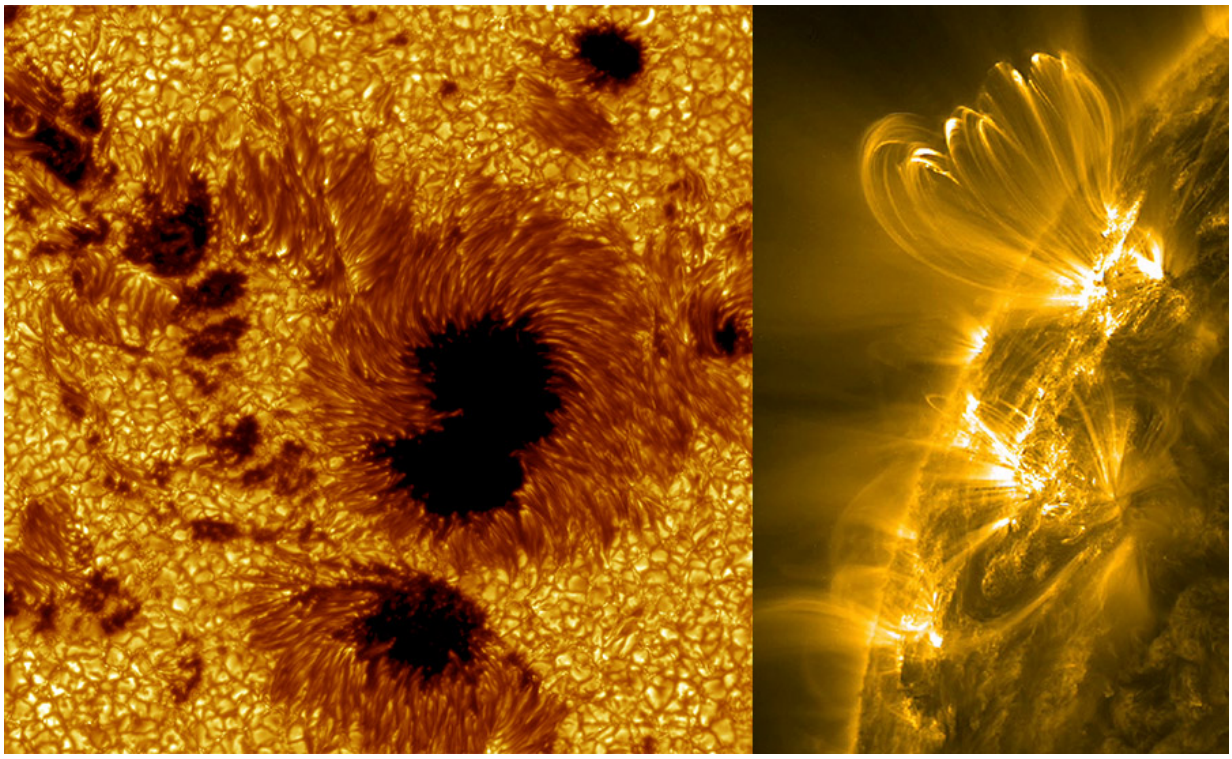

Figure 1.1: Sunspots (left) and Coronal loops (Right).
Credit: Royal Swedish Academy of Sciences - Göran Scharmer and Mats Löfdahl (sunspots) and NASA/SDO and the AIA, EVE, and HMI science teams (coronal loops)

1.1.2 What Is A Sunspot

Sunspots are regions on the surface of the Sun found in the photosphere that appear darker and also cooler compared to other areas on the Sun. Sunspots are known for their dark hue, which can be attributed to their average temperature of 3,800 degrees Kelvin, much lower than the average temperature of 5,800 degrees Kelvin in other areas. Despite their seemingly small size in solar images, sunspots can have a diameter of up to 50,000 kilometers. While the exact process of sunspot formation remains unclear, it is believed to occur in regions of the Sun that experience intense magnetic activity. [52] (Figure 1.1).

1.2 Solar Flares

1.2.1 What Are Solar Flares

A sudden release of distorted magnetic fields that produces a huge amount of energy and drives that energy into space creates a sudden flash of light known as a solar flare. Solar flares are regarded as the most significant explosive events within our solar system. The duration of flares might range from a few minutes to several hours. This electromagnetic radiation propagates at the speed of light. Occasionally, the energy that is discharged causes a rapid increase in the speed of extremely energetic particles, such as protons and electrons. These high-energy particles take tens of minutes to reach the Earth. [34].

Earth's volcanic explosions produce energy that is less than a millionth of that released by a solar flare. Solar flares are generally more visible due to their powerful X-ray and ultraviolet emissions, however, they can also be observed in white light.

1.2.2 Classification Of Solar Flares

Solar eruptions and the accompanying storms can release significant amounts of radiation that can impact Earth or travel long distances through space. To help classify these events, the National Oceanic and Atmospheric Administration (NOAA) has established categories based on the size and intensity of solar flares. These categories use letters A, B, C, M, and X, with X-class flares being the most powerful and A-class flares being the weakest. Similar to the Richter scale used for earthquakes, each letter represents a 10-fold increase in energy output. A, B, and C flares typically have minimal effects on Earth, while M-class flares can lead to brief radio blackouts at the poles and minor radiation storms that could pose a risk to astronauts. Table 1.1 shows the various categories and their respective magnitudes in W/m^2.

Table 1.1: Solar Flare Classes

Class	W/m^2 **between 1 & 8 Ångströms**
A	$< 10^{-7}$
B	$\geq 10^{-7} < 10^{-6}$
C	$\geq 10^{-6} < 10^{-5}$
M	$\geq 10^{-5} < 10^{-4}$
X	$\geq 10^{-4}$

1.2.3 Prominent Solar Flare Events

Solar flares exert a significant impact on the immediate space weather surrounding the Earth. Solar proton events refer to the ability of solar wind or stellar wind to generate streams of extremely energetic particles. These particles have the potential to influence the Earth's magnetosphere. In addition, significant solar flares are occasionally followed by coronal mass ejections (CMEs), which can initiate geomagnetic storms capable of disabling satellites and disrupting terrestrial electric power grids for prolonged durations. These instances have been documented for the past 150 years. The following are some of the severe solar flare episodes that were felt on Earth or detected by satellites in orbit:

1. An exceptionally powerful solar flare incident occurred on September 1, 1859. The phenomenon is sometimes referred to as the "Carrington event". The occurrence was initially documented by Richard Carrington, a prominent solar astronomer from England. As a consequence of that incident, energetic particles penetrated Earth's atmosphere and overwhelmed the planet's magnetic field, leading to extensive devastation on the surface. Global telegraph system experiencing widespread disruption. The telegraph offices were set ablaze by the ignition. A vibrant display of aurora was observed at latitudes close to the tropics, specifically over Cuba, the Bahamas, Jamaica, El Salvador, and the Hawaiian Islands. Despite the disconnection of batteries,

the transmission of messages persisted due to the induction of electric currents in the lines caused by Aurora. [8].

2. August 1972 saw one of the strongest solar storm series in recorded history. High energy particles, solar flares, and geomagnetic storms were linked to the solar storms. Sea mines in Vietnam were activated by this storm. This incident interfered with the satellite, the electrical grid, and communication. This incident took place between the lunar missions of Apollo 16 and Apollo 17. Had this occurred during the journey, the particles may have struck astronauts outside of Earth's shielding magnetic field, potentially posing a serious risk to their lives.

3. A more serious case of space weather phenomena is the breakdown of the Hydro-Québec power grid on March 13, 1989 as a result of geomagnetically induced currents (GICs). Attributable to a damaged transformer, this incident resulted in a widespread power outage lasting in excess of 9 hours, impacting a population exceeding 6 million residents. This event was caused by a geomagnetic storm, which originated from a coronal mass ejection (CME) that was expelled from the Sun on March 9, 1989. [34]

4. Between mid-October and early November 2003, there was a significant solar flare outburst, which was one of the largest. This solar flare incident is the largest ever recorded by the Geostationary Operational Environmental Satellite (GOES) system. The GOES detectors were overwhelmed by this event, resulting in a projected classification. The flare caused the Sun's magnetic field lines to elongate, and subsequently, they exceeded their maximum stretching capacity. Consequently, a huge blast erupted on the surface of the Sun, resulting in coronal mass ejections (CME). Coronal Mass Ejections (CMEs) have the ability to release billions of metric tons of ionized gas and subatomic particles into space, traveling at a velocity of five million miles per hour. This event caused a break in the satellite-based communication infrastructure. This occurrence caused a blackout in Sweden lasting 90 minutes. Air traffic controllers modified the flight path to circumvent elevated altitudes in the vicinity of the polar

regions.

5. The solar storm of 2012 was an exceptionally massive and powerful coronal mass ejection (CME) event that took place on July 23rd of that year. This storm narrowly missed the Earth with a deviation of roughly nine days, due to the Sun's equator rotating on its axis every 25 days. Consequently, the area from where the outburst originated was not directly facing the Earth during that period. The magnitude of the eruption was similar to the 1859 Carrington incident, which resulted in global disruption of electrical infrastructure, mostly telegraph stations.

1.2.4 Goal

In this research work, we seek to predict the changes in active regions by employing deep learning methods created for video predictions. For this purpose, we will utilize Solar Dynamics Observatory (SDO) images produced from Helioseismic and Magnetic Imager instrument.

1.2.5 Significance of The Goal

The scientific inquiry into the fundamental process driving the emergence of active regions remains an open problem, prompting extensive research efforts to construct models and forecast the dynamics in these regions. Researchers such as Bobra et al. [5], Qahwaji et al. [42], Li et al. [27] and Nishizuka et al. [35] have been leading the way in developing techniques for feature selection of parameters of active regions and utilizing machine learning algorithms to forecast solar flares. Also, Rempel et al. [45] have investigated the use of numerical simulations in modeling active region scale flux emergence. Despite the numerous technological improvements, there remains a limited comprehension of the phenomenon surrounding active regions. The objective of this project is to forecast the fluctuations in active regions in order to enhance comprehension of active regions and advance the accuracy of solar flare prediction.

1.3 Research Questions

In this research, we are looking for the answers to the following questions:

1. Could deep learning models designed for solving next-frame prediction be adapted to predict the changes in active regions and produce practical results better than conventional baseline approaches?

2. Could we incorporate the knowledge of the 3D viewing geometry and the Sun's rotation to simplify the learning process of predicting the changes in active regions?

3. Could we improve the predictions of the changes in active regions by modifying the architecture of deep learning models designed for video prediction to exploit information from previous time steps?

1.4 Research Contributions

This research presents the following contributions:

1. We present the optimal normalization scale to use in predicting changes in active regions.

2. We show that deep learning models for Next-Frame Prediction can predict changes in active regions.

3. We develop a convolutional neural network (CNN) to predict the changes in the magnetic field of active regions using sequences of solar images.

4. We propose a loss function to improve the performance of the model by incorporating gradient information and the structural similarity index of the predictions.

1.5 Outline

This write-up is structured as follows. Chapter 2 describes the background of this dissertation. Chapter 3 presents some related works to this research and Chapter 4 describes the data used. In Chapter 5, we explore the changes of active regions with respect to the Sun and the results obtained. Chapter 6 presents our proposed model and normalization technique to predict changes in active regions. Finally, Chapter 7 summarizes the findings, a review of the results, and proposals for future research.

Chapter 2

Background

Neural networks are computational models that mimic the structure and function of the human brain. The system comprises interconnected nodes, or neurons, arranged in layers. Every neuron receives input signals, processes them, and produces an output signal. In artificial intelligence (AI) and machine learning, there are three fundamental approaches: supervised learning and unsupervised learning, but a third is a hybrid approach called semi-supervised learning.

Supervised learning involves using labeled data to train algorithms to predict output labels based on input attributes. Unsupervised learning analyzes unlabelled data to identify patterns or structures present in the dataset. Semi-supervised learning combines labeled and unlabeled data to enhance learning performance, particularly in situations when acquiring labeled data is difficult. Each method utilizes specific algorithms and strategies designed for various data kinds and learning goals.

Since the beginning of neural networks, numerous models have been developed to solve a particular objective, utilizing the aforementioned basic calculations. Among them, Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) are particularly notable and serve as the foundation for many others. Currently, attention-based networks have been developed, which have replaced RNN for sequence/time-based predictions.

2.1 Convolutional Neural Network (CNN)

2.1.1 Convolution Operation

The convolution operation is usually denoted by an asterisk. The convolution of f and g is defined as follows

$$s(t) = (f * g)(t) = \int_{-\infty}^{\infty} f(a)g(t-a)da \tag{2.1}$$

and described as weighted average of $f(a)$ at the moment t where the weighting is given by $g(-a)$ with a shift of amount t. The weighting function emphasizes different parts of the input function based on the value of t.

The function f is known as the input, the function g is called the kernel and the output is referred to as the feature map in convolutional network terminology.

t will be discretized while working with data and then t can only take on integer values. In that case the discrete convolution can be defined as:

$$s(t) = (f * g)(t) = \sum_{a=-\infty}^{\infty} f(a)g(t-a) \tag{2.2}$$

In cases where the input is a multidimensional array, the kernel is likewise a multidimensional array. Tensors are the term used to describe these multidimensional arrays. The input and kernel parts are kept in different locations. It is presumed that both the input and kernel functions have a value of zero everywhere except for a finite set of places where the values are recorded. Hence, the infinite summation can be represented as a summation over a limited number of array elements. As a result, convolution is defined as follows:

$$S(i,j) = (F * G)(i,j) = \sum_{m}\sum_{n} F(m,n)G(i-m, j-n) \tag{2.3}$$

One of the excellent properties of convolution is the commutative property. So the previous equation is equivalent to (2.4).

$$S(i,j) = (G * F)(i,j) = \sum_m \sum_n F(i-m, j-n)G(m,n) \tag{2.4}$$

The following figure shows how 2-D convolution works, where x is the input and y is the feature map produced with h'' as the kernel.

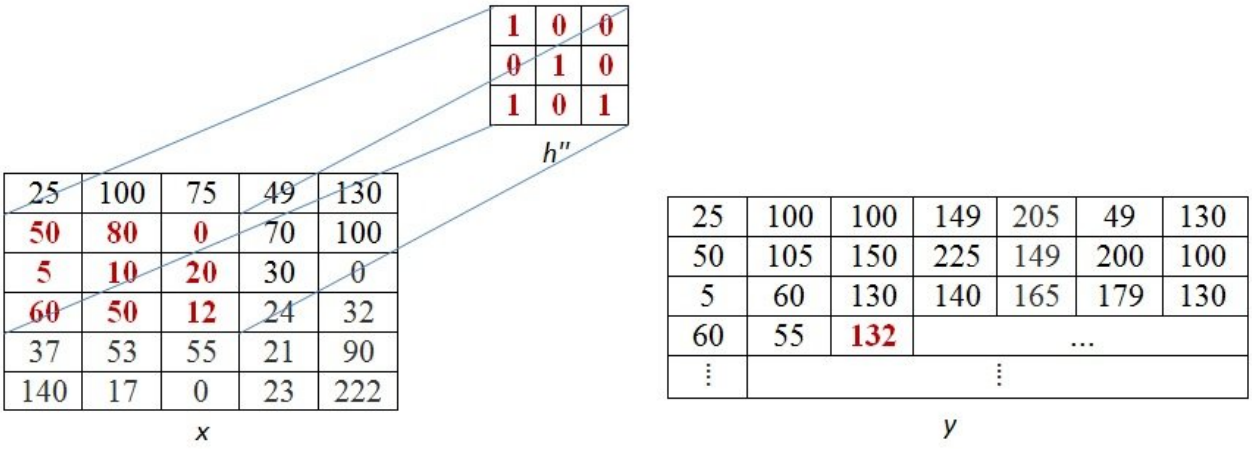

Figure 2.1: An example of 2-D convolution

2.1.2 Pooling

Following the execution of several simultaneous convolutions, a collection of linear activations is generated. Each linear activation is subsequently passed through a nonlinear activation function, such as the rectified linear unit. This step is referred to as the detector stage. Subsequently, the pooling function is employed to substitute the network's output at a specific position with a concise statistical representation of the neighboring outputs. Pooling can be implemented using several non-linear functions, such as max pooling, average pooling, L2 norm, and weighted average. Max pooling involves selecting the highest value within a rectangular region, while average pooling calculates the mean value within

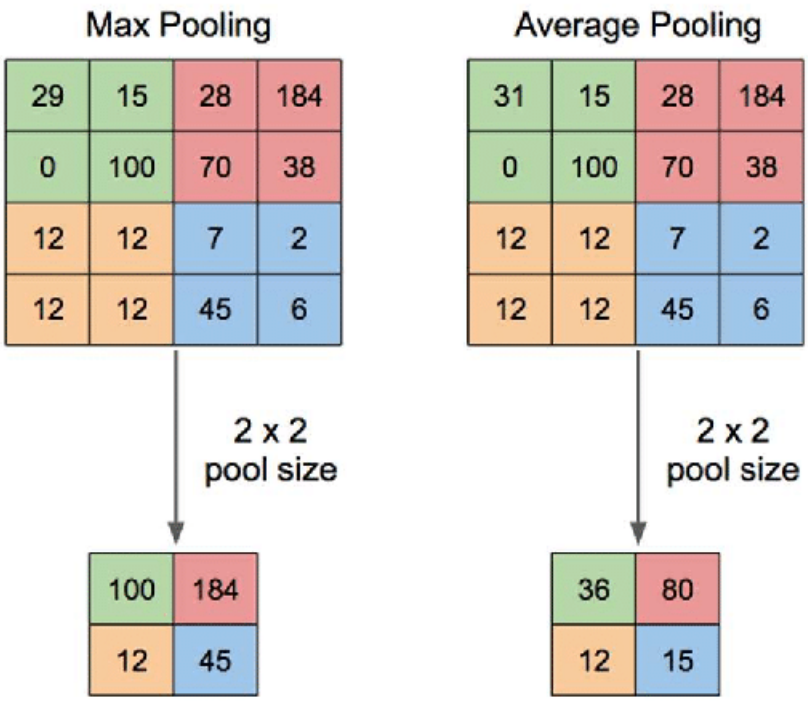

Figure 2.2: Max pooling vs Average Pooling

a rectangle region.

The significance of a certain feature is presumed to be greater than its precise positioning in relation to other features. The translation invariance quality is highly beneficial in this scenario. Pooling enhances the representation by making it nearly invariant to translations.

Pooling over spatial regions produces invariance to translation, but pooling over the outputs helps the features to learn which transformations to become invariant to.

Pooling units are typically less in number compared to detector units due to the utilization of summary statistics for pooling regions that are separated k pixels apart, as opposed to being spaced 1 pixel apart. This implies that the subsequent layer has roughly k times less inputs to handle, resulting in computational and statistical efficiency. This also decreases the memory demands for storing the bottlenecks.

2.1.3 Activation

Sigmoid or Logistic Activation Function

The mathematical form of sigmoid function is

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.5}$$

This activation function is very popular among deep learning researchers and engineers because of its advantages -

- The sigmoid function is inherently nonlinear, hence any linear combination of this function will likewise be nonlinear. This function exhibits a continuous and gradual change in its gradient as well.
- This function takes any real value as input, and output is always in the range $(0, 1)$, while the range for linear function is $(-\infty, \infty)$.
- From the figure 2.3 of the sigmoid function, it is clear that the function is very steep in the range $(-2, 2)$. Within this range, the function will exhibit a substantial variation in response to a minor alteration in z. As a result, there will be a positive gradient. Furthermore, it is evident that the output of this function falls within the range of 0 to 1, which is advantageous for classifiers as it allows for clear differentiation in predictions.

The main drawback of this activation function is outside of the range $(-2, 2)$, the function changes very little for any amount of change in z. Consequently, the gradient will approach zero. The issue at hand is commonly referred to as the "vanishing gradients" problem. This property is unfavorable. In this scenario, the gradient reaches a value of 0, resulting in the absence of signal propagation to the weights and, subsequently, to the data.

Hyperbolic tangent

Mathematically hyperbolic tangent is expressed as

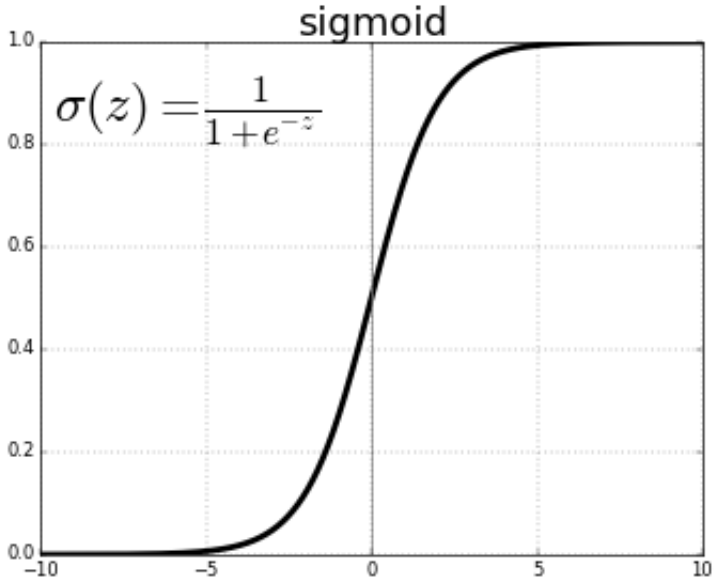

Figure 2.3: Sigmoid

$$tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{2.6}$$

which can be written as

$$tanh(x) = 2\sigma(2x) - 1 \tag{2.7}$$

That means *tanh* is a scaled sigmoid function. This activation function has comparable qualities to the sigmoid function. This function accepts any real value as input and produces an output within the $(-1, 1)$ range. The gradient of this function is stronger than the sigmoid function. In addition to this, *tanh* is zero-centered, unlike the sigmoid function.

Rectified Linear Units (ReLU)

The Rectified Linear Unit (ReLU) is the most prevalent and extensively employed activation function. The function is defined as the positive part of its argument, $R(x) = max(0, x)$. That means this function takes a real-valued number, and the output is x if x is greater than or equal to zero and zero if x is less than zero.

Sigmoid or *tanh* activation function is computationally expensive since it processes

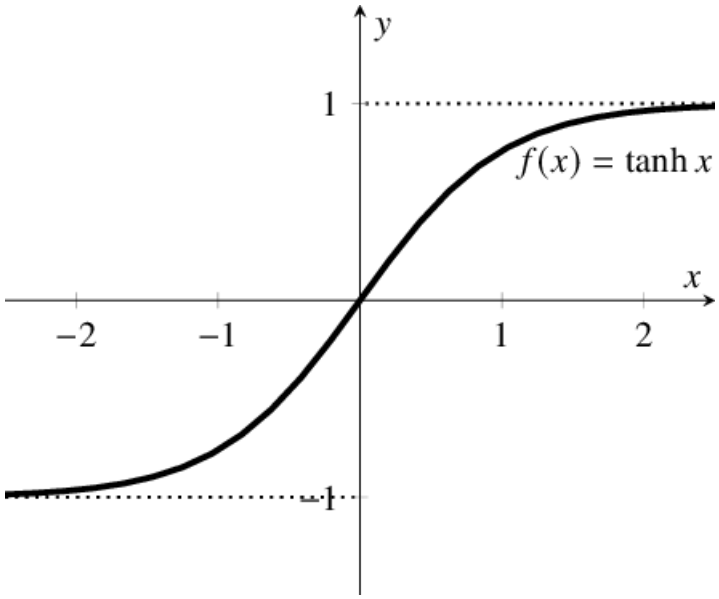

Figure 2.4: Hyperbolic tangent

practically all of the activations in a network, resulting in a dense output description. ReLU enhances the efficiency of the network by selectively deactivating specific neurons, hence making the network lighter.

The characteristic of the ReLU is that for negative values of x, this function is zero, which means the gradient is zero. This will result in not adjusting the weights of the neurons during backpropagation. So these neurons will never activate again. This gives rise to the "dying ReLU" problem.

Leaky ReLU

This function was introduced as a solution to the issue of "dying ReLU". In order to address this issue, the leaky ReLU activation function is specifically constructed to have a slight negative slope at the point where the ReLU function would otherwise output zero, that is, at $x < 0$. The range of the Leaky ReLU is $(-\infty, \infty)$. Usually, the slope is set to be at 0.01. If the slope is not at 0.01, then it is called Randomized ReLU. This leak, as seen in figure 2.6, helps to increase the range of the ReLU function. In addition, the slope can be defined as a parameter for each neuron, as demonstrated in the Parametric Rectified

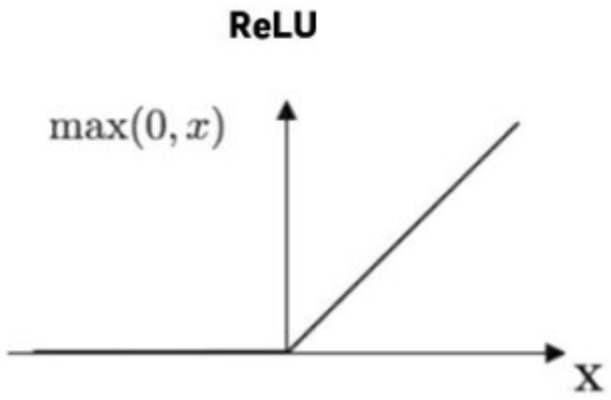

Figure 2.5: Rectified Linear Units

Linear Unit (PReLU).

Leaky ReLU is defined as

$$\begin{cases} ax & \text{if } x < 0 \text{ where a is } 0.01 \\ x & \text{if } x \geq 0 \end{cases} \tag{2.8}$$

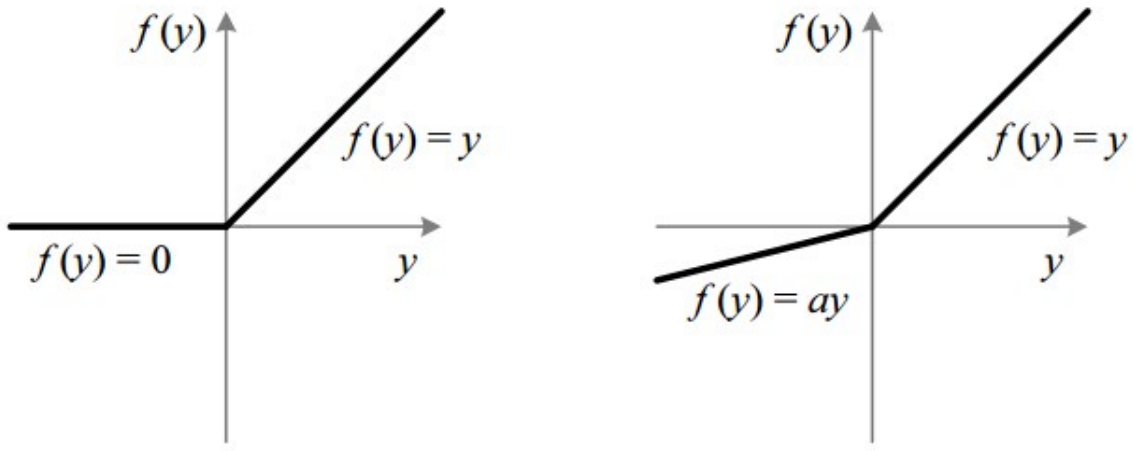

Figure 2.6: ReLU vs Leaky ReLU

ELU

The Exponential Linear Unit (ELU) is a mathematical function that approaches zero more rapidly and yields more precise outcomes. Similar to Leaky ReLU, ELU also necessitates a positive alpha constant. The presence of negative inputs is what sets ELU apart from RELU. If the inputs are non-negative, they both conform to the identity function. Unlike ELU, RELU exhibits a sudden smoothing effect, while ELU progressively smooths until its output reaches the value of -α.

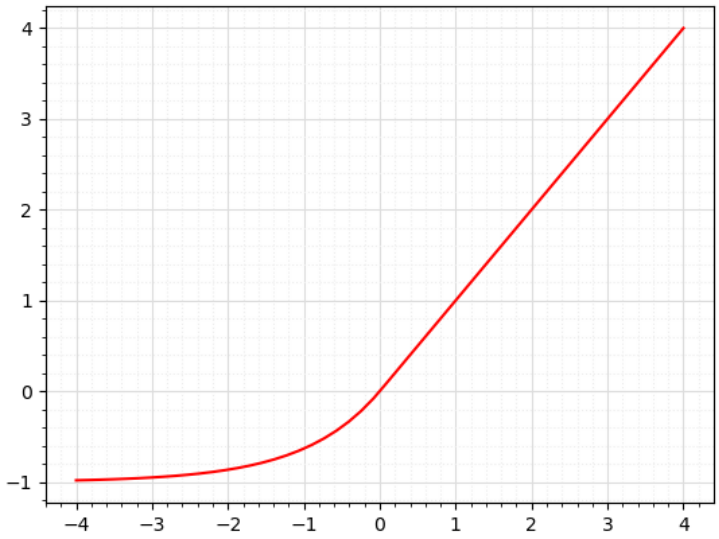

Figure 2.7: Exponential Linear Unit (ELU) with $\alpha = 1$

Swish

As an activation function, Swish has the formula $f(x) = x \cdot sigmoid(\beta x)$, where β is an adjustable parameter that can be acquired through learning. In most implementations, β is not used, in which case $x\sigma(x)$ is used, which is essentially $\beta = 1$. The function is a continuous, non-linear function that has a lower bound and no upper bound. It consistently performs as well as or outperforms the Rectified Linear Unit (ReLU) on deep neural networks.

GELU

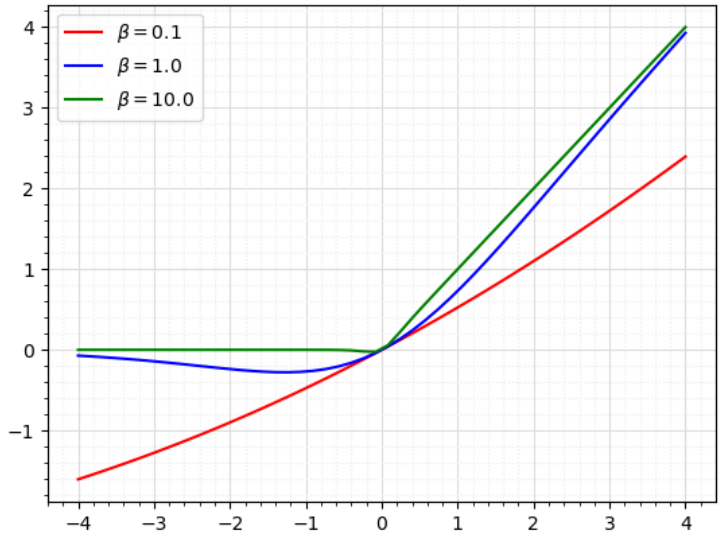

Figure 2.8: Swish with $\beta = 0.1, 1.0, 10.0$

GELU, which stands for Gaussian Error Linear Unit, is $x\Phi(x)$, where $\Phi(x)$ is the standard Gaussian cumulative distribution function. The GELU non-linearity assigns weights to inputs based on their value rather than gating inputs based on their sign, like ReLUs. The formula for GELU is:

$$\text{GELU}(x) = 0.5x\left(1 + \tanh\left(\frac{2}{\pi}\left(x + 0.044715x^3\right)\right)\right) \tag{2.9}$$

GELU has garnered prominence in transformer-based designs, namely in the domain of natural language processing applications, where it has demonstrated enhanced performance in comparison to alternative activation functions such as ReLU or sigmoid. GELU is characterized by its computational efficiency and ease of implementation.

Linear

A linear function in which the activation is directly proportional to the input, which is the weighted sum from the neuron.

2.1.4 Loss Function

Loss functions quantify the accuracy of predictions. The goal is to minimize the loss function. The gradients are computed by utilizing the loss function, and the weights are subsequently adjusted based on these gradients. The neural network is trained by means of this update.

Mean Squared Error (MSE) is employed as the loss function to quantify the discrepancy between the expected outcome and the actual outcome. It is mostly used in regression tasks. MSE is given by,

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{2.10}$$

where $y_i's$ are the true values and $\hat{y}_i's$ are the predicted values.

Another loss function is **Mean Absolute Error (MAE)**, which is given by,

$$MAE = \mid y_i - \hat{y}_i \mid, \tag{2.11}$$

where $y_i's$ and $\hat{y}_i's$ are the true and predicted values, respectively.

Binary Crossentropy Loss is used for Binary Classification tasks. This loss is implemented using the *sigmoid* activation function.

Categorical Crossentropy is also used in multi-class classification tasks. The final layer output is passed through a $softmax$ activation so that each node outputs a probability value between (0 -1). The softmax is represented as

$$\sigma(\vec{z_i}) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \text{ where } z \text{ is the final layer} \tag{2.12}$$

2.2 Recurrent Neural Network

A Recurrent Neural Network (RNN) is a specific type of neural network that is designed to handle a sequence of values $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(n)}$. This has the capacity to handle sequences of varying lengths. Recurrent neural networks employ the concept of parameter sharing to

distribute the parameters among various components of the model. Sharing information is essential when there is a correlation between data that appears in multiple locations within a sequence.

The output of a recurrent network is a sequential series, where each element of the output is determined by the preceding elements of the output. The output is derived by applying the same rule to the prior outputs.

Multiple types of recurrent neural networks exist. For example:

Example - 1 : Recurrent networks exhibit recurrent connections among hidden units, resulting in an output being generated at each time step (Figure 2.9). This function assigns a set of input values, denoted as x, to a corresponding set of output values, denoted as o. Loss L quantifies the extent of deviation between the output o values and the target values y.

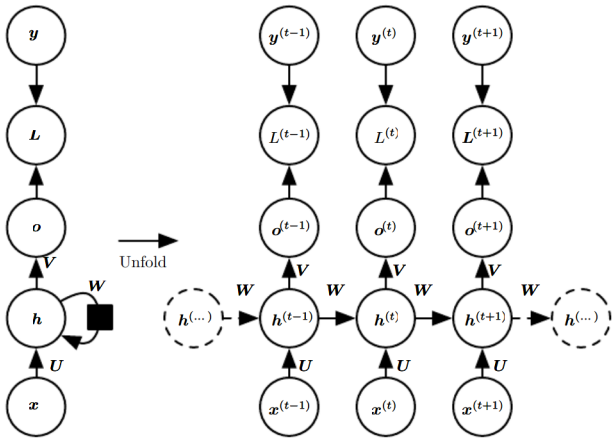

Figure 2.9: Recurrent Neural Network - Type 1

In the figure $\boldsymbol{x}^{(t)}$ represents the input, $\boldsymbol{h}^{(t)}$ represents the hidden layer activations, $\boldsymbol{o}^{(t)}$ represents the output, $\boldsymbol{y}^{(t)}$ represents the target and $\boldsymbol{L}^{(t)}$ represents the loss at each time step t.

We take the hyperbolic tangent activation function as the activation function for the hidden units and use it to create the forward propagation equations for this recurrent network. Also, we take the output to be discrete. The unnormalized log probabilities of

every potential target are the output. Next, the softmax operation is performed to acquire a vector $\hat{\boldsymbol{y}}$ of normalized probabilities over the output. $\boldsymbol{h}^{(0)}$ is the initial state. And the updated equations are the following:

$$\boldsymbol{a}^{(t)} = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \tag{2.13}$$

$$\boldsymbol{h}^{(t)} = tanh(\boldsymbol{a}^{(t)}) \tag{2.14}$$

$$\boldsymbol{o}^{(t)} = \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \tag{2.15}$$

$$\hat{\boldsymbol{y}}^{(t)} = softmax(\boldsymbol{o}^{(t)}) \tag{2.16}$$

Here, $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{W}$ represent the input-to-hidden, hidden-to-output and hidden-to-hidden connections respectively. And $\boldsymbol{b}$ and $\boldsymbol{c}$ are the bias vector along with the weight matrices.

Example - 2 : Recurrent networks are characterized by connections that allow information to flow from the output of one-time step to the hidden units of the next time step, resulting in an output being generated at each time step. (Figure 2.10).

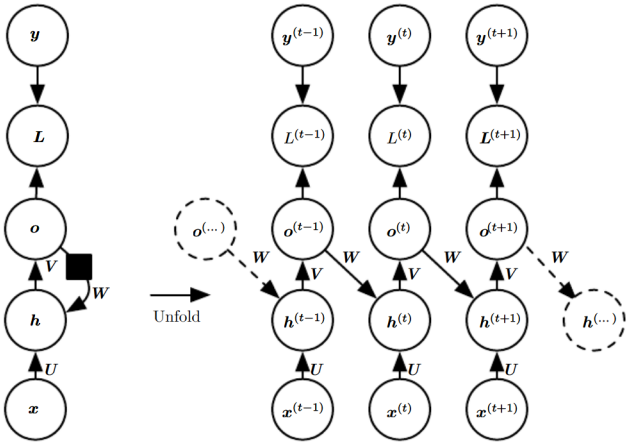

Figure 2.10: Recurrent Neural Network - Type 2

Training this recurrent network to input a particular output into $\boldsymbol{o}$ and only the information in $\boldsymbol{o}$ is allowed to be sent to the future. The previous $\boldsymbol{h}$ is connected to the current $\boldsymbol{h}$ through the output produced by the previous $\boldsymbol{h}$. Crucial historical context is absent

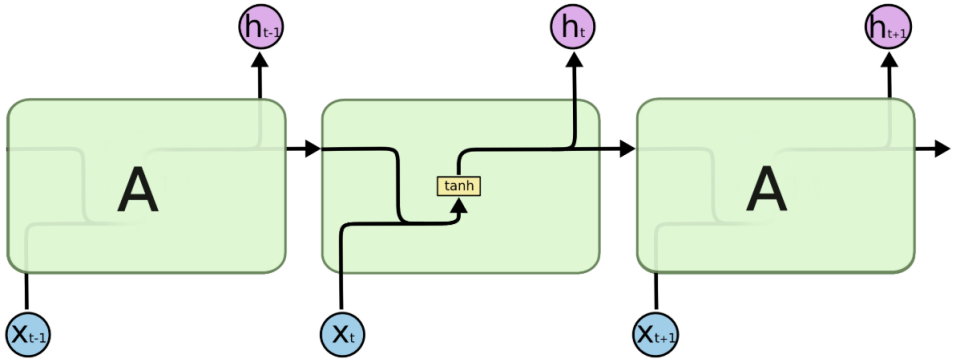

Figure 2.11: Standard RNN

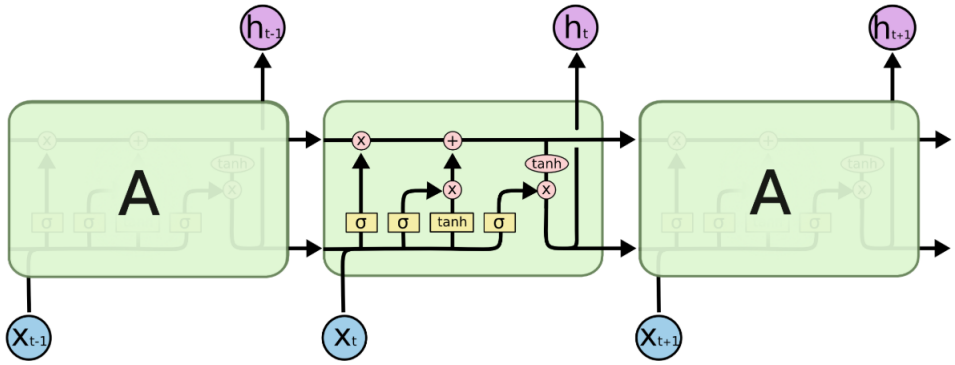

Figure 2.12: LSTM

unless $\boldsymbol{o}$ is very high-dimensional. It may be easier to train, but this makes the recurrent network less powerful [23].

2.2.1 LSTM

Generally, recurrent neural networks have a chain form of repeating modules of neural network [37]. In standard RNNs, the structure of this repeating module is straightforward, such as a single hyperbolic tangent layer (Figure 2.11).

However, in LSTM, each successive module consists of four layers that interact in a highly unique manner (Figure 2.12).

The initial stage of LSTM involves determining the specific information that should be

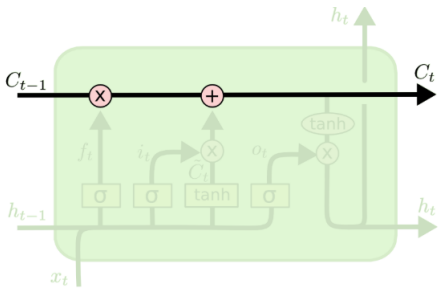

Figure 2.13: Cell of LSTM

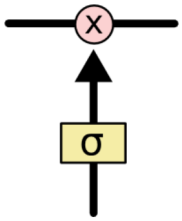

Figure 2.14: Structure of gate of LSTM

discarded from the cell state. The cell state is a crucial component of the LSTM model (Figure 2.13). The cell state propagates along the whole sequence, exhibiting little linear interactions. The information traverses it without undergoing any alteration. The gates of LSTM, a component of its architecture, regulate the manipulation of cell states by either introducing or removing information. Gates comprise a sigmoid layer and a point-by-point multiplication operation(Figure 2.14). The sigmoid layer determines the proportion of each component that should be allowed to pass through. The sigmoid layer produces an output that falls within the range of 0 and 1, where 0 means no information is going through and 1 means all information is going through. In the first step, the forget gate layer (Figure 2.15) decides what information needs to be removed from the cell state.

$$f_t^j = \sigma\left(b_f^j + \sum_k U_f^{j,k} x_t^k + \sum_k W_f^{j,k} h_{t-1}^k\right), \tag{2.17}$$

where b_f, U_f, and W_f represent the biases, input weights, and recurrent weight for the forget gates, respectively, x_t is the current input vector, and h_t is the current hidden layer

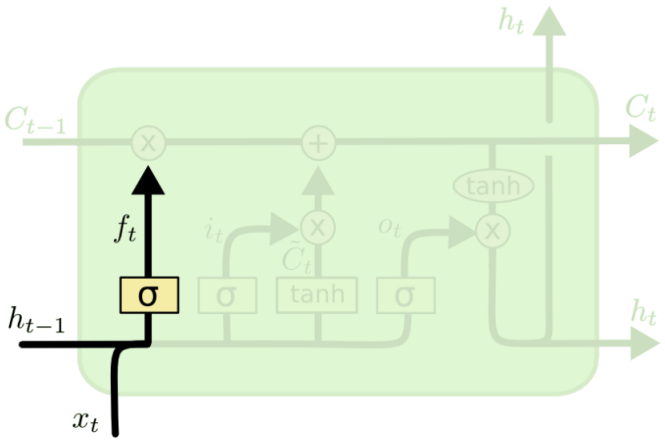

Figure 2.15: Forget gate of LSTM

vector that contains the outputs of all the LSTM cells.

During the subsequent phase, the LSTM algorithm determines the specific data that should be retained and stored within the cell state. This process is divided into two sections. The input gate layer determines which values should be updated (Figure 2.16), and the tanh layer creates a vector of new candidate values, $\widetilde{C}_t$, to update.

$$i_t^j = \sigma\left(b_i^j + \sum_k U_i^{j,k} x_t^k + \sum_k W_i^{j,k} h_{t-1}^k\right), \tag{2.18}$$

$$\widetilde{C}_t^j = tanh\left(b_C^j + \sum_k U_C^{j,k} x_t^k + \sum_k W_C^{j,k} h_{t-1}^k\right), \tag{2.19}$$

where b_i, U_i and W_i represent the biases, input weights, and recurrent weight for the input gates, respectively. b_C, U_C, and W_C are the biases, input weights, and recurrent weights for the hyperbolic tangent layer to create the candidate values.

Based on the provided information, the cell state will be updated (Figure 2.17).

$$C_t^j = \sigma\left(f_t^j * C_{t-1}^j + i_t^j * \widetilde{C}_t^j\right), \tag{2.20}$$

The output gate o_t^j that uses a sigmoid unit for gating stops the output h_t^j of the LSTM cell:

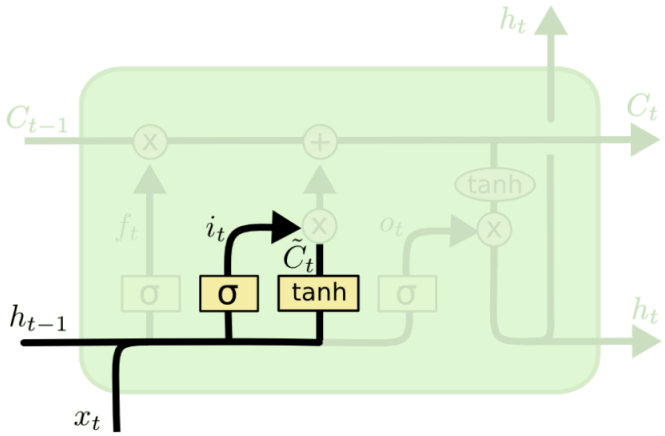

Figure 2.16: Input gate of LSTM

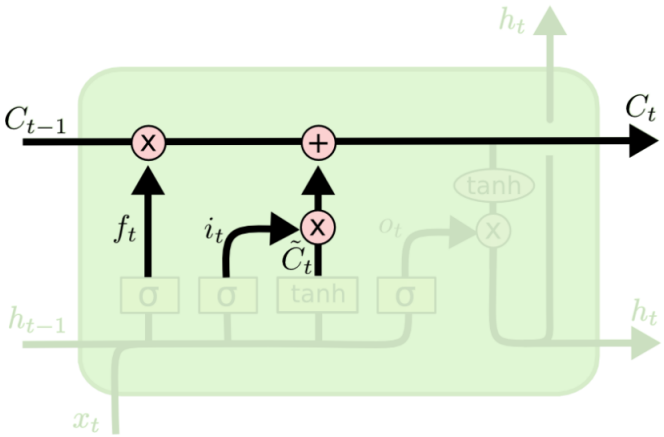

Figure 2.17: Updated state of LSTM

$$h_t^j = tanh\left(C_t^j\right) o_t^j \tag{2.21}$$

$$o_t^j = \sigma\left(b_o^j + \sum_k U_o^{j,k} x_t^j + \sum_k W_o^{j,k} h_{t-1}^k\right), \tag{2.22}$$

where b_o, U_o and W_o represent the biases, input weights, and recurrent weight for the output gates, respectively.

2.3 Attention

Traditional neural networks, such as those built of RNN or LSTM, sometimes struggle to effectively process and retain extensive amounts of information. These networks, commonly referred to as seq2seq models, are widely recognized for their proficiency in language modeling. This network's architecture has an encoder and decoder. The encoder converts sequential data into a context vector of a fixed length. A major drawback of fixed-length context vectors is that the network cannot recall long utterances. After processing a chain of information or a statement, the model may forget the beginning. We can fix the problem by incorporating into the model an attention mechanism.

An attention layer in deep learning is a method that enables models to selectively concentrate on particular segments of the input sequence while making predictions or generating outputs. Attention mechanisms, initially popularized by the Transformer model, have now become an essential element of many neural network topologies, particularly in tasks related to natural language processing.

Basic Operation An attention layer fundamentally calculates attention scores for every component in the input sequence. The scores ascertain the individual contribution of each constituent to the final result. The attention scores are usually calculated using a compatibility function that compares the current input element with the context or query given by the model.

Types of Attention The mechanism of attention can be categorized as follows:

1. **Self Attention** - Self-attention involves comparing the input sequence to itself in order to calculate attention scores. Every individual piece within the sequence has the ability to interact with and take into account other items within the same sequence, thereby encapsulating the interconnections and associations present in the input. This is also known as intra-attention [56].

2. **Global Attention** - During global attention or Inter-Attention, every element in the input sequence of the data is compared to a different context vector. This enables the model to choose to focus on various segments of the input sequence [30].

3. **Local Attention** - Local attention focuses the attention mechanism on only certain areas surrounding each patch in the input sequence, resulting in lower computing costs than global attention. This form of attention is primarily utilized in the network that deals with the task of processing images [30].

4. **Multi-Head Attention** - Multi-head attention integrates various attention mechanisms simultaneously, enabling the model to collectively focus on distinct segments of the input sequence. This improves the model's capacity to capture a wide range of patterns and connections [56].

Chapter 3

Related Work

3.1 Active region and Solar Flare Prediction

Solar flares, which involve high-energy charged particles and electromagnetic radiation, have the potential to rapidly impact radio communication, the accuracy of Global Positioning Systems (GPS), and the well-being of satellites and astronauts. Solar flares hold economic significance, so measures have been implemented to mitigate or avert their detrimental effects. The measures undertaken involve conducting research to comprehend the source, mechanisms, and ramifications of solar flares, as well as creating solar flare prediction technologies.

Studies on solar flares and their related energy release mechanisms indicate a strong association between them and magnetic activity [41]. Therefore, to understand and forecast solar flares, it is essential to analyze the magnetic field arrangement of the solar atmosphere. While it is not possible to directly map the magnetic field in the corona, it is possible to map the magnetic field in the photosphere. Previously, the majority of photospheric magnetic field data only included the magnetic field's line-of-sight component. The few datasets that included the whole vector components either had temporal gaps or only covered a portion of the solar disk. The Helioseismic and Magnetic Imager (HMI) on the Solar Dynamics Observatory (SDO) has been consistently monitoring and documenting the complete photospheric vector magnetic field since its launch in 2010 [47]. From the start of its operation in May 2010, HMI has mapped the vector magnetic field every 12 minutes 98.44% of the time [17].

The majority of existing studies on solar flare prediction utilize photospheric magnetic

field data to quantify active regions (ARs). When parameterizing, some researchers utilize energy, helicity, currents, and shear angles [14, 22, 26, 33], while some also use the magnetic field topology [48] and others use the integrated Lorentz force exerted by an AR [11]. The primary objective of these parameterizations is to establish a correlation between the behavior of the photospheric magnetic field and solar activity, often occurring in the chromosphere and the transition area of the solar corona.

The precise nature of the link between the magnetic fields in the photosphere and the corona during a solar flare is not fully understood. As a result, flare prediction has relied on statistical and machine learning methods to identify correlations rather than on theoretical models that could provide causal linkages [5].

The majority of machine learning methods have framed the problem as a binary classification task. In this task, an active region is classified as belonging to the positive class if it results in one or more flares within a specified period and as belonging to the negative class otherwise. Li et al. [28], Qahwaji et al. [42], Song et al. [51], Yu et al. [62], Yuan et al. [63], Ahmed et al. [1] used nonlinear Machine Learning (ML) algorithms to forecast solar flares. These studies were made using line-of-sight magnetic field data, solar radio flux, or metadata (e.g., McIntosh class, sunspot number) to characterize their features. Leka et al. [25] pioneered the utilization of vector magnetic field data for flare prediction by employing data from the Mees Solar Observatory Imaging Vector Magnetograph. This was done through the application of discriminant analysis, a linear classifier approach. Other prediction models used so far include the superposed epoch analysis [31], statistical analyses [4, 10], and support vector machine (SVM) [5, 28, 36].

The forecasting approaches often employ morphological or physical data derived from active regions. Ongoing research aims to identify the optimal and efficient predictive parameters for training statistical or machine learning models. The idea of sunspots classification as proposed by McIntosh et al. [32] led Lee et al. [24] to investigate the correlation between sunspot classifications and solar flares. Based on the current data, it is understood that the physical factors, such as the length of the neutral line [9], the gradient of the magnetic

field [7], the highly stressed longitudinal magnetic field [18], the distance between active regions and predicted active longitudes [18], and the Zernike moment of magnetograms [43], enhance the prediction powers when utilized in predictive models.

Moreover, the changes in the physical parameters in active regions were analyzed by Yu et al. [62], and Huang et al. [19]. In addition, Wheatland et al. [61] suggested that the past occurrences of solar flares have a crucial role in forecasting future flares. Although all these parameters were utilized, the data collected from active regions indicates that they were largely similar to one another. Consequently, they did not exhibit superior performance compared to other parameters in solar flare prediction models [3,26]. Obtaining the essential predictive characteristics from ARs has emerged as a bottleneck to enhancing the precision of solar flare prediction models.

There is a limited correlation between physical and morphological factors and solar flares. One discipline in machine learning has gained traction due to the rise in computational capabilities and the ability to extract useful predictive parameters from data with minimal human interaction. This discipline is known as deep learning [2, 16, 23]. Therefore, Nishizuka et al. [35] introduced the Deep Flare Net (DeFN), a sophisticated neural network, as a solution for predicting solar flares. This design employs manually computed features extracted from automatically identified sunspots. The identification of sunspots was achieved by establishing a threshold value of 140 G in accordance with the guidelines specified by [36]. They assessed their findings by employing the operational setup, which involved dividing the dataset into training and testing subsets based on chronological order. They achieved a better TSS (True skill statistic) as compared to other machine learning models like SVM, KNN (k-nearest neighbor), and ERT (extremely randomized trees).

Active regions undergo a transformation from their initial emergence to eventual disappearance. They vanish either via dying out or by shifting beyond the visible surface of the Sun. Changes in appearance are intricate spatiotemporal non-stationary processes since the combined distribution of neighboring pixel values undergoes transformations in both space and time [40]. These processes can be divided into deterministic and stochastic elements.

Research in neural networks has investigated spatiotemporal prediction in the framework of video classification and next-frame prediction. Modeling such features has been done using Convolutional Neural Networks (CNN) [21] and Recurrent Neural Networks (RNN) [55].

3.2 Next Frame Prediction

Next-frame prediction combines two deep learning techniques, namely *predictive learning* and *image generation*. In *predictive learning*, our objective is to simulate many potential outcomes of the future by utilizing data from the past. Recurrent networks are employed to collect patterns in sequence data and utilize them as a foundation for forecasting future outcomes. In *image generation*, the process involves extracting attributes from an existing dataset in order to generate novel images. Autoencoders and Generative Adversarial Networks (GANs) [13] are the prevailing networks utilized in image creation. The autoencoder consists of two components: the encoder and the decoder. The encoder processes the image and converts it into a latent variable, while the decoder uses the latent variable to reconstruct the original image. GANs consist of two components: the generator model and the discriminator model. The generator model produces authentic-looking images, while the discriminator categorizes both the generated images and the samples obtained from the generator model as either true or fake.

Ranzato et al. [44] suggested use a recurrent model to forecast frames within a limited domain of patch clusters. Subsequently, after the successful implementation of sequence-to-sequence mapping in language modeling through the utilization of Long Short-Term Memory (LSTM) networks, Srivastava et al. [54] adapted the approach to video prediction. Shi et al. [50] pioneered the incorporation of the convolution operator into recurrent state transition functions and put forth the concept of the Convolutional LSTM. With Convolutional LSTM as the base idea, Villegas et al. [57] and Patraucean et al. [39] created recurrent models to utilize optical flow-guided properties. Kalchbrenner et al. [20] proposed the concept of encoding the temporal, spatial, and chromatic structures of videos.

The Video Pixel Network (VPN) demonstrated proficient predictions but with substantial computational complexity. The PredNet model proposed by Lotter et al. [29] used Convolutional LSTM units in a top-down and bottom-up approach to create next-frame prediction of a video sequence. Wang et al. [58] implemented the zig-zag memory flow into the Convolutional LSTM, hence enhancing its capacity to represent short-term video dynamics.

The deep learning models mentioned above, which are designed for predicting future frames, can be categorized into two architectures based on how they predict future images or frames: sequence-to-one and sequence-to-sequence architecture. The first set processes a series of images, ranging from time step t to $t+h$, and generates a forecast of the subsequent time step, $t+h+1$, in the form of a frame or image. This architecture predominantly emphasizes the spatial arrangement of the input frames. In the second category of architectural designs, known as sequence-to-sequence architecture, the model receives temporal frames as input. Each frame at time step t is individually processed by the network, which then generates a prediction for the subsequent time step, $t+1$. This process is repeated until the desired frame is reached at a specific time interval. This architecture largely emphasizes temporal succession. Autoencoders are extensively utilized in both of these architectural types.

The recurrent model developed by Villegas et al. [57] is an example of sequence-to-one architecture that could predict up to 128 frames into the future. PredNet [29] is an example of a sequence-to-sequence architecture that can predict up to five frames into the future.

Chapter 4

Data

4.1 Data Description

We employ the data generated by the Helioseismic and Magnetic Imager (HMI) instrument, which is installed on NASA's Solar Dynamics Observatory (SDO). NASA's Solar Dynamics Observatory (SDO) is a satellite launched in 2010 as part of the Living With a Star (LWS) program. Its purpose is to monitor and document the activities of the Sun. The satellite is equipped with three instruments for monitoring the Sun. These are:

- The Atmospheric Imaging Assembly takes pictures of the Sun at the following wavelengths 94, 131, 171, 193, 211, 304, 335, 1600, 1700 and 4500 Å at a resolution of 4096×4096 pixels (approximately 1 arcsec).
- The Helioseismic and Magnetic Imager (HMI) which captures the oscillations and magnetic field of the Sun with a resolution of 4096×4096 pixels (pixel size of 0.5 arcsec) [47].
- The Extreme Ultraviolet Variability Experiment (EVE) measures the Sun's extreme ultraviolet spectral irradiance from 1 to 1050 Å.

The Joint Science Operations Center (JSOC) at Stanford University, Lockheed Martin Solar & Astrophysics Laboratory (LMSAL), and the Laboratory for Atmospheric and Space Physics (LASP) at the University of Colorado, Boulder provide access to images generated by the HMI, AIA, and EVE sensors.

By the end of 2022, the number of refereed scientific papers utilizing data from SDO exceeded 5,915. The significant scientific breakthrough can be credited to the spacecraft's and its instruments' dependability since its launch more than a decade ago, the consistent and high-quality data collected, the mission's transparent data policy, and the easily accessible tools provided by various science data centers and institutions for researchers and enthusiasts interested in studying the Sun [12].

4.1.1 HMI Instrument

The SDO/HMI is a space-based device created to replace the Michelson Doppler Imager on the Solar and Heliospheric Observatory. This device quantifies the oscillations and magnetic field of the Sun. Both measurements are obtained from the photosphere, yielding comprehensive full-disk photospheric vector magnetic field data. Data from HMI is processed at JSOC. Documentation, tools, and APIs required for accessing and manipulating the data held at JSOC are readily available to assist individuals in their research endeavors. APIs like Astropy and Sunpy, specifically developed for the Python programming environment, serve as valuable tools for researchers to manipulate data obtained from the HMI device.

Two camera configurations of the HMI instrument acquire complete views of the Sun's whole surface after the images have passed through a sequence of bandpass filters. Two cameras generate a 12-filtergram every 45 seconds. One camera records the six polarization states, while the other captures the right and left circular polarization for six distinct wavelengths.

Prior to reaching JSOC, the filtergrams are initially transmitted to a ground station located at White Sands, New Mexico. This takes place after the preprocessing of the data at the satellite level. Additional processing is conducted at White Sands prior to the transmission of the data to JSOC. Upon arrival to JSOC, the data is merged with the satellite's flight dynamics data to provide calibrated Level 1 filtergrams. These filtergrams yield four distinct data products: continuum filtergrams, dopplergrams, Line-of-Sight (LoS) data, and vector magnetograms. The LoS observables code is being used to calculate HMI

observables, including the Doppler velocity and LoS magnetic field intensity. Couvidat et al. [6] talk of an MDI-like algorithm for computing the LoS observables. The Line-of-Sight magnetograms have a cadence of 12 minutes.

In a magnetogram, the presence of grey areas signifies the absence or weakness of a magnetic field, whereas black and white areas indicate the presence of strong magnetic fields. The practice of using the colors grey, black, and white is typically employed by astrophysicists. The regions with the darkest shades correspond to areas of "south" magnetic polarity, which indicates an inward-directed magnetic field going toward the center of the Sun and having a negative value. Conversely, the whiter parts represent "north" magnetic polarity, indicating an outward-directed magnetic field moving towards us and having a positive value.

From the time of the launch of the SDO mission, the HMI instrument has not stopped working, producing approximately one terabyte of data per day. The times during which the flow of data has paused include the eclipse of the Sun by the Earth, poor weather conditions at the ground station, ground equipment failures, and maintenance.

4.2 Understanding the Data

To conduct the experiments, we compiled a dataset spanning from May 2010 to December 2018. This dataset was obtained from the preprocessed and publicly accessible dataset provided by Galvez et al [12]. The preprocessing steps involved removing incorrect frames, compensating for orbital changes and sensor deterioration, and reducing the image size from its original dimensions of 4096×4096 to 512×512. The date range was chosen because it covered almost one complete Solar Cycle which is Cycle 24, as shown in figure 4.1. We used the LoS magnetograms portion, that is, the data obtained from the Helioseismic Magnetic Imager (HMI).

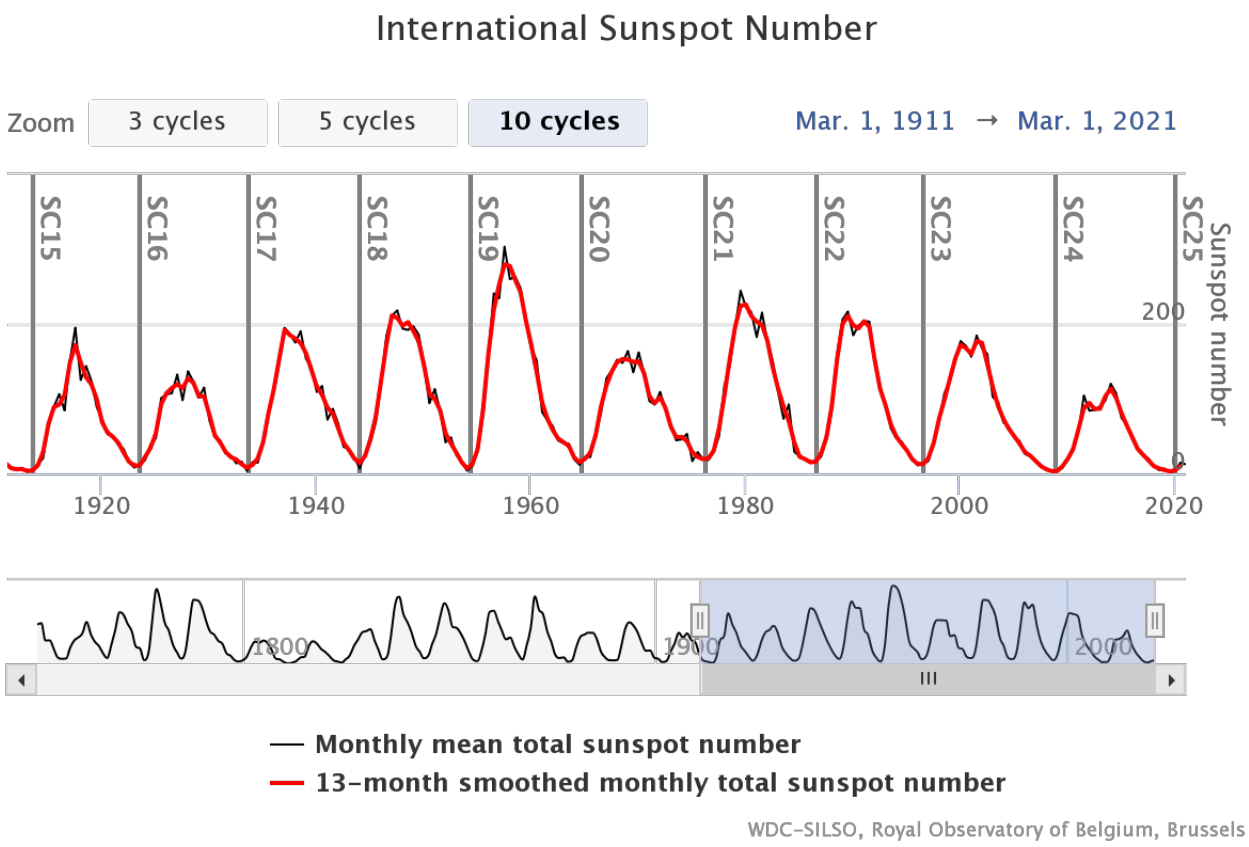

Figure 4.1: Graph of the number of sunspots for Solar Cycles ranging from March 1911 to March 2021

Chapter 5

Predicting Changes In a Fixed Area of The Sun

5.1 Data Processing

In this section, we perform the experiment to answer the question, "Could deep learning models designed for solving next-frame prediction be adapted to predict the changes in active regions and produce practical results better than conventional baseline approaches?" To accomplish this, regions of size 96×128 centered above the equator were cut out of the magnetograms as shown in figure 5.1.

Hence, while constructing a sequence of, say, 10 magnetograms, the absence of a single magnetogram renders the sequence incorrect. The number of sequences that can be generated from the dataset is determined by the specified number, as the sequence size is arbitrary. For our studies, we utilize a sequence with a length of 10. We employed two distinct scaling functions in conjunction with three varying time intervals during the data processing. Thus, for each time interval, two different scaling methods were employed, namely:

- Minimum-Maximum Normalization (Min-Max Norm) - Using the minimum and maximum values of the images to normalize them to $0 - 1$ range as shown in equation (5.1).
- Sigmoid Normalization (Sig-Norm)- Using the sigmoid function to normalize them to $0 - 1$ range as shown in equation (5.2).

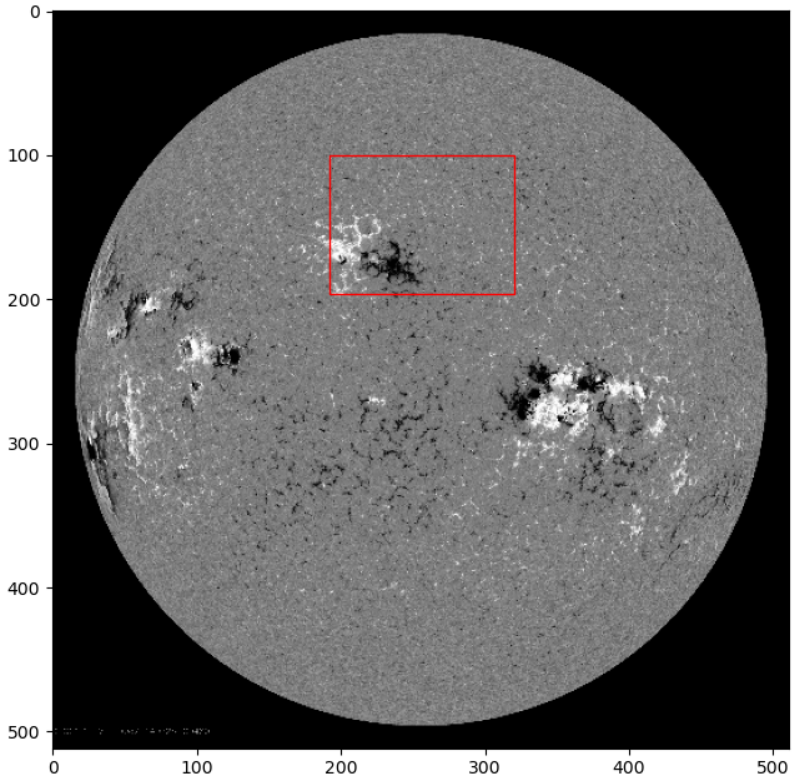

Figure 5.1: Region cutout shown with the red rectangle

The operations done on an image, X, in equation (5.1) and equation (5.2) are carried out element-wise. Since the images are of size 96×128, the matrices returned in the two equations are of the same size as the image. The value of k is set to be 0.03 in equation (5.2). And the maximum pixel value of the images is known to be as high as 5000 [47]; therefore, b is set to be 5000 whilst a, representing the minimum value, is set to be -5000.

$$\text{min\_max} = \frac{\mathrm{X} - \mathrm{a}}{\mathrm{b} - \mathrm{a}}. \tag{5.1}$$

$$\text{sig\_norm} = \frac{1}{1 + \mathrm{e}^{-\mathrm{X}*\mathrm{k}}} \tag{5.2}$$

We applied different values of k in equation 5.2 to see how the data distribution was affected. Two magnetograms were selected at random, one captured during the period of solar minimum and the other at the period of solar maximum. We selected two values of k, namely 0.01 and 0.001, with a step size of 0.01 and 0.001, respectively. To determine the appropriate value for k, we analyze the graph depicted in the figures 5.2 and 5.3 by focusing more on the graph from the solar minimum. In this case, we selected the value of 0.03 due to its favorable bell-shaped characteristic and minimal impact on the extreme values (0 and 1). However, when considering values ranging from 0.001 to 0.007, there was no noticeable difference in the distribution of pixel values, as depicted in the figures 5.4 and 5.5.

Also, in processing the data, we used the following time intervals:

- input = 1 hour; output = 1 hour
- input = 3 hour; output = 3 hour
- input = 6 hour; output = 6 hour

To ensure that the sequence remains 10 frames long, we raised the frequency of the dataset to 36 minutes and 72 minutes for the time intervals of *3-hour input; 3-hour output* and *6-hour input; 6-hour output* correspondingly. For the *1-hour input; 1-hour output*, we

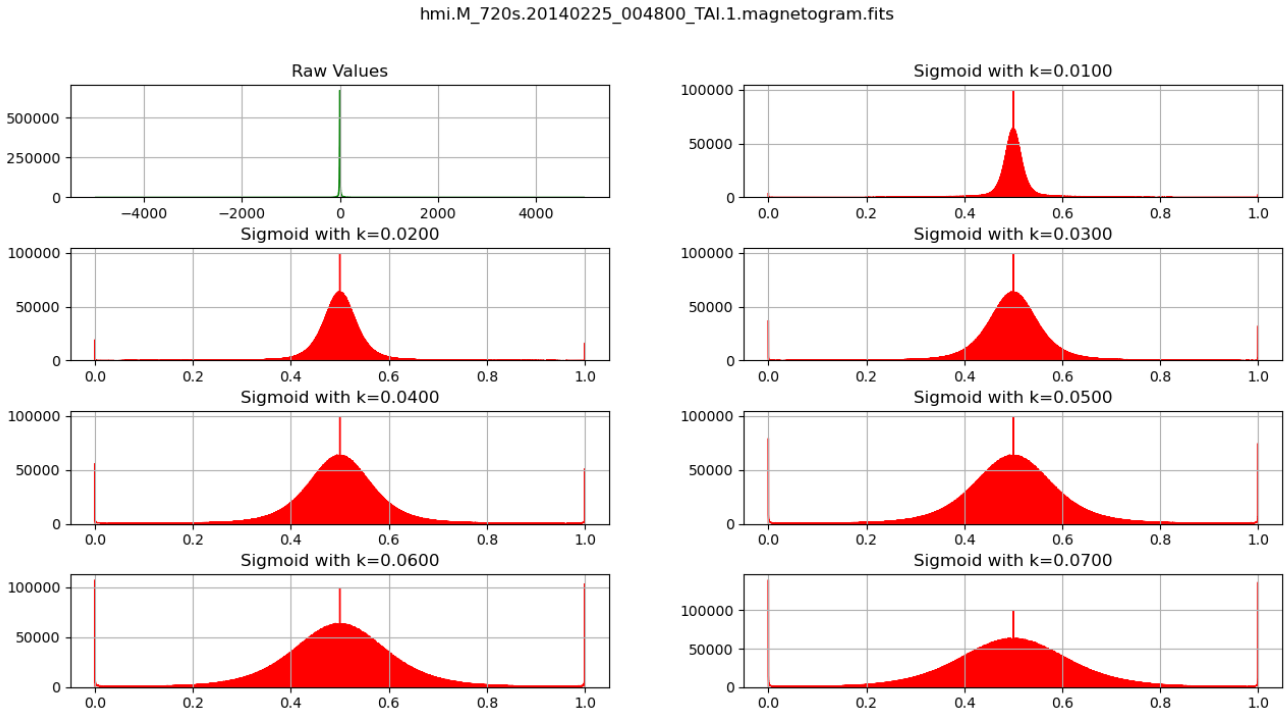

Figure 5.2: Plot of distribution of pixel values of the image from solar maximum after transformation using k ranging from $0.01 - 0.07$.

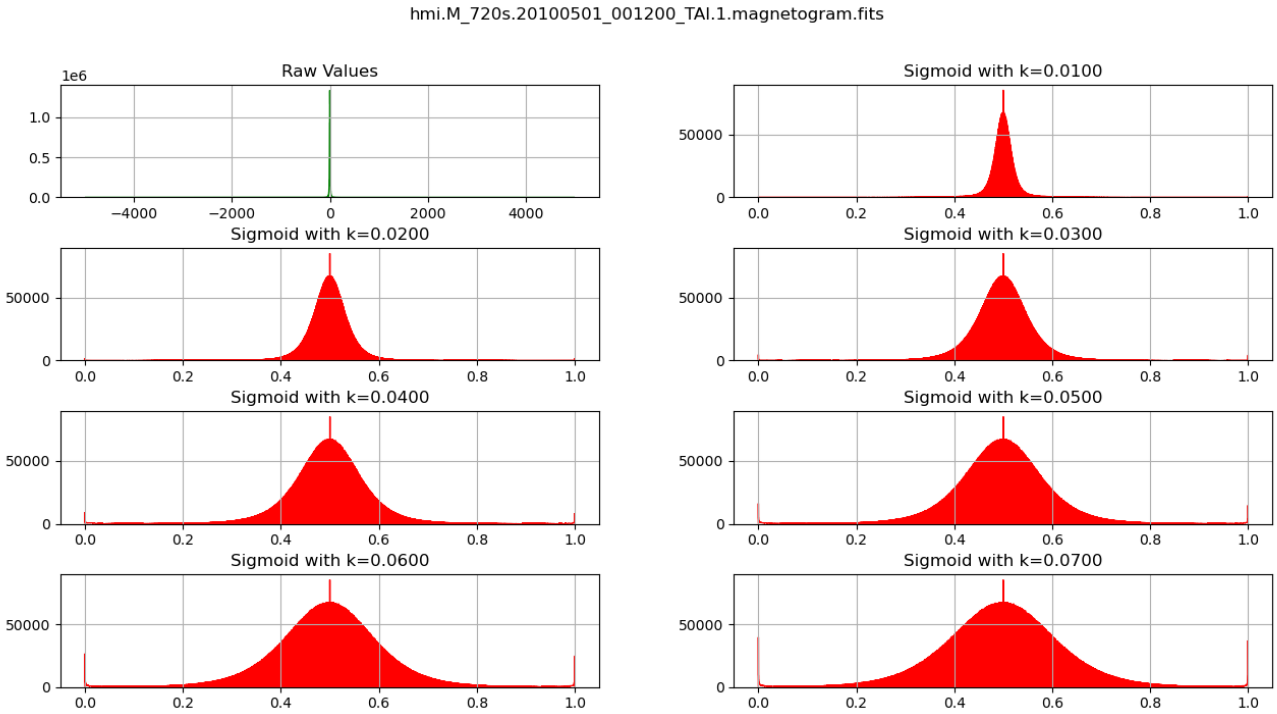

Figure 5.3: Plot of distribution of pixel values of the image from solar minimum after transformation using k ranging from $0.01 - 0.07$.

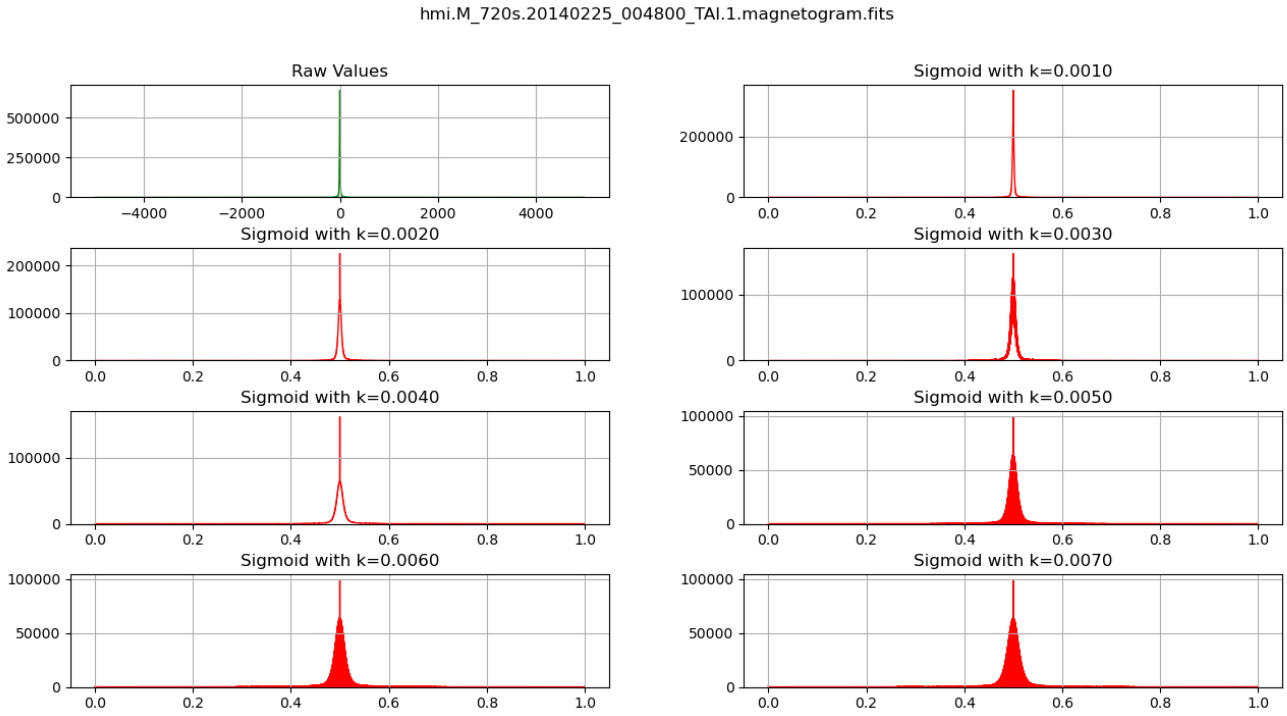

Figure 5.4: Plot of distribution of pixel values of the image from solar maximum after transformation using k ranging from $0.001 - 0.007$.

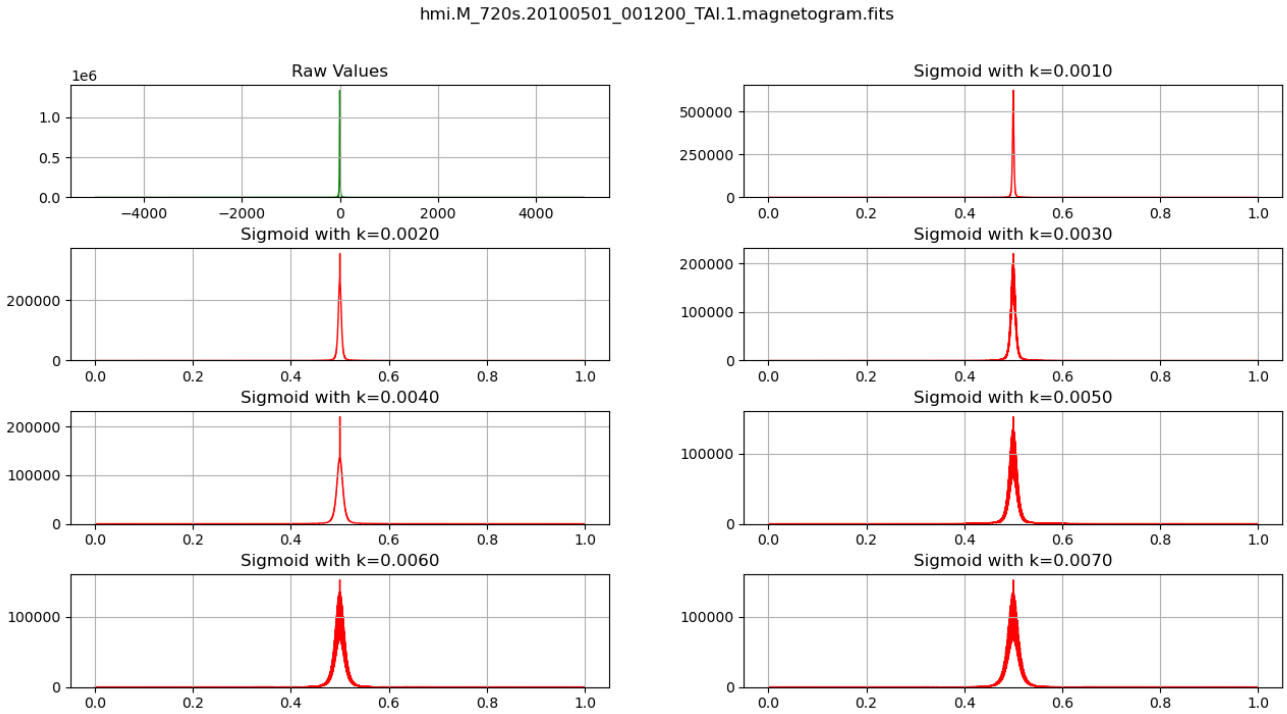

Figure 5.5: Plot of distribution of pixel values of the image from solar minimum after transformation using k ranging from $0.001 - 0.007$.

maintained the 12-minute cadence utilized in the original dataset. The dataset obtained according to the aforementioned parameters yielded six distinct datasets for conducting our experiment. Table 5.1 shows the number of sequences generated for each time interval.

Table 5.1: Number of Sequences

Input/Output	**Training**	**Testing**	**Validation**
1 Hour / 1 Hour	161880	76457	17040
3 Hour / 3 Hour	39330	18442	4140
6 Hour / 6 Hour	9120	4050	922

Each dataset was subdivided into training, testing, and validation using contiguous sampling figures as illustrated in figure 5.6. For example, for 1000 sequences generated, the first 20 sequences are placed in the training portion, the next 5 in the validation portion, and the next 10 after that in the testing portion, and the process repeats until all the 1000 sequences are partitioned. This was done to capture the changes on the Sun as it moves from solar minimum to solar maximum in each partition and also prevent information leakages from the testing portion into the training portion.

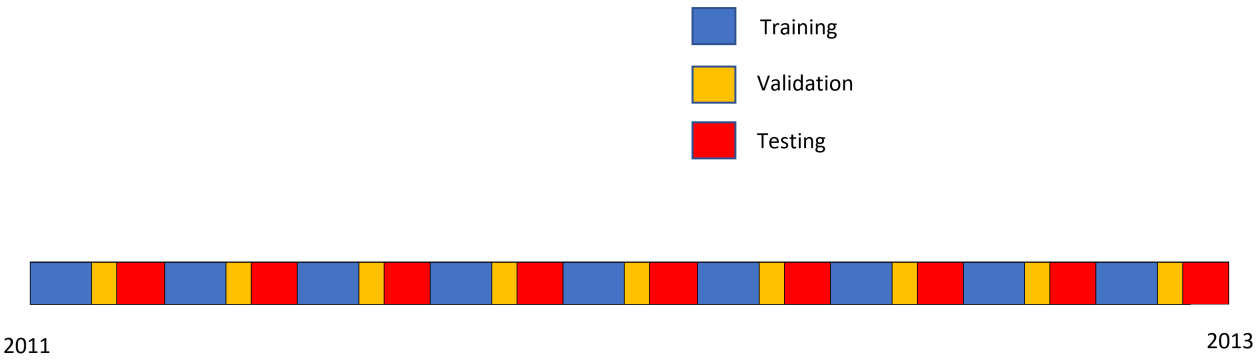

Figure 5.6: Contiguous Sampling from 2011 to 2013

5.2 Models

5.2.1 Memory In Memory Network (MIM)

The Memory In Memory (MIM) Network is a type of RNN architecture that is also considered a sequence-to-sequence architecture. It was developed as a modification of PredRNN [58] to address issues such as the saturation of forget gates.

In order to address the issue of saturation, memory units are organized in a cascaded manner, which also facilitates the representation of non-stationary changes. The MIM network consists of two primary components: the non-stationary module (MIM-N) and the stationary module (MIM-S), as depicted in figures 5.7 and 5.8 accordingly. The combination of these two components creates the MIM block. The MIM-N module takes two consecutive hidden states (H_t^{l-1} and H_{t-1}^{l-1}) as inputs and calculates the difference between them to capture the non-stationary variations. This results in the production of D_t^l as depicted in figure 5.7. In the MIM-S module, the output D_t^l from the MIM-N module is combined with the previous outer temporal memory C_{t-1}^l to capture the approximation stationary changes in spatiotemporal sequences.

The horizontally-transited memory is denoted by S and N, while the differential attributes are denoted by D, and the memory cells are denoted by C. The subsequent equations illustrate the computations occurring in MIM-N and MIM-S:

MIN-N

$$g_t = tanh(W_{xg} * (H_t^{l-1} - H_{t-1}^{l-1}) + W_{ng} * N_l^{t-1} + b_g)$$
$$i_t = \sigma(W_{xi} * (H_t^{l-1} - H_{t-1}^{l-1}) + W_{ni} * N_{t-1}^l + b_i)$$
$$f_t = \sigma(W_{xf} * (H_t^{l-1} - H_{t-1}^{l-1}) + W_{nf} * N_{t-1}^l + b_f))$$
$$N_t^l = f_t \bigodot N_{t-1}^l + i_t \bigodot g_t$$
$$o_t = \sigma(W_{xo} * (H_t^{l-1} - H_{t-1}^{l-1}) + W_{no} * N_t^l + b_o)$$
$$D_t^l = \text{MIM-N}(H_t^{l-1}, H_{t-1}^{l-1}, N_{t-1}^l) = o_t \bigodot tanh(N_t^l),$$

MIM-S

$$g_t = tanh(W_{dg} * D_t^l + W_{cg} * C_{t-1}^l + b_g)$$
$$i_t = \sigma(W_{di} * D_t^l + W_{ci} * C_{t-1}^l + b_i)$$
$$f_t = \sigma(W_{df} * D_t^l + W_{cf} * C_{t-1}^l + b_f)$$
$$S_l^t = f_t \bigodot S_{t-1}^l + i_t \bigodot g_t$$
$$o_t = \sigma(W_{do} * D_t^l + W_{co} * C_{t-1}^l + W_{so} * S_t^l + b_o)$$
$$T_t^l = \text{MIM-S}(D_t^l, C_{t-1}^l, S_{t-1}^l) = o_t \bigodot tanh(S_t^l),$$

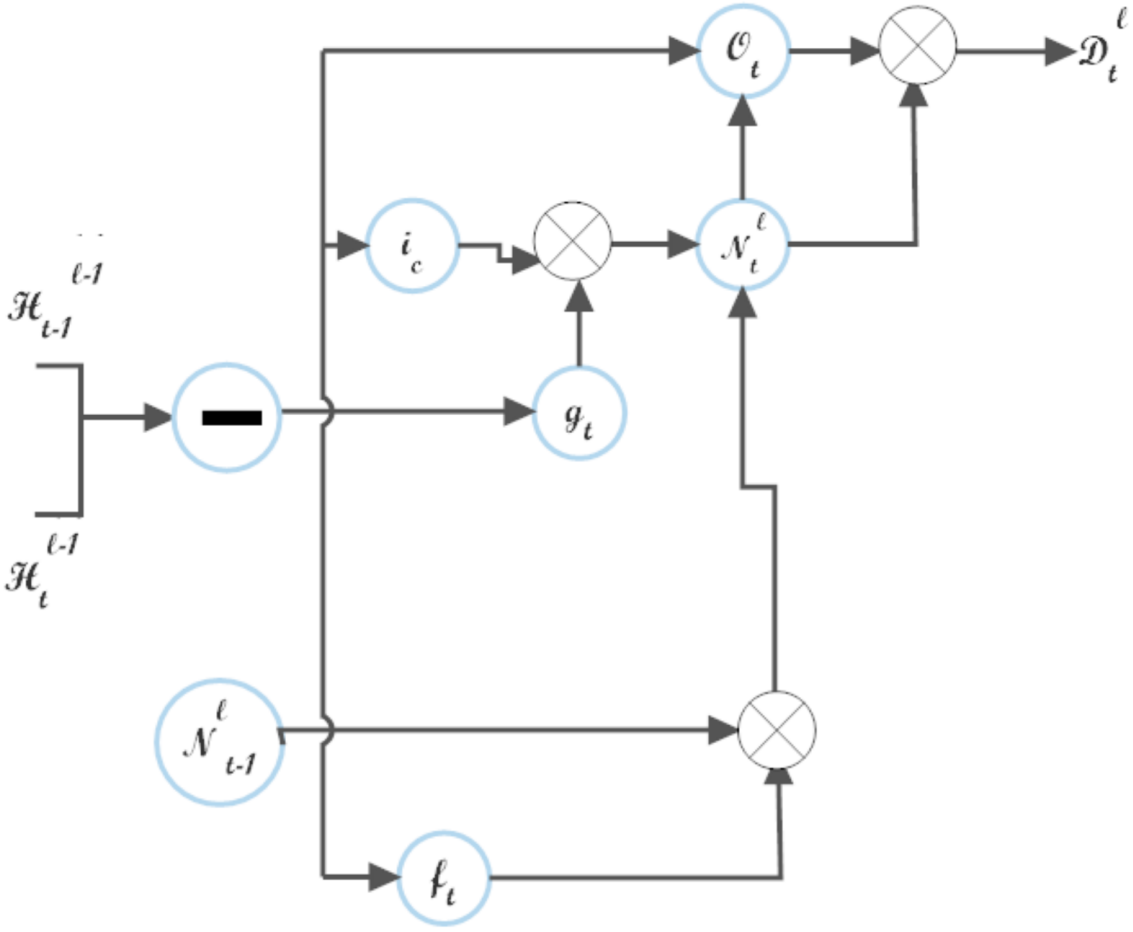

Figure 5.7: Schematic of MIM-N [59]

The MIM network is formed by vertically arranging numerous MIM blocks to generate a model for the spatiotemporal process and enhance the precision of predicting future frames.

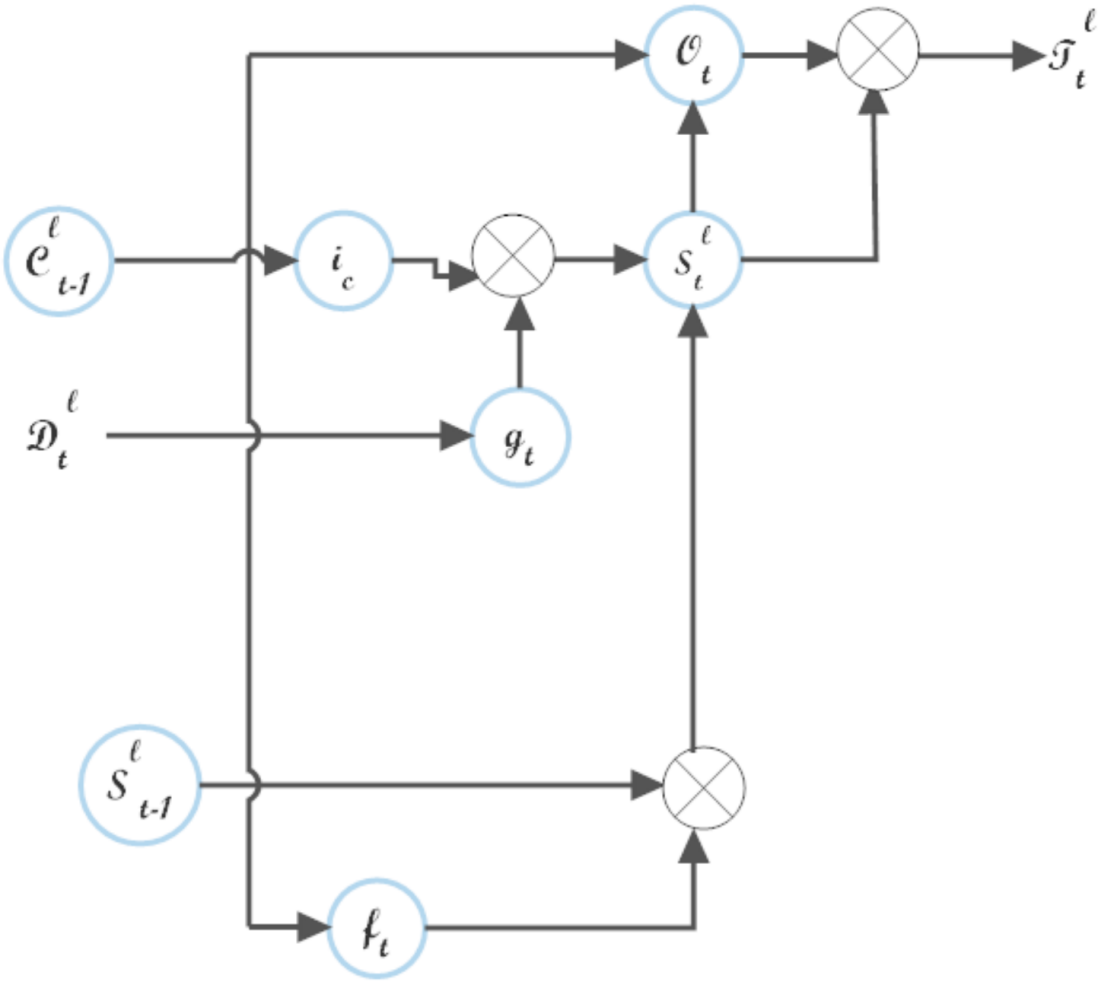

Figure 5.8: Schematic of MIM-S [59]

The network configuration of the MIM blocks is illustrated in Figure 5.9. The preceding hidden states H_{t-1}^{l-1} and H_t^{l-1} are transmitted to the MIM block at a non-initial timestamp $t \neq 1$ and layer $l \neq 1$ in order to provide the differentiated features for subsequent processing. These states are shown by green arrows in figure 5.9. The spatiotemporal LSTM (ST-LSTM) is employed for generating the hidden representations in the initial layer, as there is no preceding layer. The non-stationarity of the sequence becomes more apparent when temporally adjacent hidden representations are compared. This is because the spatiotemporal dynamics in local areas are encoded into the hidden states through the bottom ST-LSTM layer [59].

The MIM-N and MIM-S extract both stationary and non-stationary features, which are then transmitted through the yellow arrows in figure 5.9. The MIM network generates a single frame at a specific timestamp. The green arrows in figure 5.9 depict the diagonal state transition patterns of hidden representations for differential modeling. The black arrows in figure 5.9 depict the zigzag state transition patterns of the memory module. The input to the network, as seen in figure 5.9, can be either the ground truth frame from the input sequence or the frame generated at the prior timestamp [59].

5.2.2 Baseline

Due to the novelty of our investigation, we were unable to make comparisons with existing research. Consequently, we employed a straightforward reference point to evaluate our effort. Considering the intricate spatiotemporal patterns on the Sun's surface, including its rotation and magnetic field fluctuations, our approach is to use the previous input frame as the predicted image for the output sequence. In short, we employed a no-change baseline approach, wherein the predicted images remain identical to the previous input frame.

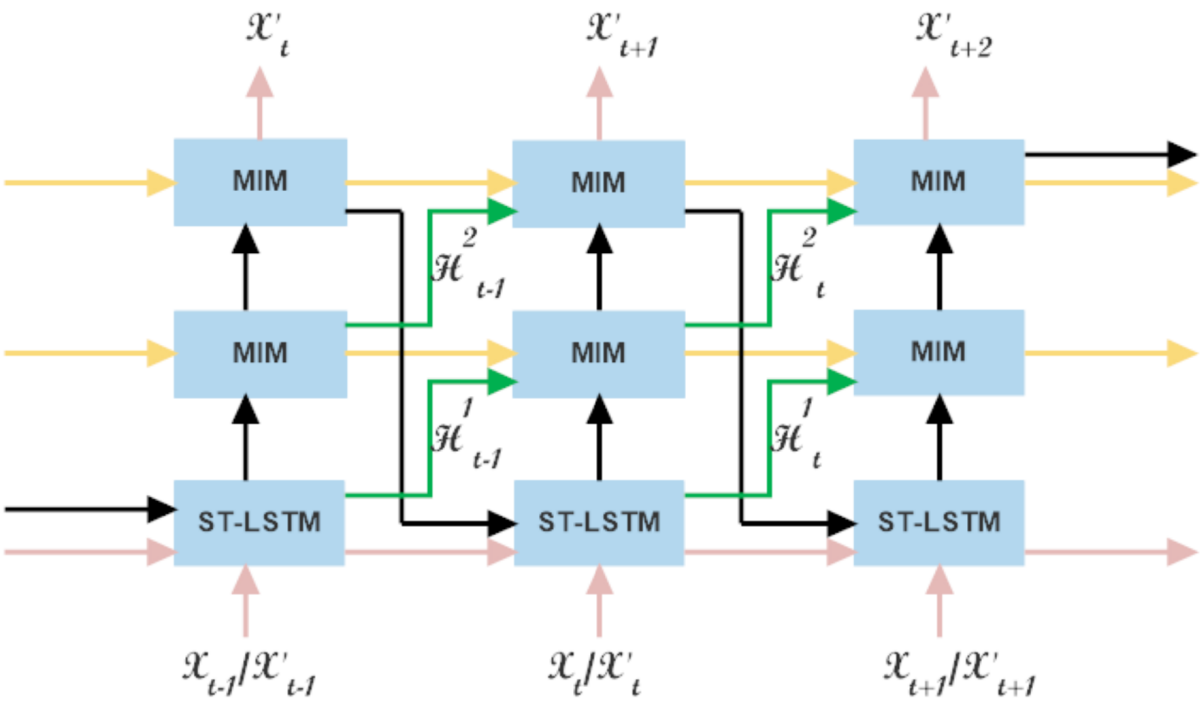

Figure 5.9: A MIM network with two MIMs and one ST-LSTM. [59]

5.2.3 Regression

In addition to the baseline, we employ the regression model to predict the spatial and temporal variations of the Sun. In order to accomplish this, we select the first rows of the sequences from the training set and subsequently train a regression model to establish a mapping between the inputs (the first rows of each frame in the input sequence) and the output (the first rows of each frame in the output sequence). Following this training, we apply it to the first row of the input sequence of the test dataset. This process is repeated for each row until the final row, resulting in a total of 96 rows.

5.3 Results

Initially, we implemented the no-change baseline on the testing part of each dataset. Then, we trained both the regression model and the Memory In Memory (MIM) Network using the training subset of each dataset. Finally, we compared the performance of both models against each other. For evaluation purposes, we utilize the structural similarity index measure (SSIM) [60] and the mean square error (MSE), averaging over their respective test sets. The Mean Squared Error (MSE) measures the dissimilarity between the two images, while the Structural Similarity Index (SSIM) offers a quantitative assessment of the similarity between the two images. Consequently, a lower Mean Squared Error (MSE) and a higher Structural Similarity Index (SSIM) imply a better prediction. The time interval (cadence) between t = 1 and t = 2, as indicated in the results, is 12 minutes. Consequently, when t is equal to 3 and 6, there is a time interval of 36 minutes, and this pattern continues.

By modifying the batch size to 1, the total sequence length to 10, and the input length to 5, while keeping all other parameters unchanged against those used by Wang et al. [59], including the utilization of the L1 loss function, we conducted training using the MIM network. The training process consisted of 170000, 160000, and 80000 iterations for 1-hour, 3-hour, and six-hour time intervals respectively. This training was performed on a

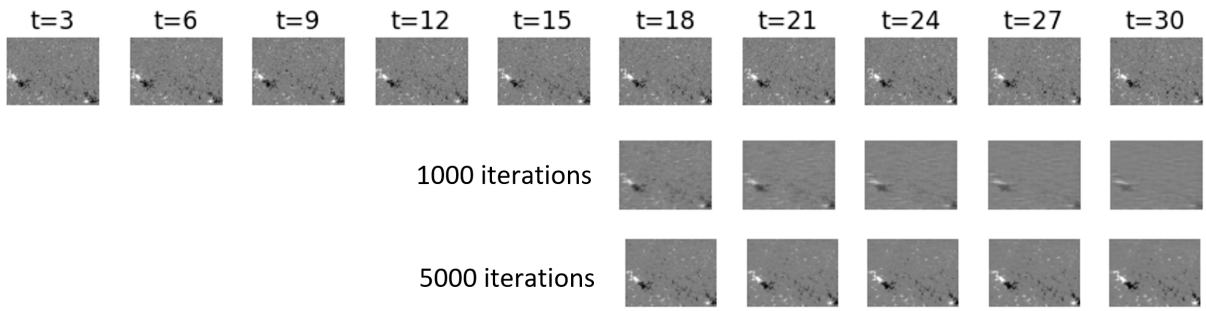

Figure 5.10: Sequence tested at different iteration step

single NVIDIA GeForce RTX 2080 Ti GPU. The entire training process took approximately 3 to 8 days to complete. The variation in the number of iterations arises from the difference in the number of sequences accessible for each time period, with each iteration signifying the traversal of a single sequence across the network. Figure 5.10 shows validation of the network after 1000 and 5000 iterations during training of the network using the sig-norm 3-hour time interval dataset.

Also, the figures 5.10, 5.11, 5.12 and 5.13 shown have been color-coded with 1 being white and 0 being black to help visualize the changes in images both in the ground truth and the predicted one. The ground truths are presented in the first row, whilst the predicted ones are shown in the second and/or third row.

Upon visual inspection in figure 5.12 for example, it can be seen that the sig-norm images show good detail on what is happening and produced a good prediction visually, though it performed poorly against the min-max Norm in all reported metrics.

Deep neural networks perform well when images are scaled to the range of 0 to 1. To normalize the values in the images, which range from -5000 to 5000, we utilize the min-max normalization technique. However, the pixel values are not uniformly distributed as the majority of values fall within the range of $0.4-0.6$. This gives rise to the issue of similar values, particularly during the training phase when values are approximated. Consequently, the network made a prediction that there was no activity in the image, resulting in the prediction of pixel values that are close to the midpoint (0.5). In figure 5.11, which is the

min-max norm for a time interval of 6 hours, this problem becomes more evident as the network fails to make a good prediction. In this figure 5.11, gray images (almost no visual changes) were predicted, which shows that the network predicted the images to have pixel values within a very small range. This led to the application of sig-norm (equation(5.2)) to help solve this issue. Based on the results as seen in Figure 5.12, that is, sig-norm for the 6-hour time interval, sig-norm helps solve this issue in normalization.

For the results obtained for the MIM network in tables 5.2, 5.3, and 5.4, the SSIM for the min-max norm is close to 1 and shows slight improvement over the baseline. However, in their corresponding MSEs, there is over 43% improvement.

Regarding the regression model used, it demonstrates superior performance compared to the baseline model in both Mean Squared Error (MSE) and Structural Similarity Index (SSIM). However, when compared to the MIM model, it only outperforms in terms of MSE. The reason for the strong performance in MSE can be related to the regression model's objective of minimizing the mean square error, while the MIM aims to minimize the L1 error. A marginal disparity in performance is observed in table 5.4, where the regression model outperformed MIM in SSIM by a difference of 0.0002. Figure 5.13 displays a prediction generated using the trained regression model. This image depicts the identical test sample as seen in figure 5.12. Figure 5.12 provides greater detail, particularly in the higher regions and the sunspot area located at the bottom right of the predicted images.

Since the MIM network was created and trained for perceptual tasks, it shows this characteristic in the SSIM metric, where it did better than the baseline and the regression model in almost all the results.

In figures 5.11 and 5.12 for the 6-hour interval, the test sample captured a sunspot. From the ground truth row in all the figures, it can be seen that the sunspot is moving towards the right (rotation of the Sun). This non-stationarity is captured by the MIM network, and it is seen in the row showing the predicted images.

In the SSIM and MSE metrics under sig-norm in tables 5.2, 5.3 and 5.4, the MIM network and the regression model did well not to deviate by higher margins as the time

intervals were increased.

Table 5.2: One Hour Input to Predict the Next One Hour

Model		**Min-Max Norm**		**Sig Norm**	
		MSE	***SSIM***	***MSE***	***SSIM***
Baseline		4.52e-05	0.9927	0.0408	0.2845
Reg Row-Row		**2.23e-06**	0.9979	**3.18e-03**	0.4930
MIM		9.62e-06	**0.9982**	0.0145	**0.5778**
% Improvement	Reg Row-Row	95.1	0.52	92.2	73.3
Against Baseline	MIM	78.7	0.55	64.5	103.1

Table 5.3: Three Hour Input to Predict the Next Three Hour

Model		**Min-Max Norm**		**Sig Norm**	
		MSE	***SSIM***	***MSE***	***SSIM***
Baseline		6.09e-05	0.9914	0.0502	0.1404
Reg Row-Row		**3.44e-06**	0.9969	**3.97e-03**	0.3898
MIM		1.51e-05	**0.9973**	0.0182	**0.4478**
% Improvement	Reg Row-Row	94.4	0.55	92.1	177.6
Against Baseline	MIM	75.2	0.60	63.7	218.9

Table 5.4: Six Hour Input to Predict the Next Six Hour

Model		**Min-Max Norm**		**Sig Norm**	
		MSE	***SSIM***	***MSE***	***SSIM***
Baseline		3.13e-05	0.9949	0.0427	0.1274
Reg Row-Row		**2.61e-06**	**0.9975**	**4.01e-03**	0.3135
MIM		1.76e-05	0.9973	0.0184	**0.3668**
% Improvement	Reg Row-Row	91.7	0.26	90.6	146.1
Against Baseline	MIM	43.8	0.24	56.9	187.9

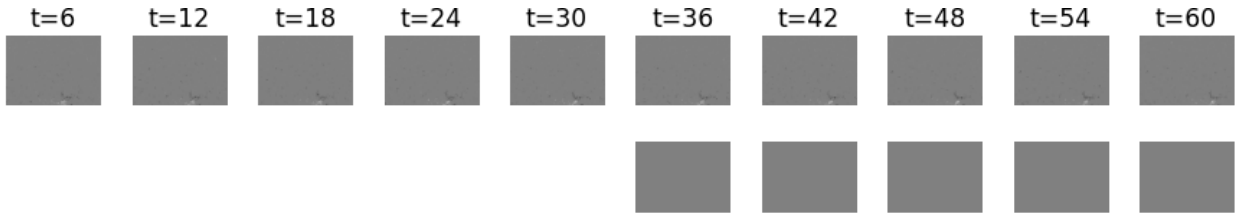

Figure 5.11: MIM: Min-Max Norm - Input: 6 Hour Output: 6 Hour

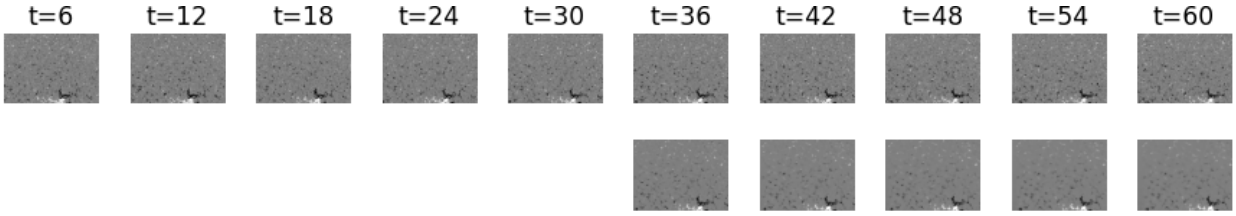

Figure 5.12: MIM: Sig Norm - Input: 6 Hour Output: 6 Hour

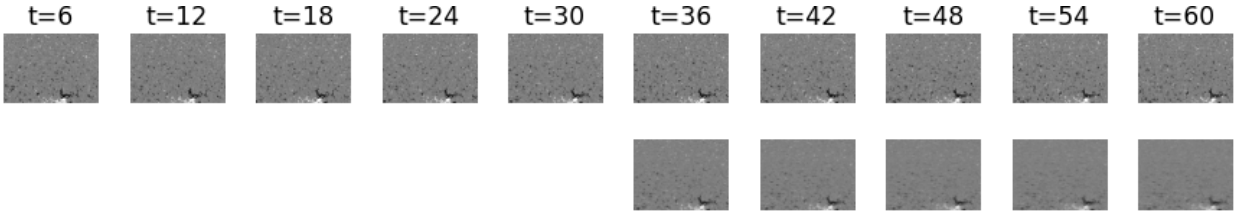

Figure 5.13: Regression: Sig Norm - Input: 6 Hour Output: 6 Hour

Chapter 6

Predicting Changes In Active Regions In Tracked Areas of the Sun

6.1 Data Processing

To capture the changes taking place in the active regions and prevent the network from focusing only on the sun's rotation, as seen from the results in Chapter 5, we leverage the knowledge of the 3D viewing geometry and the Sun's rotation to preprocess the data to remove the Sun's rotation. To create the sequence, we start by tracking regions as they appear on the visible sun disk from the far left to the right. These regions are above and below the equator, spanning 40° in longitude and 30° in latitude (0° to 30° both north and south). A sequence to be considered must have at least one sunspot in the region as it moves to the middle of the sun. These regions are warped into a rectangular area. Looking at these regions forming the sequence, one will see the active regions or sunspots not moving to the right as the Sun rotates; rather, it will be seen that the active regions in these images are shrinking or expanding. This is similar to following the region with a camera right on top of it. The criteria used in generating and selecting the sequences are as follows:

1. A sequence spans 36 hours, the time from the first input frame/image to the target frame.

2. The temporal distance between subsequent images is 6 hours.

3. For a sequence to be considered, no intermediate frames must be missing after sam-

pling. If there is, we go into the future or past 24 minutes to find the closest frame to replace the missing one.

4. The region being tracked spans 40° in longitude and 30° in latitude (0° to 30° both north and south).

5. The frame before the target frame (the last input frame) must have an active region or part of an active region to be evaluated as a valid sequence. This is the frame located in the middle of the sun.

6. The time difference from one sequence to the next subsequent sequence to be generated is 1 hour if the previous sequence is valid and 12 minutes if it is not.

With this criteria, each sequence had six images. From the northern part of the sun above the equator, 28,506 sequences were generated, and for the south, 22,639 were generated (approximately . 26% (5,867) more sequences in the north than in the south). In total, there are 51,145 sequences (6.1G). Sequences from the south and the northern data, partitioned by years, were used for training. The partitioning was done using this criterion: 2010–2012 as one group, 2013–2015 as another, and 2016–2018 as the last group. Using k-fold cross-validation, one group was left out to be used for testing. The results presented are the average of the three runs across the dataset while keeping the data generated from the south constant throughout the training. Logarithmic scaling is applied to the images using the following formula, and *sign* returns 1 or -1 for each pixel value, whether it is positive or negative, respectively:

$$\begin{aligned} \text{Log\_scale} &= sign(x) * \log_e \left(1 + \frac{|x|}{\epsilon}\right) \\ where \quad \epsilon &= 10 \end{aligned} \tag{6.1}$$

The Log scale has been applied to the pixel intensity values, ranging from -5000 to 5000, as shown in 6.1 and 6.2 for different values of ϵ. From the graph, $\epsilon = 1e1 or 10$ provides a good non-linear characteristic where it starts to saturate after 2000 and -2000 marks while

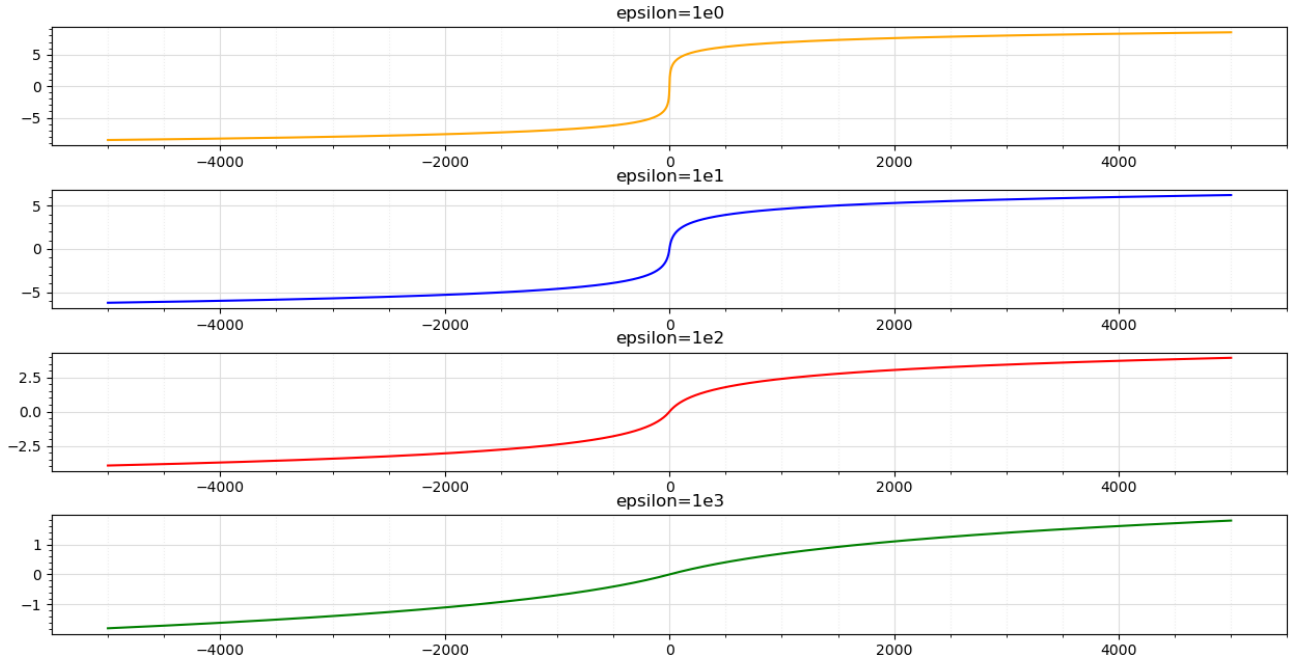

Figure 6.1: log scale with ϵ with positive exponents

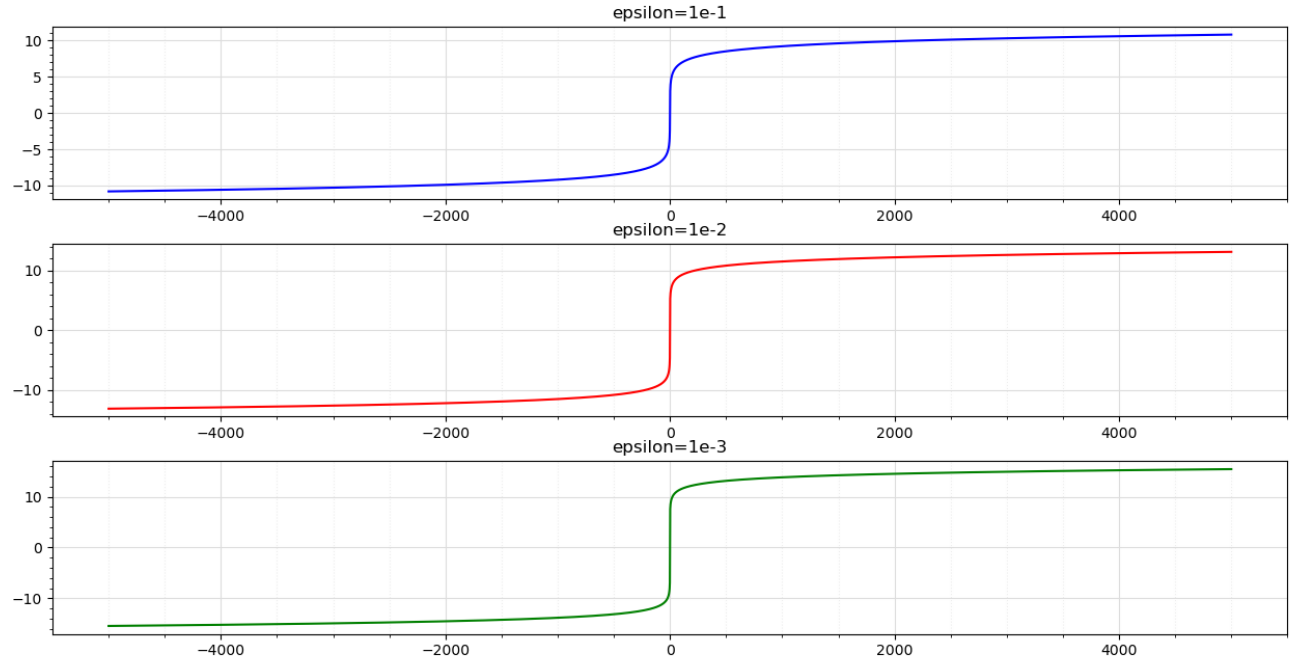

Figure 6.2: log scale with ϵ with negative exponents

also providing a reasonable range. It also offers a sharp rise or a reasonable deviation from the zero mark as pixel values move in the negative or positive direction.

6.2 Model

6.2.1 Baseline

For the baseline, we use the copy-last-frame methodology. Here the last input frame, the fifth frame of the input sequence, is used as the predicted frame.

6.2.2 Convolutional Neural Network (CNN)

Building The CNN

To build the CNN model to predict the target image, we begin by stacking convolutional layers on top of each other to generate a deep convolutional network (Deep CNN). Each layer has a different number of 3×3 convolutional filters, followed by the ReLU activation function. The padding parameter is set to 'same' with a constant stride of 1. The final or output layer has the same 3×3 filter, but the activation function is linear, with stride being one and the number of filters set to 1. With padding set to 'same' and stride as 1, during convolution with the filters, the input to the layer is padded with zeros evenly to the left/right or up/down to maintain the size after convolution. Therefore, the size of the input to the CNN is the same as the output. The architectural diagram for the Deep CNN is shown in figure 6.3.

Adding Skip Connections and dropout

Skip connections, also known as residual connections, allow the flow of data from one layer of the network to the latter layers. This also allows the flow of gradients during backpropagation to bypass some intermediate layers and flow to earlier layers. This helps to solve problems such as vanishing gradients or optimization difficulties associated with

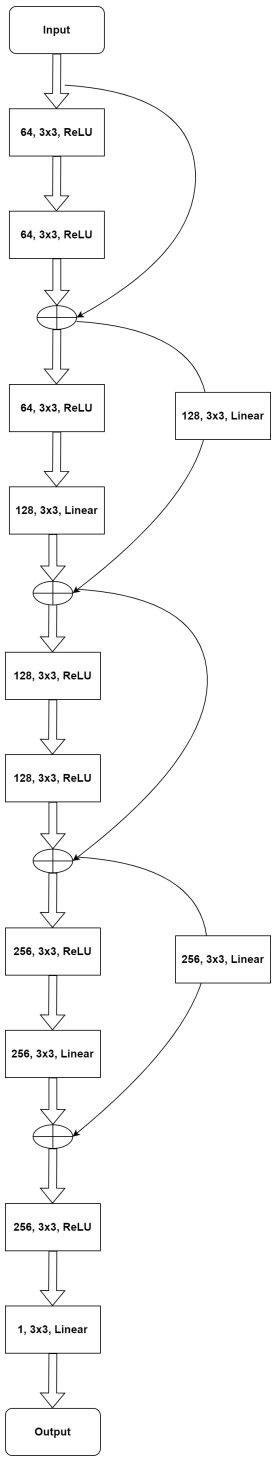

Figure 6.3: Deep CNN with each layer showing the number of filters, size of each filter, and the activation function applied

training very deep neural networks. Also, it helps in preserving and allows for the transfer of important features from layers close to the input layer to the latter layers [15].

Dropout is one of the regularization techniques used in deep learning to prevent overfitting. The primary concept underlying dropout is to incorporate randomness or unpredictability throughout the training process, compelling the network to acquire more resilient and generalized representations of the data. This is accomplished by randomly deactivating a certain proportion of neurons in a layer during training, a process known as "dropping out." This implies that the outputs of these neurons are temporarily omitted during both the forward and backward passes of the training process [53]. We applied dropout after each convolutional layer except the last or output layer, and a skip connection was incorporated after every two hidden layers, as shown in figure 6.4. A dropout rate of 0.3 was used.

Cascading - Using sub-networks

In Next Frame prediction, one technique used to predict the image at the timestep t_n is to train the model to predict the frame at the timestep t_n. Subsequently, to do this for the image at timestep t_{n+1} is to iteratively pass the next predicted image as the input to the model. From this technique, it can be seen that moving from one timestep to another involves a pass through the same model. Therefore, in between each subsequent image in the training process lies the model, which we refer to as a sub-network. Taking this concept, instead of training iteratively, we build sub-networks sandwich between the subsequent images, as shown in figure 6.5 in a cascading fashion. With this, the images in the input sequence are not concatenated along the channel axis rather, they are added at the input and output layers of the sub-networks. $\hat{Y}_4$ is the final output or the predicted image. Each subnet is composed of the architecture shown in figure 6.3.

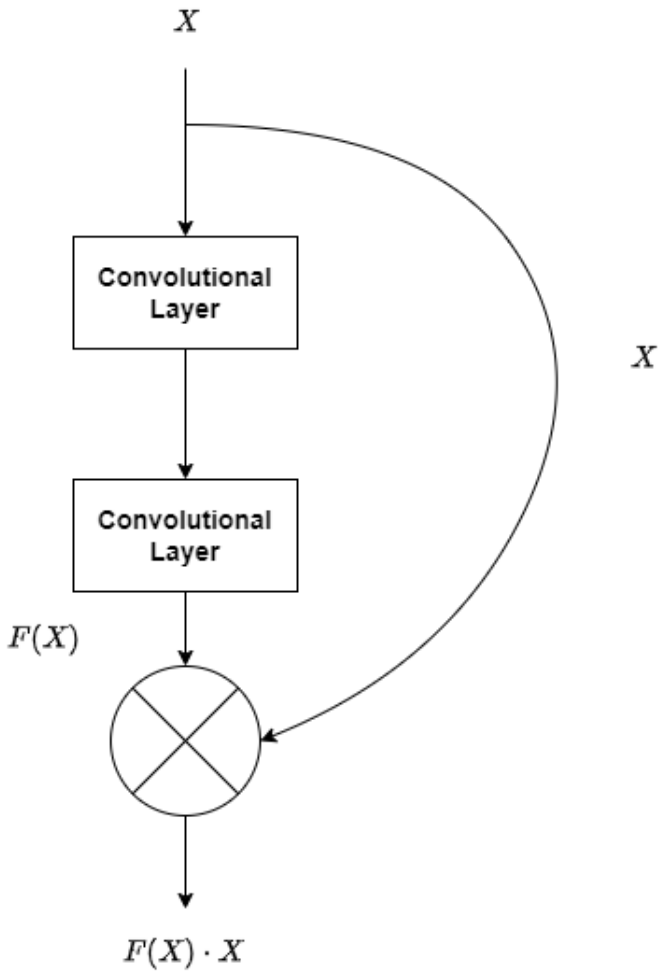

Figure 6.4: Skip Connection

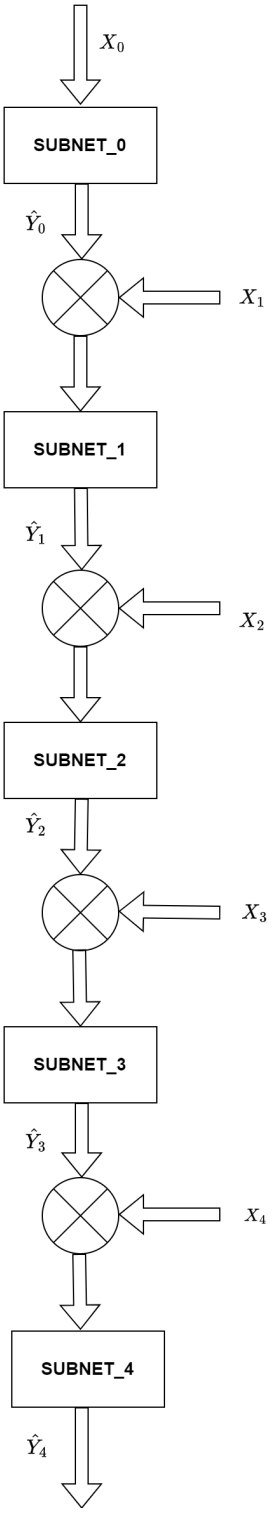

Figure 6.5: Cascading Networks

CNN - Training

The CNN model is trained end-to-end with an input of size 64×80 to generate an output of the same size. The input is a sequence consisting of five images/frames. These images are concatenated along the channel axis and feed into the network. A batch size of 32 is used. The model is complied with Adam as the optimizer with a learning rate of 0.0001 and the loss function as Mean Squared Error. The model is trained on NVIDIA GeForce RTX 2080 Ti GPUs.

6.2.3 Loss and Metrics

The following equations were incorporated into the model as the loss function, either by themselves or a combination of them for training. y is the target image, and $\hat{y}_i$ is the predicted image, and n is the number of samples or batch size.

Mean Squared Error (MSE) is used both as a metric and a loss function in equation 6.2.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{6.2}$$

Structural Similarity Index Measure (SSIM) is a commonly used metric for evaluating the similarity between two images. It evaluates an image's perceived quality based on its luminance, contrast, and structural information, imitating the attributes of the human visual system [60]. It is used as a metric, as shown in equation 6.4, and as a loss function weighted by 0.5, as shown in equation 6.3.

$$\text{SSIM\_loss} = 0.5 \times (1 - \text{SSIM}) \tag{6.3}$$

$$
\begin{aligned}
\text{SSIM} &= \frac{(2\mu_x\mu_y + C_1) \cdot (2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1) \cdot (\sigma_x^2 + \sigma_y^2 + C_2)} \\
c_1 &= (K_1 L) \quad \text{and} \quad c_2 = (K_2 L) \\
K_1 &= 0.01, \quad K_2 = 0.03 \\
x &= y_i, \quad y = \hat{y}_i
\end{aligned}
\tag{6.4}
$$

μ_x and μ_y represent the means of the two compared images.

σ_x and σ_y represent the standard deviations of the two compared images.

σ_{xy} represents the covariance between the two images.

C_1 and C_2 are constants to stabilize the division with a weak denominator.

Mean Absolute Error (MAE) is used only as a loss function as shown in equation 6.5.

$$
MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{6.5}
$$

Image Gradient represents the rate of change of pixel intensity in an image. The image gradient computation is commonly performed by employing differential operators, such as the Sobel, Prewitt, or Scharr operators. These operators provide an approximation of the derivative of the image intensity function in relation to spatial coordinates. These operations are used on the image to calculate the magnitude and direction of the gradient at each pixel.

N_X and N_Y are the magnitudes of the gradient of the target image in the x (horizontal) and y (vertical) directions, respectively, while M_X and M_Y are the magnitudes of the gradient of the target image in the x (horizontal) and y (vertical) directions, respectively (equation 6.6). Gradient Difference Magnitude (GradDiffMag or gd) is the distance between the gradients of the target image and the predicted image (equation 6.7), while Gradient Mean Squared Error (GradMean or gm) is the mean squared error between the gradient of the target image and the predicted image (equation 6.8). gd and gm are both used as loss functions.

$$N_X, N_Y = \text{Grad}(Y_i)$$
$$M_X, M_Y = \text{Grad}(\hat{Y}_i) \tag{6.6}$$

$$\text{gd} = \sqrt{\sum ((N_X - M_X)^2 + (N_Y - M_Y)^2)} \tag{6.7}$$

$$\text{gm} = \frac{(N_X - M_X)^2 + (N_Y - M_Y)^2}{n} \tag{6.8}$$

6.3 Results

6.3.1 6-hour Prediction

For this experiment, we trained the Deep CNN and Cascading CNN models to predict the changes in the active region after 6 hours. From the table 6.1, Cascading CNN outperforms the baseline in terms of MSE and SSIM by approximately 40% and 21%, respectively. Figure 6.6 shows a prediction made by the baseline, shown as last input, and the prediction made by the Cascading Model, shown as predicted. The blue and red areas show the active region as captured by the area under observation. Despite the performance of the Cascading Model based on the metrics reported, predictions from the Cascading Model look blurry, which is one of the problems convolutional models suffer from as they try to minimize the loss function, which in this case, is MSE. The prediction of the baseline looks sharper. Since a more significant portion of the target image has pixel values around zero, the model focuses on minimizing these errors and therefore makes good predictions in these areas but fails to produce sharper images around the active regions. This can be seen in figure 6.7, where regions having zero pixel values are mostly blue in the predicted image by the Cascading Model. The Difference plot 6.7 shows the absolute difference between the baseline prediction and the target image and between the Cascading model prediction and the target image.

Table 6.1: Predicting the Next Six Hours. (% Improvement is made in comparison to the baseline)

Model	**MSE**(10^{-2})	**% MSE improvement**	**SSIM**	**% SSIM improvement**
Copy Last Frame (Baseline)	35.82	0	0.4596	0
Deep CNN	21.61	39.67	0.5508	19.84
Cascading CNN	**21.4**	**40.26**	**0.5545**	**20.65**

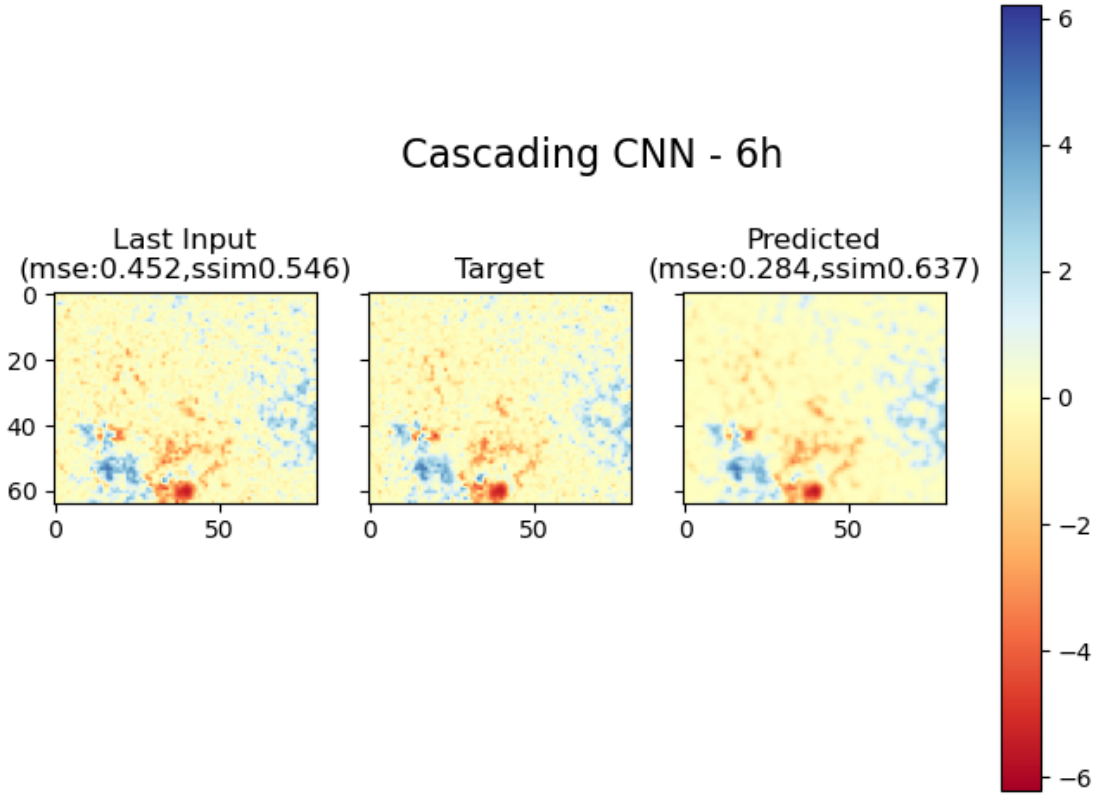

Figure 6.6: Baseline vs Predicted Image from cascading CNN

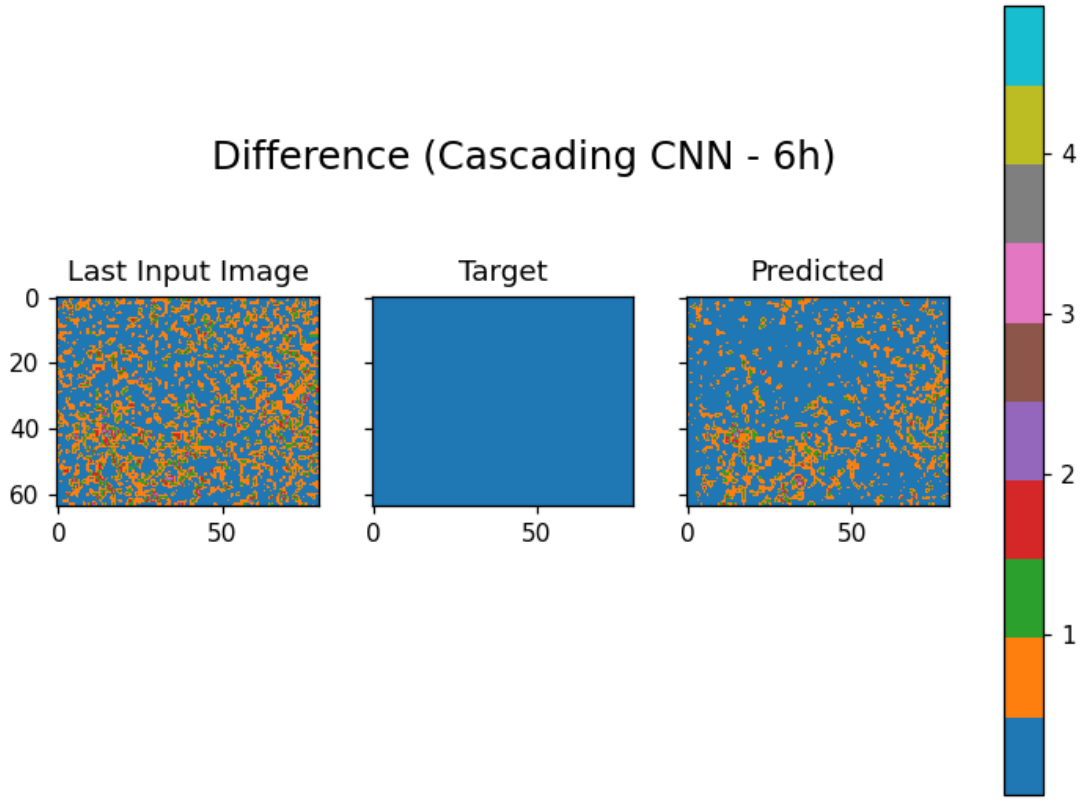

Figure 6.7: Absolute Difference - Baseline vs Predicted Image from cascading CNN

6.3.2 12-hour Prediction

To predict the next 12 hours after the last input image, we regenerated new sequences using the same criteria as used in generating the sequences for the experiment carried out in Section 6.3.1 except in this case, the target image is 12 hours ahead of the last input image. Table 6.2 shows the results obtained. The architecture of the Cascading CNN model shown in table 6.2 is the same as the one shown in figure 6.5, which was used in Section 6.3.1. The only change was the loss function used in training the model. The loss function is specified in the parenthesis in table 6.2 under the *Model* column. gm_mae_ssim loss function is a linear combination of gm, mse and $ssim$ loss functions, as shown in equation 6.9 while mse_ssim is the linear combination of mse and $ssim$ as shown in equation 6.10.

$$gm\_mae\_ssim = 0.1 * gm + mae + ssim\_loss \tag{6.9}$$

$$mse\_ssim = mse + ssim\_loss \tag{6.10}$$

From the table, Cascading CNN (MSE) with MSE as the loss function outperformed the baseline in MSE by approximately 44.43% while Cascading CNN trained with mse_ssim as the loss function outperformed the rest by 37%. The prediction made by Cascading CNN shown in figure 6.8 is much blurrier compared to the prediction made by the same model for predicting the next six hours in figure 6.6 even though it shows a higher percentage improvement in MSE and SSIM. To improve on the blurriness of predictions, we incorporated gradient information and the structural similarity index of the image as part of the loss function. Our experiment shows that adding gradient information and/or SSIM to the loss function improves the perceptual metric for the prediction, but the mean squared error degrades slightly. So while there was a significant improvement in SSIM for the models using gm_mae_ssim and mse_ssim, MSE degraded by less 1%.

Also, the difference image in figure 6.9 shows the absolute difference between the target and the predictions for the baseline and Cascading CNN (MSE) model, respectively.

Table 6.2: Predicting the Next 12 Hours. The loss function used is shown in parenthesis for Cascading CNN. (% Improvement is made in comparison to the baseline)

Model	**MSE**(10^{-2})	**% MSE improvement**	**SSIM**	**% SSIM improvement**
Copy Last Frame (Baseline)	45.43	0	0.3651	0
Cascading CNN (mse)	**25.06**	**44.84**	0.4889	33.91
Cascading CNN (gm_mae_ssim)	25.14	44.66	0.4921	34.78
Cascading CNN (mse_ssim)	25.29	44.33	**0.5002**	**37.00**

Though the predicted image by Cascading CNN (MSE) is less blue (zero), it can be seen that the central area, which contains the active regions, is less red than the prediction of the baseline. The red color shows areas whose absolute difference between the target image and prediction is above 1. Figure 6.10 shows the histogram for the predicted images shown in figure 6.8. The histogram for the predicted image by Cascading CNN (MSE) shows that the model predicted a high number of pixels with an intensity close to zero. This is evident in the blurriness of the predicted image shown in figure 6.8.

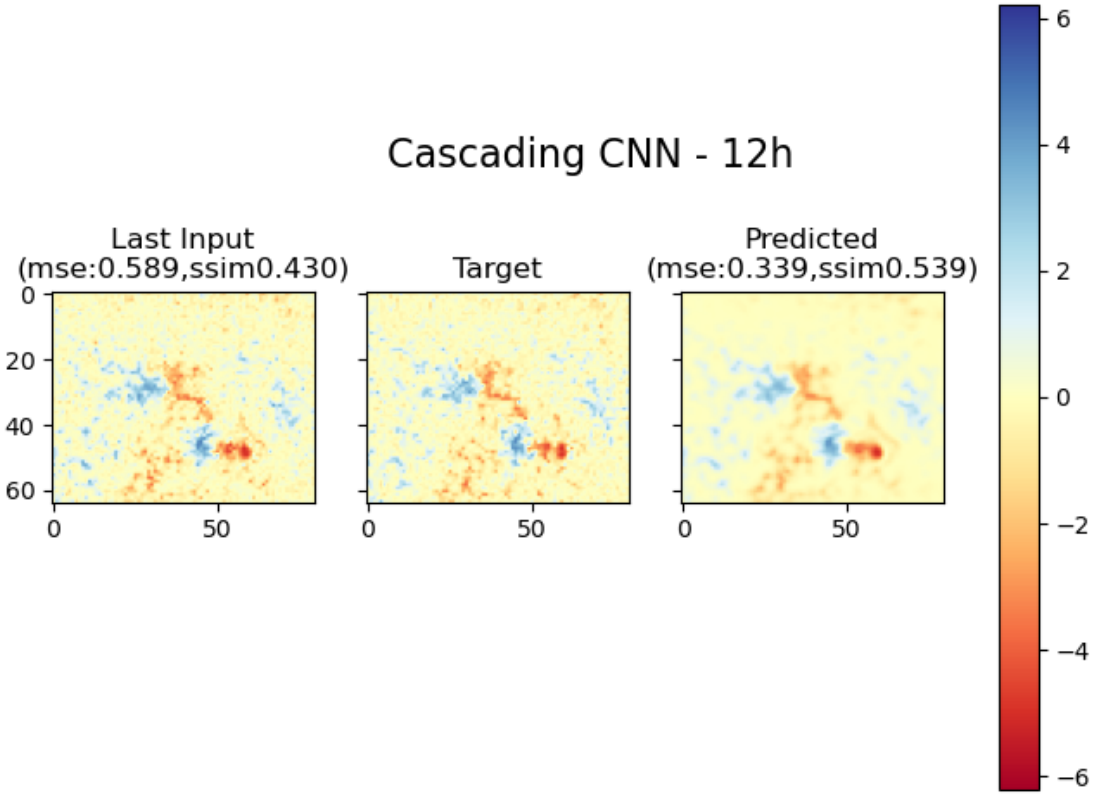

Figure 6.8: Baseline vs Predicted Image from the model with MSE as loss function

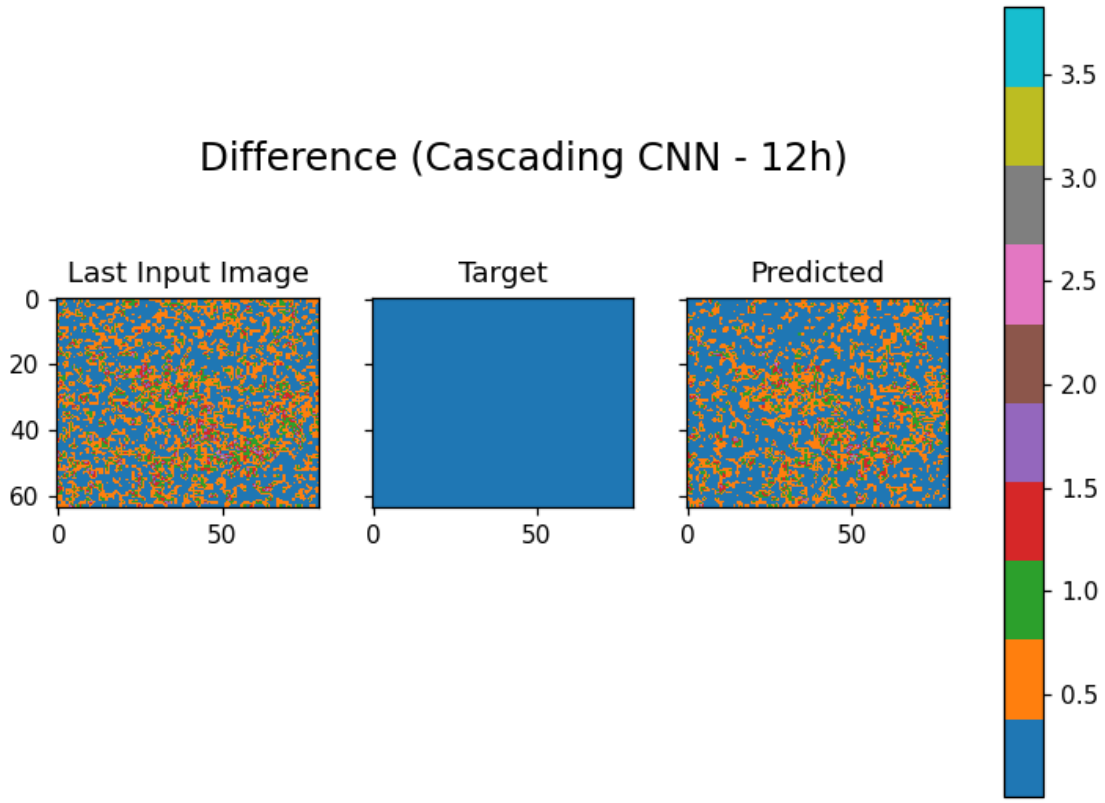

Figure 6.9: Difference - Baseline vs Predicted Image from the model with MSE as loss function

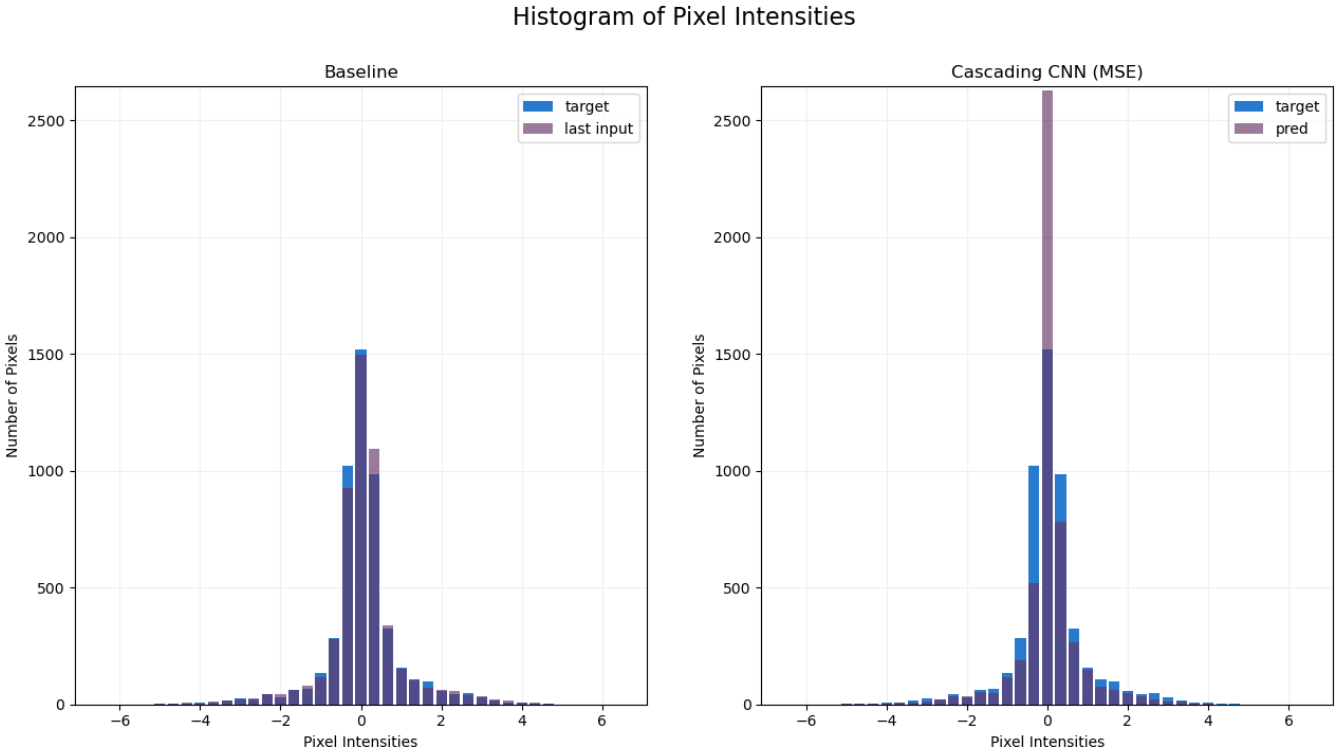

Figure 6.10: Histogram - Baseline vs Predicted Image from the model with MSE as loss function

Chapter 7

Conclusion and Future Work

In this work, we show that it is possible to model the spatiotemporal changes in active regions on the sun by using deep neural networks. The spatiotemporal changes, as seen in this work, can be grouped into two categories:

1. Changes of active regions as the Sun rotates: Predicting Changes In a Fixed Area of The Sun

2. Evolution of the active regions: Predicting Changes In Active Regions In Tracked Areas of the Sun

In general, we can conclude from the results obtained that

1. We can predict the changing magnetic field in Active regions on the Sun in a data-driven approach

2. Using exponential/logarithmic scaling helps the trained model to produce good predictions

3. By modeling solar rotation as part of the data preparation stage, there was an increase in the baseline performance

4. By incorporating information on the Sun's rotation into the sequence, we can predict further into the future compared to predicting changes in the active regions within a fixed area.

Not only can our proposed deep neural networks capture the sun's rotation, but they can also capture the changes in the active regions.

In Chapter 3, we discussed the advancements that have been made in Next-Frame prediction and the various techniques employed in training the various models to predict the next timestep. Also, we showed the work that has been done in solar flare predictions by the Astrophysics community and how it has been found that there is a correlation between solar flare occurrence and the magnetic activity of active regions or sunspots on the sun.

From Chapter 4, we described the data used in our work and how it was originally obtained. We explore the different instruments on board NASA's Solar Dynamics Observatory (SDO), which is a satellite that has been studying the sun since the early part of 2010. The data used in this work is obtained from the Helioseismic and Magnetic Imager on SDO, which has been preprocessed and cleaned by Galvez et al. [12] and downsampled to be used by machine learning researchers. This data span the year 2010 to 2018.

In Chapter 5, we applied three different models, namely:

1. Baseline (No change/Copy Last Frame)
2. Regression
3. Memory in Memory

to capture the changes of active regions with respect to the Sun. In our investigation, we discovered that scaling methods such as Min-Max Normalization, normally used in deep learning, cannot be applied to HMI data. We proposed the sigmoid normalization to scale the data for training. Using this scaling, it was found that it helps machine learning models predict images that are visually good evaluated using the Structural Similarity Index Measure (SSIM) metric.

In the final chapter, that is, Chapter 6, we proposed another scaling method called log scale and a deep neural architecture called Cascading CNN that can predict the changes occurring in active regions. Also, we concluded that the addition of the gradient information and structural similarity index between the target image and the predicted image can help to produce good images perceptually, though this caused a slight drop in the performance of the model in terms of Mean Squared Error (MSE).

Modeling the spatiotemporal changes on the Sun is a challenging task, as evidenced by our results obtained so far. Though this work begins the exploration of predicting the changes in active regions in a data-driven fashion, we believe these two ways can be used to improve on modeling the active regions on the sun:

1. Super Resolution: Super-resolution techniques based on deep learning have made great strides recently, yielding images with high fidelity that are on par with or better than those derived from more conventional interpolation techniques. They are now indispensable tools for improving image quality in a wide range of applications, helping to advance decision-making, analysis, and interpretation of images across various sectors. Currently, most of the images generated are blurry, especially as we predict further into the future. By post-processing these images through models designed for super-resolution, we believe these can help produce sharper images.

2. Incorporating Data from Other Instruments from SDO: Galvez et al. [12] in their work on developing a dataset fit for machine learning algorithms, they showed that there exists not only a correlation between the physical properties of data from the Atmospheric Imaging Assembly (AIA) and HMI but that a deep convolutional neural network can be trained to translate HMI observables to AIA observables. Combining data from these different sources can improve the predictions made by deep neural networks since neural networks do well when there is more data (data-hungry models).

3. Penalizing the Model for Overestimation: The model tends to predict more pixels with values lying close to zero as seen in figure 6.10. By formulating a loss function or model (e.g. GANs), we can constrain the model not to predict changes in active regions outside the normal distribution of active regions and therefore produce good predictions.

Chapter 8

Appendix

8.1 Intuition behind the baseline and what a simple CNN tells us

Looking at the copy-last-frame method being used as the baseline, it assigned a weight of zero to all input frames except the last frame, which received a weight of 1. The predicted frame can be shown mathematically in the equation 8.1, where x is the sequence

$$\begin{aligned} \text{Predicted Image : } \hat{Y} &= 0 * x_i + 0 * x_{i+1} + 0 * x_{i+2} + \ldots + 0 * x_{n-1} + 1 * x_n \\ & where \quad n \quad is \quad the \quad number \quad of \quad the \quad input \quad frames \end{aligned} \tag{8.1}$$

Now, to answer the question, "How does a CNN assign weights to the input frames to minimize the MSE error?", we train a CNN with one hidden layer. The hidden layer is a convolutional layer with one filter of size 1×1 and padding 'same'. The number of parameters for this CNN equals the size of the input channel or the sequence length plus a bias. The learned weights for each image in the input sequence are plotted in figures 8.1 and 8.2.

In figure 8.1, we generated sequence as discussed in chapter 6, but in this case, we used the sequences for the region under observation for one complete rotation, that is, when the region appears on the left and transitions on the visible sun disk then vanishes on the right and reappears again after approximately 13 days. Therefore, the input sequences containing 19 images each were used to train the CNN model to predict the region under observation after it reappeared on the visible sun disk. From the figure, we can see that the model assigned the highest score or weight to the region closest spatially to the target

image (that is, '0' on the x-axis). The second highest weight was given to the image closest to the target image in terms of time.

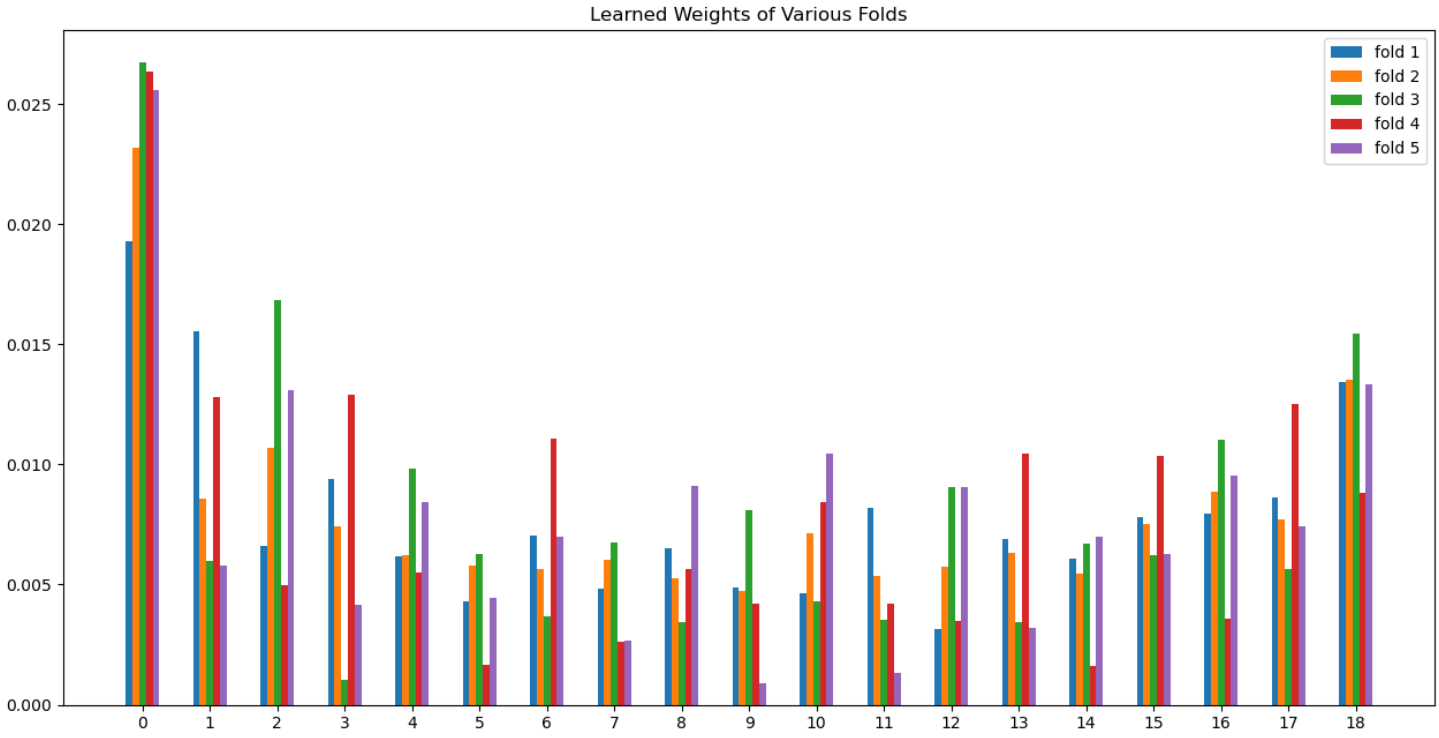

Figure 8.1: Using sequence generated from one complete rotation

In another experiment, we trained the CNN model using the sequences generated in Chapter 6, and the model assigned the highest score to the image closest to the target image. Each sequence was subdivided into sub-sequences of six images each. The CNN model was trained with these sub-sequences where the first five images of each sub-sequence were used as input and the last as output. While the baseline assigns one to the last input image and zero to the others, the CNN model assigned the highest weight to the last input image and the second highest weight to the image before the last input image. This shows that though the last input is essential to the prediction, other information can be captured

from previous timesteps. This is shown in figure 8.2.

In these two experiments, the CNN model outperformed the baseline in MSE and SSIM but failed to outperform the baseline in SSIM for the first experiment. From this observation, the CNN model learns about the spatial and temporal properties of the input sequence and assigns weights to them accordingly.

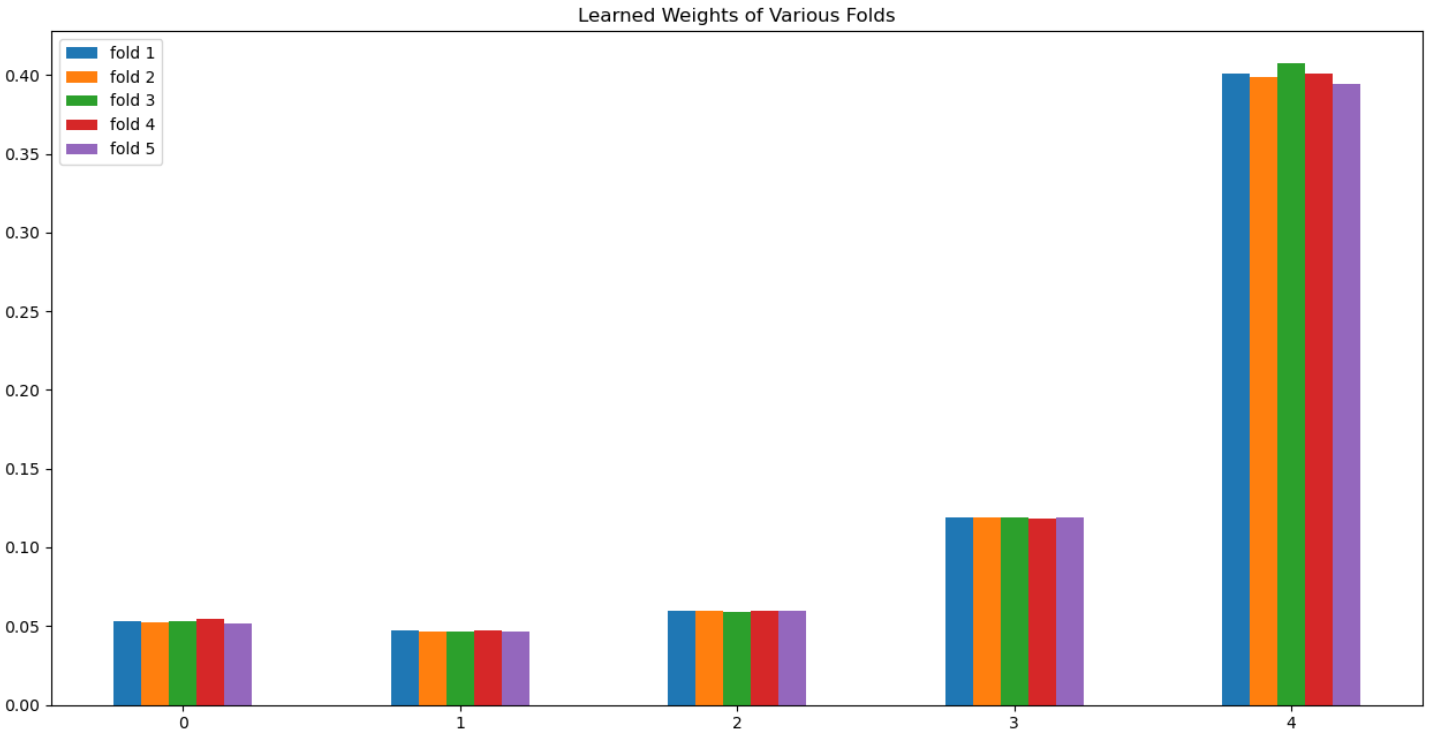

Figure 8.2: Using five previous time steps

8.2 Machine Learning Models and Deep Neural Networks

Several machine learning models and deep neural networks have predictive capability. We handpicked a few of them and trained them to predict the next image after 6 hours. The criteria used to generate the sequence is the same as in chapter 6, but in this experiment, the input sequences comprise only one image used to predict the image at the next timestep. The following models were trained and compared against each other using MSE and SSIM metrics:

- Copy Last Frame Baseline - This model used the input image as the predicted image.
- AfnoNet/FourCastNet - We used the model proposed by Pathak et al. [38], which is a modified form of vision transformer that uses an Adaptive Fourier Neural Operator (AFNO) as an efficient token mixer.
- Autoencoder - This is composed of the decoder part - three blocks, each comprising of 2 convolutional layers followed by a max-pooling layer, and the encoder part - three blocks, each comprising of 2 convolutional layers followed by an upsampling layer.
- CNN - A convolutional neural network comprising two hidden layers with 3×3 filters and the number of filters as 8 and 16 for the first and second hidden layers, respectively.
- MLP - This is a sequential model composed of four dense layers.
- Regression - We used the same regression model and the methodology used in training the regression model in Chapter 5.
- UNet - This model is similar to an autoencoder, but residual connections exist between the encoder and the decoder. Ronneberger et al. proposed this model [46].

Table 8.1: Results of Different Machine Learning Models (% Improvement is made in comparison to the baseline)

Model	**MSE** (10^{-2})	**% MSE Improvement**	**SSIM**	**% SSIM Improvement**
Copy Last Frame (Baseline)	35.99	0	0.4796	0
AfnoNet	23.60	34.4	0.5502	14.7
Autoencoder	25.53	29.1	0.5196	8.3
CNN	24.46	32.1	0.5386	12.3
MLP	40.25	-11.8	0.2835	-40.9
Regression	26.30	26.9	0.5187	8.2
Unet	35.69	0.9	0.3565	-25.7

From the results shown in the table, though AfnoNet outperforms all the others in terms of MSE and SSIM, it is to be noted that AfnoNet is enormous in terms of parameters compared to the others and also computationally expensive to train. CNN came out as the second best. We decided to pursue our investigation using this model since it has significant room for improvement and is computationally inexpensive to train in terms of time and resource utilization.

References

[1] O. W. Ahmed, R. Qahwaji, T. Colak, P. A. Higgins, P. T. Gallagher, and D. S. Bloomfield. Solar flare prediction using advanced feature extraction, machine learning, and feature selection. *Solar Physics*, 283(1):157–175, 2013.

[2] I. Arel, D. C. Rose, and T. P. Karnowski. Deep machine learning - a new frontier in artificial intelligence research. *IEEE Computational Intelligence Magazine*, 5(4):13–18, 2010.

[3] G. Barnes and K. D. Leka. Evaluating the performance of solar flare forecasting methods. *The Astrophysical Journal Letters*, 688(2), 2008.

[4] D. S. Bloomfield, P. A. Higgins, R. T. J. McAteer, and P. T. Gallagher. Toward reliable benchmarking of solar flare forecasting methods. *The Astrophysical Journal Letters*, 747(2), 2012.

[5] M. G. Bobra and S. Couvidat. Solar flare prediction using SDO/HMI vector magnetic field data with a machine-learning algorithm. *The Astrophysical Journal*, 798(2):135, 2015.

[6] S. Couvidat, J. Schou, R. Shine, R. Bush, J. Miles, P. Scherrer, and R. Rairden. Wavelength dependence of the helioseismic and magnetic imager (hmi). instrument onboard the solar dynamics observatory (sdo). *Sol Phys*, 275:285–325, 2012.

[7] Y. Cui, R. Li, L. Zhang, Y. He, and H. Wang. Correlation between solar flare productivity and photospheric magnetic field properties. *Sol Phys*, 237:45–59, 2006.

[8] J. Eastwood, E. Biffis, M. Hapgood, L. Green, M. Bisi, R. Bentley, R. Wicks, L.-A. McKinnell, M. Gibbs, and C. Burnett. The economic impact of space weather: Where do we stand? *Risk Analysis*, 37(2):206–218, 2017.

[9] D. A. Falconer. A prospective method for predicting coronal mass ejections from vector magnetograms. *Journal of Geophysical Research: Space Physics*, 2001.

[10] D. A. Falconer, R. L. Moore, A. F. Barghouty, and I. Khazanov. Prior flaring as a complement to free magnetic energy for forecasting solar eruptions. *The Astrophysical Journal*, 757(1), 2012.

[11] G. H. Fisher, D. J. Bercik, B. T. Welsch, and H. S. Hudson. Global forces in eruptive solar flares: The lorentz force acting on the solar atmosphere and the solar interior. *SoPh*, 277:59, 2011.

[12] R. Galvez, D. F. Fouhey, M. Jin, A. Szenicer, A. Muñoz-Jaramillo, M. C. M. Cheung, P. J. Wright, M. G. Bobra, Y. Liu, and J. Mason. A machine learning dataset prepared from the nasa solar dynamics observatory mission. *The Astrophysical Journal*, 242(7), 2019.

[13] I. J. Goodfellow, J. Pougetabadie, M. Mirza, B. Xu, D. Wardefarley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. In *In NeurIPS Volume 3*, pages 2672–2680, 2014.

[14] M. J. Hagyard, J. B. S. Jr., D. Teuber, and E. A. West. A quantitative study relating observed shear in photospheric magnetic fields to repeated flaring. *Sol Phys*, 91:115–126, 1984.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2016.

[16] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[17] J. T. Hoeksema, Y. Liu, K. Hayashi, X. Sun, J. Schou, S. Couvidat, A. Norton, M. Bobra, R. Centeno, K. Leka, G. Barnes, and M. J. Turmon. The helioseismic and

magnetic imager (hmi) vector magnetic field pipeline: Overview and performance. *Sol Phys*, 289:3483–3530, 2014.

[18] X. Huang and H.-N. Wang. Solar flare prediction using highly stressed longitudinal magnetic field parameters. *Research in Astronomy and Astrophysics*, 13(3), 2013.

[19] X. Huang, D. Yu, Q. Hu, H. Wang, and Y. Cui. Short-term solar flare prediction using predictor teams. *Sol Phys*, 263:175–184, 2010.

[20] N. Kalchbrenner, A. van den Oord, K. Simonyan, I. Danihelka, O. Vinyals, A. Graves, and K. Kavukcuoglu. Video pixel networks. In *In ICML*, 2017.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[22] B. J. LaBonte, M. K. Georgoulis, , and D. M. Rust. Survey of magnetic helicity injection in regions producing x-class flares. *The Astrophysical Journal*, 671:955, 2007.

[23] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[24] K. Lee, Y.-J. Moon, J.-Y. Lee, K.-S. Lee, and H. Na. Solar flare occurrence rate and probability in terms of the sunspot classification supplemented with sunspot area and its changes. *Sol Phys*, 281:639–650, 2012.

[25] K. D. Leka and G. Barnes. Photospheric magnetic field properties of flaring versus flare-quiet active regions. ii. discriminant analysis. *The Astrophysical Journal*, 595(2), 2003.

[26] K. D. Leka and G. Barnes. Photospheric magnetic field properties of flaring versus flare-quiet active regions. iv. a statistically significant sample. *The Astrophysical Journal*, 656:1173, 2007.

[27] R. Li, Y. Cui, H. He, and H. Wang. Application of support vector machine combined with K-nearest neighbors in solar flare and solar proton events forecasting. *Advances in Space Research*, 42(9):1469–1474, 2008.

[28] R. Li, H.-N. Wang, H. He, Y.-M. Cui, and Z.-L. Du. Support vector machine combined with k-nearest neighbors for solar flare forecasting. *Chinese Journal of Astronomy and Astrophysics*, 7(3):441, 2007.

[29] W. Lotter, G. Kreiman, and D. Cox. Deep predictive coding networks for video prediction and unsupervised learning, 2017.

[30] M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, 2015.

[31] J. P. Mason and J. T. Hoeksema. Testing automated solar flare forecasting with 13 years of michelson doppler imager magnetograms. *The Astrophysical Journal*, 723(1), 2010.

[32] P. S. McIntosh. The classification of sunspot groups. *Solar Physics*, 125(2):251–267, 1990.

[33] R. L. Moore, D. A. Falconer, and A. C. Sterling. The limit of magnetic-shear energy in solar active regions. *The Astrophysical Journal*, 750(24), 2012.

[34] NASA. `https://www.nasa.gov/mission_pages/sunearth/spaceweather/index.html`.

[35] N. Nishizuka, K. Sugiura, Y. Kubo, M. Den, and M. Ishii. Deep flare net (defn) model for solar flare prediction. *The Astrophysical Journal*, 858(2):113, May 2018.

[36] N. Nishizuka, K. Sugiura, Y. Kubo, M. Den, S. Watari, and M. Ishii. Solar flare prediction model with three machine-learning algorithms using ultraviolet brightening and vector magnetograms. *The Astrophysical Journal*, 835(2):156, 2017.

[37] C. Olah. `http://colah.github.io/`.

[38] J. Pathak, S. Subramanian, P. Harrington, S. Raja, A. Chattopadhyay, M. Mardani, T. Kurth, D. Hall, Z. Li, K. Azizzadenesheli, P. Hassanzadeh, K. Kashinath, and A. Anandkumar. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

[39] V. Patraucean, A. Handa, and R. Cipolla. Spatio-temporal video autoencoder with differentiable memory. In *In ICLR Workshop*, 2016.

[40] K. G. Pillai, R. A. Angryk, J. M. Banda, M. A. Schuh, and T. Wylie. Spatio-temporal co-occurrence pattern mining in data sets with evolving regions. In *2012 IEEE 12th International Conference on Data Mining Workshops, Brussels*, pages 805–812, 2012.

[41] E. Priest and T. Forbes. The magnetic nature of solar flares. *The Astron Astrophys*, 10:313–377, 2002.

[42] R. Qahwaji and T. Colak. Automatic short-term solar flare prediction using machine learning and sunspot associations. *Solar Physics*, 241(1):195–211, 2007.

[43] A. Raboonik, H. Safari, N. Alipour, and M. S. Wheatland. Prediction of solar flares using unique signatures of magnetic field images. *The Astrophysical Journal*, 834(1), 2016.

[44] M. Ranzato, A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra. Video (language) modeling: a baseline for generative models of natural videos. *arXiv preprint arXiv:1412.6604*, 2014.

[45] M. Rempel and M. C. M. Cheung. Numerical simulations of active region scale flux emergence: from spot formation to decay. *The Astrophysical Journal*, 785(2):90, Mar 2014.

[46] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[47] J. Schou, P. H. Scherrer, R. I. Bush, R. Wachter, S. Couvidat, M. C. Rabello-Soares, R. S. Bogart, J. T. Hoeksema, Y. Liu, T. L. D. Jr., D. J. Akin, B. A. Allard, J. W. Miles, R. Rairden, R. A. Shine, T. D. Tarbell, A. M. Title, C. J. Wolfson, D. F. Elmore, A. A. Norton, and S. Tomczyk. Design and ground calibration of the helioseismic and magnetic imager (hmi) instrument on the solar dynamics observatory (sdo). *Sol Phys*, 275:229–259, 2012.

[48] C. J. Schrijver. A characteristic magnetic field pattern associated with all major solar flares and its use in flare forecasting. *The Astrophysical Journal Letters*, 655(L117), 2007.

[49] U. SciEd. `https://scied.ucar.edu/sun-active-region`.

[50] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W. chun Woo. Convolutional lstm network: A machine learning approach for precipitation now-casting. In *In NeurIPS*, 2015.

[51] H. Song, C. Tan, J. Jing, H. Wang, V. Yurchyshyn, and V. Abramenko. Statistical assessment of photospheric magnetic features in imminent solar flare predictions. *Solar Physics*, 254(1):101–125, 2009.

[52] space.com. `https://www.space.com/14736-sunspots-sun-spots-explained.html`.

[53] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[54] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. In *In ICML*, 2015.

[55] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *In NeurIPS*, 2014.

[56] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[57] R. Villegas, J. Yang, S. Hong, X. Lin, and H. Lee. Decomposing motion and content for natural video sequence prediction. In *In ICLR*, 2017.

[58] Y. Wang, M. Long, J. Wang, Z. Gao, and S. Y. Philip. Predrnn: Recurrent neural networks for predictive learning using spatiotemporal lstms. In *In NeurIPS*, 2017.

[59] Y. Wang, J. Zhang, H. Zhu, M. Long, J. Wang, and P. S. Yu. Memory in memory: A predictive neural network for learning higher-order non-stationarity from spatiotemporal dynamics. *cs.LG*, 2019.

[60] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. In *IEEE Transactions on Image Processing 13(4):600.* IEEE, 2004.

[61] M. S. Wheatland. A bayesian approach to solar flare prediction. *The Astrophysical Journal*, 609(2), 2004.

[62] D. Yu, X. Huang, H. Wang, and Y. Cui. Short-term solar flare prediction using a sequential supervised learning method. *Sol Phys*, 255:91–105, 2009.

[63] Y. Yuan, F. Y. Shih, J. Jing, and H.-M. Wang. Automated flare forecasting using a statistical learning technique. *Research in Astronomy and Astrophysics*, 10(8), 2010.

Curriculum Vitae

Godwill Amankwa is the second son born to Mr. Michael Mensah and Mrs. Georgina Asare. He completed St. Augustine's College in 2012 for his high school education and entered Kwame Nkrumah University of Science and Technology to pursue a bachelor's degree in Computer Engineering. After successfully completing all coursework, projects, and project defense, he graduated in June 2016. He continued to pursue his postgraduate studies in 2018.

Email: gamankwa@miners.utep.edu