Deep Graph Neural Networks for Spatiotemporal Forecasting of Sub-Seasonal Sea Ice: A Case Study in Hudson Bay

by

Zacharie Gousseau

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2024

© Zacharie Gousseau 2024

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis introduces GraphSIFNet, a novel graph-based deep learning framework for spatiotemporal sea ice forecasting. GraphSIFNet employs a Graph Long-Short Term Memory (GCLSTM) module within a sequence-to-sequence architecture to predict daily sea ice concentration (SIC) and sea ice presence (SIP) in Hudson Bay over a 90-day time horizon. The use of graph networks allows the domain to be discretized into arbitrarily specified meshes. This study demonstrates the model's ability to forecast over an irregular mesh with higher spatial resolution near shorelines, and lower resolution otherwise. Utilizing atmospheric data from ERA5 and oceanographic data from GLORYS12, the model is trained to model complex spatial relationships pertinent to sea ice dynamics. Results demonstrate the model's superior skill over a linear combination of persistence and climatology as a statistical baseline. The model showed skill particularly in short- to medium-term (up to 35 days) SIC forecasts, with a noted reduction in root mean squared error by up to 10% over the statistical baseline during the break-up season, and up to 5% in the freeze-up season. Long-term (up to 90 days) SIP forecasts also showed significant improvements over the baseline, with increases in accuracy of around 10% even at a lead time of 90 days. Variable importance analysis via feature ablation was conducted which highlighted current sea ice concentration and thickness as critical predictors. Thickness was shown to be important at longer lead times during the melting season suggesting its importance as an indicator of ice longevity, while concentration was shown to be more critical at shorter lead times which suggests it may act as an indicator of immediate ice integrity. The thesis lays the groundwork for future exploration into dynamic mesh-based forecasting, the use of more complex graph structures, and mesh-based forecasting of climate phenomena beyond sea ice.

Acknowledgments

I offer my sincere gratitude to everyone who has provided guidance and support over the course of the past two years.

First and foremost, I thank my supervisor, Andrea Scott, for her guidance, the generous offering of her time for weekly meetings and paper revisions, and her keen eye for detail. I also extend my thanks to Philippe Lamontagne for offering me the opportunity to work on such an interesting topic, and to Mohammad Sina Jahangir for his insightful comments and support during our weekly meetings.

I am grateful to my friends Neil, John, and Paulin, who made the time I spent in the lab deeply enjoyable through lunch-time philosophical musings, engaging conversation that sometimes (often) veered into intense debate, and our twice-daily ping-pong matches. I am equally grateful to the people I have met and the connections I have made along the way in various parts of the world: Ana, Jorge, Rita, Lily, Olivia, and everyone else.

Finalement, je tiens également à exprimer ma profonde gratitude envers mes parents et mes chers frères, dont les bienfaits sont trop nombreux pour être listés ici.

Dedication

This is dedicated to Patrick; my light, my treasure.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

SIC	Sea ice concentration
SIP	Sea ice presence
SIT	Sea ice thickness
GNN	Graph neural network
CNN	Convolutional neural network
RNN	Recurrent neural network
GRU	Gated recurrent unit
GCN	Graph convolutional network (Kipf and Welling convolution)
LSTM	Long-short term memory
GCLSTM	Graph convolutional long-short term memory
Seq2seq	Sequence-to-sequence

List of Symbols

$\|$	Concatenation operation
$\odot$	Element-wise multiplication
$\oslash$	Hadamard division
Δ	Difference operation
$\sigma()$	Sigmoid activation function
θ	Model parameters
$J(\theta)$	Loss function
η	Learning rate
$\mathcal{G}$	Graph
$\mathcal{V}, \mathcal{E}$	Sets of nodes (vertices) and edges in a graph
$\mathcal{N}(v)$	Set of neighbors of a node v in a graph
$\mathbf{h}_v, \mathbf{e}_{uv}$	Feature vectors for a node v and an edge relating node u to node v
α_{uv}	Attention coefficient
$*_{\mathcal{G}}$	Graph convolution operator
$\mathbf{A}$	Adjacency matrix
$\mathbf{D}$	Diagonal degree matrix
d_v	Degree of node v
$\mathbf{L}$	Graph Laplacian
$\mathbf{H}$	Node feature matrix
$\mathbf{W}, \mathbf{U}, \mathbf{V}$	Weight matrices
$\mathbf{X}$	Grid representation of the input data
$\mathbf{G}$	Mesh representation of the input data
$\mathbf{M}$	Grid-to-mesh mapping tensor
$\mathbf{P}$	Pixel count tensor for each mesh cell

Chapter 1

Introduction

The drastic loss of Arctic sea ice volume is one of the most visible and immediate impacts of climate change [70]. The Arctic is the fastest-warming region on Earth, and this warming is affecting the sea ice cover more than any other component of the climate system [78, 71, 8]. According to the National Snow and Ice Data Center (NSIDC), Arctic sea ice extent (SIE)—the total area of the Arctic Ocean with at least 15% ice cover—is seeing a steady decline. This is especially prominent in September when sea ice extent is at its minimum [63]. Declining sea cover is connected to increasing air temperatures, changes in atmospheric and oceanic circulation, the albedo feedback loop, and the concentration of greenhouse gases in the atmosphere [71]. The Arctic ice cover is of particular importance as it helps regulate the Earth's climate, and the decline in sea ice and subsequent loss of reflectivity directly contribute to the acceleration of climate change [50]. Changes in Arctic sea ice cover also disturb marine and terrestrial ecological dynamics [56]; create challenges for Northern communities [49]; and influence human activity as new trade routes become available through the Arctic [51]. Forecasting sea ice conditions is therefore becoming increasingly important as accurate knowledge of these changes would allow for more effective preparation.

This thesis introduces a deep learning based sea ice forecasting model that employs Graph Neural Networks (GNNs) integrated within a Long Short-Term Memory (LSTM) module to predict daily sea ice concentration (SIC) and sea ice presence (SIP) in Hudson Bay up to 90 days in advance. The choice of Hudson Bay as our study area is driven by its important role as a shipping hub, the presence of communities living within the region relying on maritime re-supply, and its unique characteristics as an in-land sea largely isolated from the wider Arctic. The 90-day forecasting horizon addresses the needs for planning and decision-making in industries such as shipping operations as well as the planning requirements of local communities residing in the region. This time horizon covers short-term (up to 7 days), medium-term (up to a month) and long-term (up to 3 months) planning needs. The study highlights the effectiveness of GNNs in handling irregular spatial domains by dividing Hudson Bay into a spatially irregular mesh with a higher resolution along shorelines. The performance of two types of spatial graph convolutions within the model are evaluated: the basic Graph Convolutional Network (GCN) and an attention-based transformer convolution. The model was trained using sea ice and oceanographic data from a coupled ice-ocean reanalysis product (GLORYS12 [30]), as well as atmospheric data from the ECMWF Reanalysis v5 (ERA5 [24]). The model's accuracy is validated by comparing its predictions to a statistical baseline and comparing forecasted and observed freeze-up and break-up dates at ports on Hudson Bay.

The remainder of this thesis is organized as:

- Chapter 2 provides background on the the problem of forecasting of sea ice, traditional and deep learning methods, with a focus on convolutional and graph neural networks.
- Chapter 3 details the methodology, including data sources, the study area, and the design and experimental setup of the GraphSIFNet model.
- Chapter 4 presents the results of applying GraphSIFNet for forecasting sea ice in Hudson Bay, comparing its performance with a statistical baseline and analyzing the model's attention mechanisms and input variable importance.
- Chapter 5 concludes the thesis by summarizing findings of this work, the advantages of GNNs, and suggesting directions for future research.

Chapter 2

Background

2.1 Problem Definition

Forecasting sea ice represents a critical challenge in climate science that carries significant spatial and temporal complexity. The goal is to predict the future state of the sea ice cover given the current atmospheric and oceanic conditions. It can therefore be formulated as a next-frame prediction problem. That is, given a sequence of input frames $\mathbf{X} = (\mathbf{X}_{t-n}, ..., \mathbf{X}_{t-1}, \mathbf{X}_t)$, with each frame $\mathbf{X_t} \in \mathbb{R}^{w \times h \times c}$ representing the state of sea ice at time t, the objective is to accurately predict the next T frames in the sequence $(\mathbf{X}_{t+1}, ..., \mathbf{X}_{t+T})$. Here, n denotes the number of frames in the input sequence, w and h indicate the spatial dimensions (width and height) of each frame, and c symbolizes the number of channels in the data. These channels represent the various atmospheric and oceanic variables which affect sea ice dynamics such as air or surface temperature, heat fluxes, or sea ice characteristics.

The task of sea ice forecasting is challenging due to several factors:

- **Temporal Dynamics**: The behavior of sea ice is characterized by seasonal variations, influenced by annual climatic cycles a well as local variations in temperature, energy fluxes and winds. These conditions dictate the formation, growth, and melting of sea ice, which vary from year to year. The predictability of sea ice state is further complicated by the unpredictability of seasonal variations, often influenced by broader climate trends.

- **Spatial Heterogeneity**: The spatial complexity of sea ice, with areas of varying thickness, concentration, and movement patterns, poses a challenge to models that must effectively capture and predict these detailed spatial variations.

- **Multivariate Influences**: Sea ice is subject to various influencing factors. Models must integrate a range of data sources, including atmospheric conditions, oceanographic data, and historical sea ice patterns, to make accurate predictions. Invariably, models cannot fully capture all relevant factors, thus compromises must be made in selecting the most impactful input variables.

- **Non-Linearity**: The interaction of factors affecting sea ice is marked by non-linearity–that is, non-linear relationships between input variables–demanding modeling approaches that can effectively capture these complex relationships.

- **Impacts of Climate Change**: The ever-changing climate adds an additional layer of complexity to sea ice prediction. As global temperatures rise, altering patterns and extents of sea ice, models must evolve to accommodate these changes.

For these reasons, an effective model for sea ice forecasting must be capable of integrating both spatial and temporal data, interpreting the impact of diverse environmental variables, and adapting to ongoing climatic changes. The model should ideally be versatile in its analytical approach, allowing for the inclusion of various data types and capable of discerning complex patterns in both the spatial distribution and temporal evolution of sea ice. Moreover, it should be sufficiently flexible to adjust to the changing dynamics of sea ice under the influence of global climate change, ensuring that forecasts remain relevant and accurate over time.

2.1.1 Statistical Forecasting Techniques

While traditional time series modeling techniques such as ARIMA have been widely used for forecasting, they are less effective for spatiotemporal forecasting due to their inherent limitations in handling spatial dependencies and complex temporal dynamics. ARIMA models, primarily designed for univariate time series, lack the capacity to effectively model spatial relationships and multi-dimensional data structures, which are critical in spatiotemporal forecasting. To address these limitations, methods like Vector Autoregression (VAR) [67] and Spatial Autoregressive (SAR) [3] models were developed, offering improved handling of multivariate data and spatial dependencies, respectively. However, these models still struggled with dynamic spatial relationships and non-linear interactions [86] [12]. Space-Time Autoregressive Integrated Moving Average (STARIMA) models [55] were introduced to better integrate spatial dependencies with temporal dynamics. Dynamic linear models (DLMs) and state space models [33] offered a framework for handling evolving temporal dynamics but were limited in their spatial modeling capabilities. These statistical models often utilize historical sea ice concentration, temperature, and other meteorological variables to make short-term forecasts. However, they lack the ability to adequately capture the complex spatial and temporal patterns inherent in sea ice dynamics needed to forecast over longer timeframes [41].

2.1.2 Dynamical Sea Ice Models

Dynamical models, often integrated within data assimilation systems, simulate the interactions between sea ice, atmosphere, and ocean by solving a set of physical equations or coupled partial differential equations. For instance, the Pan-Arctic Ice-Ocean Modeling and Assimilation System (PIOMAS) [89] is a coupled ice-ocean model that assimilates sea surface temperatures, sea ice concentration, and sea ice velocity data to simulate the evolution of sea ice thickness and distribution, among other variables. Another significant model is the Community Ice CodE (CICE) [27], developed by the Los Alamos National Laboratory. CICE can function as a standalone sea ice simulator or as a component within various climate models or data assimilation systems. The model has several interlinked modules, including a thermodynamic model for calculating snow and ice growth, an ice dynamics model predicting ice pack velocity based on its material strength, a transport model detailing the advection of ice concentration and volumes, and a ridging model that redistributes ice across thickness categories according to energy balances and strain rates. The neXtSIM-F [82] model is another modern stand-alone sea ice model based on Brittle-Bingham–Maxwell (BBM) sea ice rheology. It is forced by the TOPAZ ocean forecasts as well as the ECMWF atmospheric forecast, and assimilates OSI SAF sea ice concentration. The use of a more sophisticated ice rheology such as BBM leads to more realistic representations of processes like ice drift and ice deformation [58]. These models, while computationally intensive and requiring extensive calibration, are relatively reliable and have been extensively used by climate scientists, particularly in longer term climate models [28] (though they can also be used for short- and medium-term forecasting). The high computational cost, and rigidness of these models have inspired interest in alternative modelling techniques such as systems based on deep learning [2, 4, 1, 9].

2.2 Deep Learning

Artificial Neural Networks (ANNs) are foundational to the concept of deep learning. Originally inspired by the biological neural networks in the human brain (under a framework termed *connectionism*), ANNs were originally developed for understanding the biological functionality of the brain and providing a computational basis for neuroscience, as well as for performing tasks that traditional computer systems struggled with such as pattern recognition [83]. ANNs are composed of interconnected nodes or *neurons* organized in layers. These layers will typically consist of input, hidden and output layers. ANNs with many hidden layers are referred to as *deep* networks (hence the term *deep* learning). Multilayer Perceptrons (MLPs) are one of the most basic class of feedforward ANNs. An MLP consists of several fully-connected layers of nodes, including an input layer, one or more hidden layers, and an output layer. In a fully-connected MLP, each neuron in one layer is connected to each of the neurons in the subsequent layer according to a neuronal weight, which represents the connection's synaptic strength. The value of these weights are learned during training by way of an optimization algorithm. The key feature of ANNs such as MLPs is their ability to learn arbitrary non-linear functions, meaning they can be used for a myriad tasks such as classification, regression, or pattern recognition.

2.2.1 Basic Concepts

The following subsections provide some background on the fundamental concepts in deep learning needed to understand the model proposed in this thesis. Much more could be written about the mathematics and statistics underlying deep learning, however for conciseness only crucial high-level concepts are briefly discusses here.

2.2.1.1 Activation Functions

Activation functions are crucial components of ANNs as they introduce the non-linearity required to model arbitrary functions. They typically take the form of some simple, typically continuously differentiable, non-linear function (such as the sigmoid function), and are applied to the output of all neurons of each layer. Activation functions can take many forms depending on the task. A few common activation functions are:

Sigmoid Function A smooth function bounded between 0 and 1, historically popular in binary classification tasks as its output can be interpreted as a probability. The sigmoid function is given by

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{2.1}$$

.

Hyperbolic Tangent (tanh) Similar to sigmoid but outputs values between -1 and 1, beneficial in cases where zero-centered outputs are desired. The tanh function is given by

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{2.2}$$

.

Rectified Linear Unit (ReLU) Allows only positive values to pass through, introducing non-linearity while mitigating the vanishing gradient problem. Arguably the most used activation function in deep learning due to its computational simplicity and proven effectiveness. The ReLU function is given by

$$\text{ReLU}(z) = \max(0, z) \tag{2.3}$$

.

Leaky ReLU Variation of ReLU which allows a small, non-zero gradient when the input is negative, alleviating the ‘dying ReLU” problem, that is, neurons becoming inactive due to the zero-valued derivative when inputs are negative. The Leaky ReLU function is given by

$$\text{LeakyReLU}(z) = \max(0.01z, z) \tag{2.4}$$

.

2.2.1.2 Batch Normalization

When training ANNs, data is often split into batches, that is, small subsets of the entire training set, in order to more efficiently use computational resources, decrease training time, and increase generalization [34]. Batch normalization is a commonly used technique for improving the speed and stability when training ANNs. It helps combat the issue of covariate shift, where the statistical distribution of the inputs to each layer of a deep network change during training as a result of changing the previous layer’s weights. This is achieved by normalizing the outputs of each layer by subtracting the batch mean and dividing by the batch standard deviation.

2.2.1.3 Regularization Techniques

Regularization techniques are critical in preventing overfitting in neural networks, especially when the data or model are exceedingly complex, ensuring that the model generalizes well to unseen data. Common regularization techniques are L1/L2 regularization, dropout, and early stopping:

L1 and L2 Regularization Both L1 and L2 regularization add a penalty term to the loss function to discourage the model weights from becoming too large. In L1 regularization the penalty is proportional to the absolute value of the magnitude of the weights, while in L2 the penalty is proportional to the square of the magnitude of the weights.

Dropout Dropout refers to nullifying the output of a randomly selected set of neurons during training. That is, their contribution to the subsequent layer is temporarily ignored, discouraging the model from overly relying on a single neuron's contribution.

Early Stopping Early stopping refers to monitoring the model performance on a test set, and prematurely stopping the training procedure (prior to reaching the maximum number of training epochs) if no improvement is observed over some pre-determined number of consecutive epochs. This prevents the network from continuing to learn the noise in the training data after the model fully captures the learnable patterns in the data (known as overfitting).

2.2.1.4 Backpropagation

Backpropagation is the fundamental algorithm by which ANNs are trained. During training, weights are adjusted iteratively by calculating the gradient of the loss function with respect to the each adjustable weight of the network using the chain rule. These gradients generally correspond to the contribution of each weight to the error. After each pass, the weights are adjusted according to these gradients and the learning rate, which controls the size of the step taken when updating weights. After the gradients have been calculated, thus determining the direction in which weights should be updated, an optimizer is used to perform the update step. Common optimizers are the basic batch gradient descent [60], stochastic gradient descent (SGD) [60] and the Adam optimizer [37].

Batch gradient descent Batch gradient descent is one of the most fundamental forms of the gradient descent algorithm. In this approach, the entire dataset is used to compute the gradient of the cost function for each iteration of the training process. The weight update rule in Batch Gradient Descent is given by

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_\theta J(\theta_t) \tag{2.5}$$

where θ represents the weights, η is the learning rate, and J is the loss function computed over the entire dataset. This method ensures a stable descent towards the minimum and a consistent gradient update at each iteration. However, its major drawback is the computational inefficiency, especially with large datasets, as it requires sequentially loading the entire dataset into memory and computing gradients over all data points, leading to slow iterations. This can be remedied by using mini-batch gradient descent, where the dataset is divided into smaller subsets, allowing for more efficient processing and faster convergence.

Stochastic gradient descent (SGD) SGD is a variant of the gradient descent algorithm that updates the model's weights using only a single data point (or a small batch of data points) at a time. The weight update rule in SGD is given by:

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_\theta J(\theta_t, x^{(i)}, y^{(i)}) \tag{2.6}$$

where θ represents the weights, η is the learning rate, J is the loss function, and $(x^{(i)}, y^{(i)})$ is a sample from the training data. SGD updates weights more frequently than in batch gradient descent, thus reducing computational cost and accelerating training, particularly on large datasets. The stochasticity introduces more variability into the learning process leading to a less smooth but potentially more exploratory convergence path. This helps avoid converging on local minima, often improving generalization.

Adam Adam (adaptive moment estimation) calculates an exponential moving average of the gradient and the squared gradient, and the parameters β_1 and β_2 control the decay rates of these moving averages. The weight update rule in Adam is

$$\begin{aligned}
m_t &= \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot \nabla_\theta J(\theta_t) \\
v_t &= \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot (\nabla_\theta J(\theta_t))^2 \\
\hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\
\hat{v}_t &= \frac{v_t}{1-\beta_2^t} \\
\theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{\hat{v}_t}+\epsilon} \cdot \hat{m}_t
\end{aligned} \tag{2.7}$$

where m_t and v_t are estimates of the first and second moments of the gradients, respectively, and ϵ is a small scalar used to prevent division by zero. Adam is particularly effective in large datasets and complex models due to its adaptive learning rate.

2.2.1.5 Loss Functions

Loss functions (or cost functions) are used to calculate the disparity between the output of the model and the expected output (or ground truth). This is the J function used by the optimizer to calculate the error gradients. Many loss functions exist, including task-specific loss functions, but the two most basic loss functions are the mean squared error loss, and the cross-entropy loss.

Mean Squared Error (MSE) Common in regression tasks, it computes the average of the squares of the differences between the predicted and actual values. The formula is

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2, \tag{2.8}$$

where Y_i is the actual value and $\hat{Y}_i$ is the predicted value.

Cross-Entropy Common in classification tasks, it computes the difference between the probability distributions of the model output and the expected output. For binary classification, the formula is

$$-\sum_{i=1}^{n}[Y_i \log(\hat{Y}_i) + (1 - Y_i)\log(1 - \hat{Y}_i)]. \tag{2.9}$$

Loss functions are crucial in training ANNs as they dictate the model's goal. Loss functions can be arbitrarily constructed depending on the task at hand, including combinations of functions, if it desired that the model perform well according to multiple criteria.

2.2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNN) [40] are a type of ANN optimized for processing data that possesses a grid-like structure such as raster images. The defining feature of CNNs is the convolutional layer that operates on a localized receptive field rather than operating over all data points as did the earlier fully connected ANNs. These convolutions allow for greater spatial understanding, leading to exceptional performance in tasks that require spatial reasoning such as image classification, object detection, or action recognition [42]. The CNN convolution can be mathematically expressed as

$$(f * g)(i, j) = \sum_{m,n} f(m, n)g(i - m, j - n) \tag{2.10}$$

where f represents the input, g is a two-dimensional kernel, and i, j are pixel coordinates. By stacking convolutions, CNNs can capture increasingly complex patterns at higher levels of abstraction [47]. Convolutional layers utilize kernels with learnable weights, adjusted during backpropagation, to extract pertinent features from the input. The behavior of the convolution operation is shaped by several hyperparameters including kernel size, stride, and padding. Kernel size refers to the kernel's dimensions, stride dictates the spacing of the kernel's movement across the input, and padding involves adding extra pixels around the input to maintain the spatial dimensions of the output feature map. Furthermore, CNNs often use pooling layers for down-sampling, reducing the spatial resolution of feature maps and, consequently, the network's computational complexity. A common method in this context is max pooling, which is formulated as

$$\text{MaxPooling}(A) = \max_{\text{within window}}(A) \tag{2.11}$$

where A represents a local region in the feature map. As with MLPs, CNNs incorporate non-linearity through activation functions applied after each convolution. For a more comprehensive explanation of CNNs and their applications a good reference is Zhou et al. (2020) [91]

2.2.3 Graph Neural Networks

Graph Neural Networks (GNNs) are a class of ANNs designed to operate on graph-structured data. Like CNNs, they aim to extract patterns from data through layers that progressively capture local structures and, eventually, global context within the graph. Zhou et al. (2020) [91] gives a good comprehensive overview of GNNs, including a systematic taxonomy of model types and examples of real-world applications. This section presents a more succinct overview, focusing on concepts relevant to the model proposed in this thesis.

2.2.3.1 Fundamental Concepts

A graph, denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, consists of nodes (or vertices) V and edges E arbitrarily connecting these nodes. Each node $v \in \mathcal{V}$ can have associated features $\mathbf{h}_v$, and each edge $(u, v) \in \mathcal{E}$ can also have associated features $\mathbf{e}_{uv}$ for the edge connection node u to node v. A graph may be directed or undirected, where an undirected graph contains edges with no specified directionality ($\mathbf{e}_{uv} \equiv \mathbf{e}_{vu}$) and a directed graph contains directed edges ($\mathbf{e}_{uv} \neq \mathbf{e}_{vu}$). This distinction is important as it influences graph processing (e.g. in a graph convolution, the direction of the edges dictates the nodes involved in each node's convolution). Graph edges may also be weighted, giving each edge a scalar weight corresponding to the strength of each connection.

To efficiently represent and manipulate graphs in a computational context, graphs are often represented using matrices. Node features are represented using a feature matrix $\mathbf{H} \in \mathbb{R}^{|\mathcal{V}| \times d_n}$ for features of dimensionality d_n, while edge features are represented using a matrix $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}| \times d_e}$ for edge features of dimensionality d_e. Edge connections are typically defined using an adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ where the entry $\mathbf{A}_{ij}$ is nonzero (1 for unweighted graphs or to the weight of the edge for weighted graphs) if there is an edge from node i to node j, and 0 otherwise. This can of course efficiently be represented using sparse matrix. Alternatively, the connections can be represented using a matrix $\mathcal{E} \in \mathbb{R}^{2 \times |\mathcal{E}|}$ where the first row contains the indices i and the second row contains the indices j of the connections.

The power of GNNs lies in their ability to capture the dependencies and structural information encoded in the graph. This is achieved through a process called message passing, which iteratively aggregates and transforms feature information from neighboring nodes. The process can be generically formulated as

$$\mathbf{h}_v^{(l+1)} = \text{UPDATE}^{(l)} \left(\mathbf{h}_v^{(l)}, \text{AGGREGATE}^{(l)} \left(\left\{ \mathbf{h}_u^{(l)}, \mathbf{e}_{uv}^{(l)} : u \in \mathcal{N}(v) \right\} \right) \right) \tag{2.12}$$

where $\mathbf{h}_v^{(l)}$ is the node feature vector (or node embedding) of node v at layer l, $\mathbf{e}_{uv}^{(l)}$ is the edge feature vector (or edge embedding) of the edge connecting u to v at layer l, $\mathcal{N}(v)$ denotes the neighborhood of node v, and AGGREGATE$^{(l)}$ and UPDATE$^{(l)}$ are differentiable functions specific to the GNN architecture [21]. The AGGREGATE function combines the node and/or edge feature vectors of the neighboring nodes, creating a 'message" which summarizes the information from these nodes. This is typically a permutation-invariant function as graphs normally do not order connections. The UPDATE function then combines this message with previous embedding $\mathbf{h}_v^{(l)}$ to create the new embedding $\mathbf{h}_v^{(l+1)}$. As a concrete example, the AGGREGATE function may be a simple MLP, while the UPDATE function may be a simple summation. The initial feature vector $\mathbf{h}_v^{(0)}$ is normally the input features $\mathbf{x}_v$ themselves. Each iteration aggregates information from each node's neighbors, thus after K iterations, each node has received information from all nodes K connections away, referred to as the K-hop neighbourhood. Equation 2.12 is the general form of the graph convolution, of which many specific equations have been proposed.

2.2.3.2 Graph Convolution Techniques

Graph convolution techniques form the core of many GNN architectures. They are broadly categorized into spectral and spatial approaches, distinguished by the basis on which the convolution operates.

Spectral Convolution Methods Spectral methods, based on signal processing theory, analyze graphs in the spectral domain by decomposing graphs using the graph Laplacian [11] defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$ where $\mathbf{D}$ is the diagonal degree matrix with diagonal elements $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$ (that is, the number of edges connecting to node i). The eigenvalues of the graph Laplacian (or "spectrum") are directly related to the frequencies within the graph. Specifically, the eigenvectors of $\mathbf{L}$ represent the modes of the graph, where modes with lower eigenvalues (thus lower frequencies) capture the graph's broad structural patterns, reflecting slow changes in the signal across the graph while modes with higher eigenvalues (thus higher frequencies) capture more rapid changes, corresponding to finer details or local structures within the graph. Thus, the graph Laplacian's decomposition into its eigenvalues and eigenvectors allows us to filter and analyze the graph at different levels of granularity, facilitating the removal of noise or the extraction of meaningful patterns from the graph signal [85].

Spectral methods have a strong theoretical basis in signal processing theory, however a full explanation of the underlying theory is beyond the scope of this thesis as spectral approaches are not used in the model proposed in this thesis. Still, it is instructive to introduce basic spectral methods as spatial methods can be interpreted as an approximation of the most widely used spectral method, the Chebyshev graph convolution (ChebNet) [14]. ChebNet filters graph signals by truncating the Chebishev polynomial series defined recursively by $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$, with $T_0(x) = 1$ and $T_1(x) = x$ for some signal x. ChebNet is then defined as

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\sum_{k=0}^{K} T_k(\tilde{\mathbf{L}})\mathbf{h}_v^{(l)}\mathbf{W}_k^{(l)}\right) \tag{2.13}$$

where $\mathbf{h}_v^{(l)}$ is the feature vector of node v at layer l, K is the order of the Chebyshev polynomial, $\mathbf{W}_k^{(l)}$ is the learnable weight matrix for the k-th polynomial at layer l, T_k is the k-th Chebyshev polynomial, and $\tilde{\mathbf{L}}$ is the scaled graph Laplacian matrix defined as $\tilde{\mathbf{L}} = 2\mathbf{L}/\lambda_{max} - \mathbf{I}$, where $\mathbf{L}$ is the normalized graph Laplacian, λ_{max} is the largest eigenvalue of $\mathbf{L}$, and $\mathbf{I}$ is the identity matrix [23]. The parameter K dictates the size of the neighbourhood considered by the convolution. ChebNet is only one example of a spectral graph convolution, and many more have since been proposed [91].

Spatial Convolution Methods Unlike spectral methods, spatial methods operate directly on the graph based on the nodes' spatial relations, making them perhaps more intuitive. A basic spatial convolution can be defined by simply multiplying the node embeddings H by a weight matrix W to linearly project the embeddings, propagating the node embeddings through the graph by multiplying by the adjacency matrix A, and passing the result through some non-linear function. This may be thought of as akin to one layer in a standard MLP. More formally, for a network with L layers, the convolution for layer l is

$$\mathbf{H}^{(l+1)} = \sigma(\mathbf{A}\mathbf{H}^{(l)}\mathbf{W}^{(l)}) \tag{2.14}$$

where $\mathbf{H}^{(l)}$ is the node feature matrix (or node embeddings), $\mathbf{W}^{(l)}$ is the weight matrix of the l^{th} layer, and σ is some non-linear activation function. This is the basis for the most fundamental spatial graph convolution (often simply referred to as the graph convolutional network or GCN) by Kipf and Welling (2017) [38]. The paper however makes some modifications to this formulation. First, self-loops (connecting each node to itself) are added to the adjacency matrix by simply adding the identity matrix $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$. This encourages the model to update node embeddings with new information rather than replacing them, preserving the integrity of the original node embeddings and therefore increasing stability [38]. The adjacency matrix is also normalized using the diagonal node degree matrix $\hat{\mathbf{D}}$ of $\hat{\mathbf{A}}$ with elements $\hat{\mathbf{D}}_{ii} = \sum_j \hat{\mathbf{A}}_{ij}$. Incorporating these two elements, the graph convolution network is defined as

$$\mathbf{H}^{(l+1)} = \sigma(\hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2}\mathbf{H}^{(l)}\mathbf{W}^{(l)}). \tag{2.15}$$

Note that this convolution only aggregates node features, neglecting any edge weights or edge features. Formulated on a node-level (for a single row of $\mathbf{H}$), scalar edge weights may be added by multiplying each term in the summation by the respective edge weight e_{uv}, becoming

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\sum_{u\in\mathcal{N}(v)\cup\{v\}} \frac{e_{uv}}{\sqrt{\hat{d}_v\hat{d}_u}}\mathbf{h}_u^{(l)}\mathbf{W}^{(l)}\right) \tag{2.16}$$

where $\mathbf{h}_v^{(l)}$ is the feature vector of node v at layer l, $\mathcal{N}(v)$ denotes the set of neighbors of node v, $\mathbf{W}^{(l)}$ is the weight matrix for layer l, and $\hat{d}_v$ is the degree terms from $\hat{\mathbf{D}}$ for node v. High-dimensional edge features are not considered in this convolution, instead, a scalar edge weight e_{uv} is used.

Note that this can also be framed as a simplification of ChebNet described above when using a first-order approximation. That is, the GCN considers only the first-order term ($K = 1$) of the Chebyshev polynomial and approximates the largest eigenvalue $\lambda_{max} \approx 2$. This approximation simplifies the ChebNet from Equation 2.13 to

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\mathbf{T}_0(\tilde{\mathbf{L}})\mathbf{h}_v^{(l)}\mathbf{W}_0^{(l)} + \mathbf{T}_1(\tilde{\mathbf{L}})\mathbf{h}_v^{(l)}\mathbf{W}_1^{(l)}\right). \tag{2.17}$$

Simplifying further by using a single weight matrix $\mathbf{W}^(l) = \mathbf{W}_0^{(l)} = -\mathbf{W}_1^{(l)}$ and recalling that $T_0(x) = 1$ and $T_1(x) = x$, this becomes

$$\mathbf{h}_v^{(l+1)} = \sigma\left((\mathbf{I} + \tilde{\mathbf{L}})(\mathbf{h}_v^{(l)}\mathbf{W}^{(l)})\right). \tag{2.18}$$

Finally, recalling that $\mathbf{L} = \mathbf{D} - \mathbf{A}$ and $\tilde{\mathbf{L}} = 2\mathbf{L}/\lambda_{max} - \mathbf{I}$, this is identical in form to the GCN in Equation 2.15.

Other forms of spatial graph convolutions exist beyond the basic GCN. For instance, another popular class is the GraphSAGE (Graph Sample and Aggregating) [20] graph convolution. Instead of processing the entire graph by multiplying the adjacency matrix with the weight vector as in Equation 2.15, GraphSAGE samples a fixed number of neighbors for each node to generate updated node embeddings. The generic form of GraphSAGE is expressed similarly to the generic convolution in Equation 2.12 as

$$\mathbf{h}_v^{(l+1)} = \text{UPDATE}^{(l)}\left(\mathbf{h}_v^{(l)}, \text{AGGREGATE}^{(l)}\left(\{\mathbf{h}_u^{(l)}, \mathbf{e}_{uv}^{(l)} : u \in \mathcal{N}_s(v)\}\right)\right) \tag{2.19}$$

with the only modification being the addition of a node sampling strategy $\mathcal{N}_s$, which could take the form of a fixed random sample, for instance. The node sampling procedure improves computational complexity thereby more efficiently handling large or dynamic graphs, and allows for superior inductive learning as it can generate embeddings for nodes not seen during training, contrary to the GCN that considers all nodes.

Attention mechanisms have played a key role in many important innovations in the field of artificial intelligence from language modelling [76] to computer vision tasks [15], thus attention-based GNNs have also become popular [73]. In its simplest form, an attention-based spatial graph convolution adds an attention term to the aggregation

$$\mathbf{h}_v^{(l+1)} = \text{UPDATE}^{(l)}\left(\mathbf{h}_v^{(l)}, \sum_{u\in\mathcal{N}(v)} \alpha_{uv}^{(l)} \cdot f^{(l)}\left(\mathbf{h}_u^{(l)}, \mathbf{e}_{uv}^{(l)}\right)\right) \tag{2.20}$$

where $f()$ is an arbitrary transformation function applied to the node and/or edge embeddings, and α_{uv} is an attention coefficient applied to the edge relating node u to node v, determined using some attention mechanism. These attention mechanisms normally aid in graph based tasks by allowing the model to focus on important parts of the graph. This is achieved by allowing each node to selectively attend to its neighbors based on the input features and a set of learned weights. The primary theoretical benefit of using attention in GNNs is its ability to capture context-dependent relationships in the graph, which might be overlooked by conventional aggregation methods. For instance, in graphs with complex interactions or varying node degrees, attention can provide a more adaptable and precise way to aggregate information from neighbors.

Numerous attention-based graph convolutions have been proposed. A widely used example is the graph attention network (GAT) [77] given by

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \alpha_{uv}^{(l)} \mathbf{W}_1^{(l)} \mathbf{h}_u^{(l)}\right) \tag{2.21}$$

with attention coefficients determined by

$$\alpha_{uv}^{(l)} = \text{softmax}\left(\sigma\left(\mathbf{W}_2^{(l)}\left[\mathbf{W}_1^{(l)}\mathbf{h}_v^{(l)} \| \mathbf{W}_1^{(l)}\mathbf{h}_u^{(l)}\right]\right)\right) \tag{2.22}$$

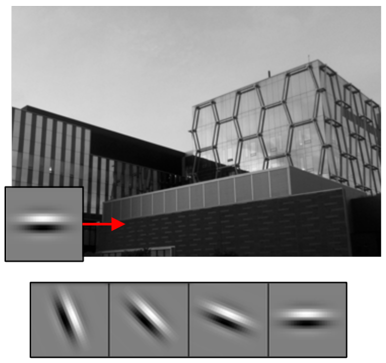
where $\mathbf{W}_1$ and $\mathbf{W}_2$ are learnable weight matrices and $\|$ denotes concatenation [85]. Another example is the transformer graph convolution [66] which uses a query-key-value self-attention mechanism rather than the simpler attention mechanism based on feature concatenation and transformation used in the GAT. The transformer convolution is given by

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\mathbf{W}_1^{(l)}\mathbf{h}_v^{(l)} + \sum_{u \in \mathcal{N}(v) \cup \{v\}} \alpha_{uv}^{(l)}(\mathbf{W}_2^{(l)}\mathbf{h}_u^{(l)} + \mathbf{W}_3^{(l)}\mathbf{e}_{uv})\right) \tag{2.23}$$

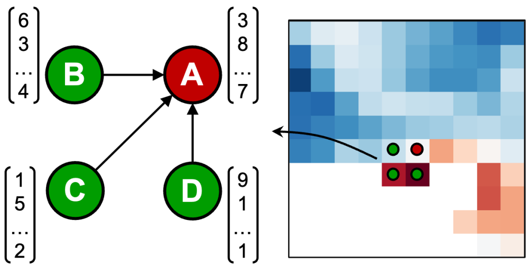
with attention coefficients determined by

$$\alpha_{uv}^{(l)} = softmax\left(\frac{(\mathbf{W}_4^{(l)}\mathbf{h}_v^{(l)})^T(\mathbf{W}_4^{(l)}\mathbf{h}_u^{(l)} + \mathbf{W}_3^{(l)}\mathbf{e}_{uv})}{\sqrt{d}}\right) \tag{2.24}$$

where $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{W}_3$, $\mathbf{W}_4$ are weight matrices and d is the feature dimensionality. This is the convolution operator used in this thesis, selected over GAT due to the ability to use edge features, and the ability to capture more nuanced node feature interactions due to the query-key-value mechanism. Note however, that in this convolution the edge features must



(a) CNN: Kernel filters (bottom figures) are learned to extract patterns in an image of a building.



(b) GNN: Non-linear functions are learned to model the relationship between neighboring nodes in a graph.

Figure 2.1: Conceptual comparison of the mechanisms of convolutional neural networks (CNN) and graph neural networks (GNN). (a) CNNs learn kernel filters which slide across the image to identify patterns in the image, such as edges or gradients. (b) GNNs learn a function to update a target node's state vector (A) by non-linearly combining the state vectors of its neighbours (B, C, D).

be projected to the same dimensionality as the node features, and they are not updated during the convolution step (i.e. they remain constant through training). In certain cases this may be a limitation of the method, however if the edge features represent physical features of graph that provide important context that we would like retained, and the desired predictions are on the node level rather than the edge level, this is not a significant limitation.

2.2.3.3 CNN vs. GNN in Sea Ice Forecasting

With the advent of deep learning, many neural network methods were developed for spatiotemporal problems, largely based on spatial convolutions with fixed-size two- or three-dimensional kernels [53]. These convolutional models are particularly well-suited for image data with a gridded structure such as images or video frames and allow for learning rich features that are present in real-world image sequences. GNNs offer a compelling alternative to CNNs for emulating models of physical processes, such as ice dynamics, for several reasons. One of the primary advantages of GNNs in this context is their inherent ability to capture the spatial relationships between neighboring nodes through graph edges, which can be arbitrarily specified. This is particularly crucial in applications like sea ice dynamics, where the spatial relationships are fundamental in determining heat and momentum exchanges, and other factors influencing ice processes. In GNNs, both nodes and edges can encode information about the system, and graph convolutions update these encodings by applying some non-linear function. This allows GNNs to effectively model the exchange of physical quantities such as heat or ice volume at a given location in space and time while accounting for the directionality of processes, which is represented by directed edges. In contrast, CNNs operate on a fundamentally different principle. They extract features such as edges or gradients from an input image by tuning kernel filters. This process involves convolving these filters over the input image to identify patterns and features at various scales and orientations. While this approach is highly effective for tasks like image recognition, where identifying and categorizing visual patterns is key, it may not be as well-suited for learning the underlying physical laws that govern interactions between points in space. CNNs typically lack the ability to explicitly model directional relationships and complex dependencies between disparate points in a spatial domain, which are critical in understanding and predicting physical phenomena like ice dynamics. A high-level visual representation of these two neural network types, highlighting their structural and functional differences, is shown in Figure 2.1. CNNs leverage spatial locality and translation invariance inherent in images through convolutional layers with fixed-size filters that extract local features across the image. Techniques such as the use of pooling operators, stride convolutions, or dilated filters can be used to capture longer-range patterns and hierarchical information [22, 88]. In contrast, message-passing GNNs can natively capture long-range patterns through edge propagation, potentially reaching across the entire graph structure given a sufficiently deep network [57]. Although in most cases the underlying graphs are too large for information to be propagated globally (e.g. a one hundred layer deep network is likely infeasible in most cases), limited information propagation across can help models gain a holistic view of the spatial domain and learn complex spatial patterns [84]. Additionally, most types of GNNs exhibit both translation and rotation invariance as convolutions are applied indiscriminately to all nodes and the aggregation operators are most often permutation invariant. Note that this is not always the case; operators based on recurrent units such as the LSTM variant of GraphSAGE [20] or sorting units such as the SortPooling aggregator [90] do not exhibit rotation invariance. Another noteworthy advantage of GNNs over CNNs is their scalability due to the inherent parallelism in their architecture, allowing for efficient processing of data over large regions or with fine resolution. This parallelism however comes at the cost of higher memory usage which may become limiting, though this can be circumvented by partitioning the graph and processing the subgraphs independently before combining the outputs. Overlapping subgraphs can be used to ensure no spatial artifacts or discontinuities arise from the partitioning.

2.2.4 Sequence Modelling

Sequence modeling plays a key role in many fields such as natural language processing (NLP), speech recognition, bioinformatics, financial forecasting, and climate modeling. Sequence modeling is the process of predicting or understanding sequences with interrelated data points, such as words, sounds, genetic data, or financial transactions. It aims to capture the dependencies and relationships within these sequences for effective analysis and prediction. One key task in the realm of sequence modelling is that of forecasting. Consider a series of data points $x_1, x_2, \ldots, x_T$. The objective is to forecast future values $x_{T+1}, x_{T+2}, \ldots, x_{T+N}$ using some function f, which is defined as

$$x_{T+n} = f(x_1, x_2, \ldots, x_T), \tag{2.25}$$

where n ranges from 1 to N, and N represents the length of the forecast horizon.

Timeseries forecasting contains additional complexities not found in analyses of static non-temporal datasets, as it requires handling the additional intricacies of temporal dynamics such as seasonality, non-stationarity, autocorrelation, and the impact of external factors. Models must therefore be sufficiently sophisticated such that they not only detect subtle temporal correlations but also be robust against irregularities and noise in the data. Models may also need to learn temporal dynamics at multiple temporal scales, or make use of multi-dimensionality in either the predictor or predictand.

Deep learning has provided powerful tools for sequence modelling, among which the Recurrent Neural Network (RNN) [75] is one of the most foundational. RNNs are designed to handle sequences of data by persisting a memory of previous inputs. Mathematically, an RNN can be described by the following equations

$$\mathbf{h}_t = \sigma(\mathbf{W}_h \mathbf{x}_t + \mathbf{U}_h \mathbf{h}_{t-1} + \mathbf{b}_h) \tag{2.26}$$

and

$$\mathbf{y}_t = \mathbf{W}_y \mathbf{h}_t + \mathbf{b}_y. \tag{2.27}$$

where $\mathbf{x}_t$ represents the input at time t, $\mathbf{h}_t$ is the hidden state at time t, $\mathbf{y}_t$ is the output vector, $\mathbf{W}_h, \mathbf{U}_h, \mathbf{W}_y$ are weight matrices, $\mathbf{b}_h, \mathbf{b}_y$ are bias vectors, and σ is some activation function. The initial hidden state ($\mathbf{h}_{-1}$) can be arbitrary selected, usually as zeros or by sampling from some distribution, while the final hidden state $\mathbf{h}_{T+1}$ can be used as a context vector which summarizes the totality of the input sequence. A visual representation of the standard RNN in its recursive and unrolled state is shown in Figure 2.2. Despite their utility in many sequence modelling tasks [43], RNNs face significant challenges, particularly the issue of vanishing and exploding gradients, which impede the network's ability to learn

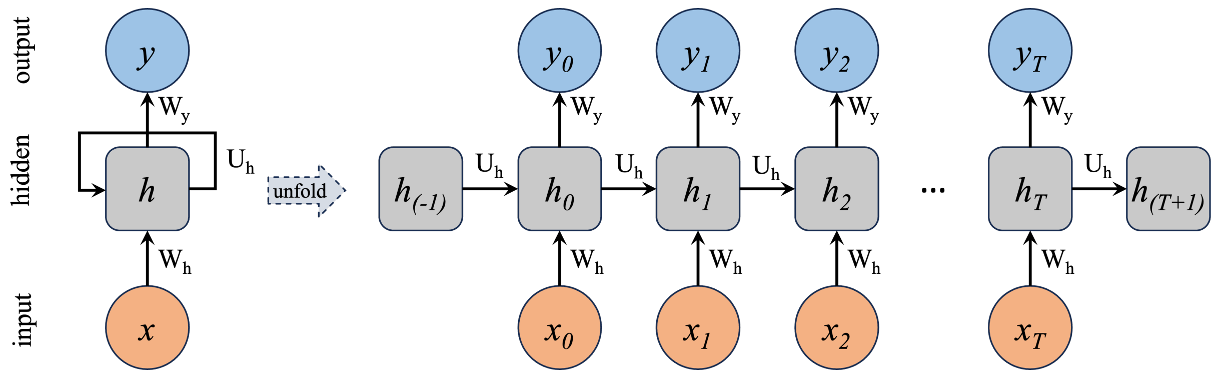


Figure 2.2: The standard RNN cell in its recursive (left) and unrolled (right) representation. Per Equation 2.26 and Equation 2.27, $\mathbf{x}_t$ denotes the input at time t, $\mathbf{h}_t$ represents the hidden state at time t, $\mathbf{y}_t$ is the output vector, and $\mathbf{W}_h, \mathbf{U}_h, \mathbf{W}_y$ are learnable weight matrices. Bias terms are omitted for simplicity.

long-range dependencies. As the sequence gets longer, the gradients of the loss function tend to either vanish (become very small) or explode (become very large), making it difficult for the RNN to learn effectively when sequences are long [64].

The Long Short-Term Memory (LSTM) [25] module was developed to address these issues. LSTMs maintain the sequential nature of RNNs but introduce a memory cell and three non-linear control gates: input, forget, and output gates which control the flow of information between LSTM cells. Cells maintains a cell state $\mathbf{c}_t$ and hidden state $\mathbf{h}_t$ which accumulates information at each step through an input gate $\mathbf{i}_t$ which allows information to be accepted into memory, a forget gate $\mathbf{f}_t$ which decides which information is discarded, and the output gate $\mathbf{o}_t$ which controls the information propagated to the final output state. This allows each cell to track gradients thereby mitigating the vanishing gradient issue during back-propagation through time which inhibits the RNN . The governing update equations of the LSTM are

$$\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f \cdot \mathbf{x}_t + \mathbf{U}_f \cdot \mathbf{h}_{t-1} + \mathbf{b}_f) \\
\mathbf{i}_t &= \sigma(\mathbf{W}_i \cdot \mathbf{x}_t + \mathbf{U}_i \cdot \mathbf{h}_{t-1} + \mathbf{b}_i) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o \cdot \mathbf{x}_t + \mathbf{U}_o \cdot \mathbf{h}_{t-1} + \mathbf{b}_o) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_c \cdot \mathbf{x}_t + \mathbf{U}_c \cdot \mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned} \tag{2.28}$$

where σ represents the sigmoid activation function, $\odot$ denotes element-wise multiplication, and tanh is the hyperbolic tangent activation function. The matrices $\mathbf{W}_{f,i,o,c}$, $\mathbf{U}_{f,i,o,c}$, and

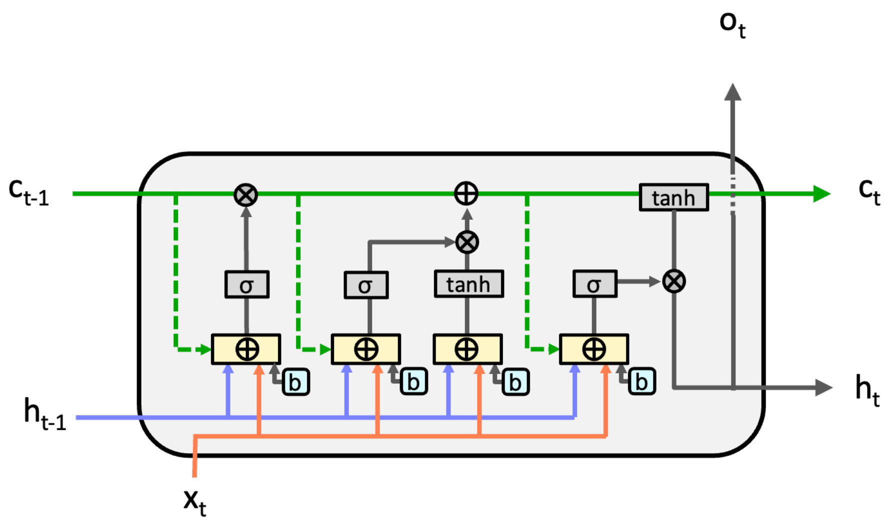


Figure 2.3: The standard LSTM cell with optional peephole connections (dotted green lines). $\oplus$ represents element-wise addition, $\otimes$ represents element-wise multiplication, and b represents the bias terms.

the vectors $\mathbf{b}_{f,i,o,c}$ are the weights and biases associated with each gate.

A variant of the LSTM is the peephole LSTM [75], which allows the gates to also look at the cell state, providing the model with greater context. The equations for a peephole LSTM are similar to that of the standard LSTM, with the addition of weight matrices $\mathbf{V}_{f,i,o}$ multiplying the cell state at each gate, given by

$$\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f \cdot \mathbf{x}_t + \mathbf{U}_f \cdot \mathbf{h}_{t-1} + \mathbf{V}_f \cdot \mathbf{c}_{t-1} + \mathbf{b}_f) \\
\mathbf{i}_t &= \sigma(\mathbf{W}_i \cdot \mathbf{x}_t + \mathbf{U}_i \cdot \mathbf{h}_{t-1} + \mathbf{V}_i \cdot \mathbf{c}_{t-1} + \mathbf{b}_i) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o \cdot \mathbf{x}_t + \mathbf{U}_o \cdot \mathbf{h}_{t-1} + \mathbf{V}_o \cdot \mathbf{c}t + \mathbf{b}_o) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_c \cdot \mathbf{x}_t + \mathbf{U}_c \cdot \mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t).
\end{aligned} \tag{2.29}$$

A visual representation of the LSTM and peephole LSTM cell is shown in Figure 2.3, where the dotted green lines represent the optional peephole connections.

Other notable variants of the recurrent network are the Gated Recurrent Unit (GRU) [10] and the Bi-direction LSTM (BiLSTM) [19]. The GRU simplifies the LSTM architecture by merging the hidden and cell states into a single vector. It uses only two gates, the reset and update gates, enabling more efficient processing of sequential data, though performance is usually poorer than LSTMs on complex sequence modelling tasks [7]. The BiLSTM enhances LSTM by processing sequences in both forward and backward directions, making it adept at capturing context by simultaneously considering information early and late timesteps in the input sequence. BiLSTMs are useful in tasks requiring more sophisticated understanding of context at the cost of greater computational requirements.

2.2.4.1 Sequence-to-Sequence

Sequence-to-sequence (seq2seq) models play a key role in timeseries forecasting as they allow mapping between input and output sequences of different lengths [74]. Seq2seq models typically use an encoder-decoder structure, where both the encoder and decoder are some recurrent module such as the RNN, GRU or LSTM (although recent advances in NLP favour transformer architectures, omitted here since they are not used in this thesis). The encoder and decoder blocks serve distinct functions. The encoder processes the input sequence, distilling it into a context vector of fixed dimensionality. This vector serves as a compressed representation of the input sequence, encapsulating its essential information. In recurrent implementations, the encoder is an RNN/GRU/LSTM that reads each element of the input sequence one at a time while updating its internal hidden state. Conversely, the decoder is tasked with generating the output sequence based on the context vector provided by the encoder. In recurrent implementations, it is also typically an RNN/GRU/LSTM and produces the output sequence one element at a time while updating its internal hidden state. For an input sequence of length M and a desired output sequence length N, the encoder processes each input sample sequentially as

$$h_m^{enc} = \mathrm{enc}(x_m, h_{m-1}^{enc}) \text{ for } m = 1, 2, ..., M \tag{2.30}$$

where enc is some recurrent cell (such as an RNN cell), x_t is the input at time t, and h_m^{enc} is the corresponding hidden state. The last hidden state of the encoder h_M^{enc} (the context vector) is then used to initiate the decoder, that is, $h_0^{dec} = h_M^{enc}$ and unrolls the sequence as

$$h_n^{dec} = \mathrm{dec}(y_{n-1}, h_{n-1}^{dec}) \text{ for } n = 1, 2, ..., N \tag{2.31}$$

where dec is also some recurrent cell and h_n^{dec} is the hidden state at each output timestep t. Each element of the decoder hidden state is transformed by some application-specific output function $f_{out}()$ as

$$y_n = f_{out}(h_n^{dec}) \tag{2.32}$$

where $f_{out}()$ may be a function such as a softmax, MLP, or a more complex function. Since the encoder and decoder operate separately and sequentially, the input length does

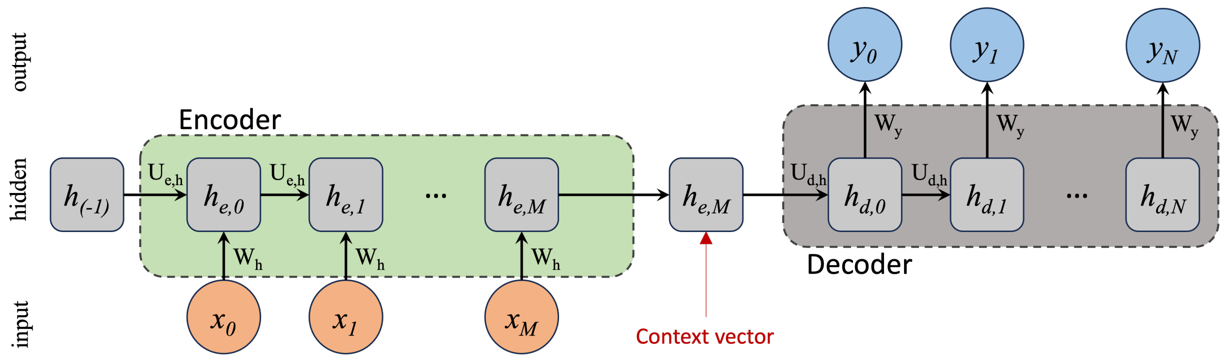


Figure 2.4: A basic sequence-to-sequence model using an RNN encoder and RNN decoder for an input sequence of length M and output sequence of length N. Per the RNN equations in Equation 2.26 and Equation 2.27, $\mathbf{x}_t$ denotes the input at timestep t, $\mathbf{h}_e, t$ and $\mathbf{h}_d, t$ represents the encoder and decoder hidden state at time t, respectively, $\mathbf{y}_t$ is the output vector, and $\mathbf{W}_h, \mathbf{U}_{e,h}, \mathbf{U}_{d,h}, \mathbf{W}_y$ are learnable weight matrices. Bias terms are omitted for simplicity.

not need to equal the output length, as is the case if a single recurrent network is used. Building on the RNN shown in Figure 2.2 based on Equation 2.26 and Equation 2.27, this encoder-decoder structure is shown conceptually in Figure 2.4.

2.2.4.2 Spatiotemporal Forecasting

Spatiotemporal forecasting extends the principles of sequence modeling to datasets where both spatial and temporal dynamics are significant. This is particularly relevant in fields such as meteorology, traffic flow prediction, and environmental modeling, where data points are not only temporally sequenced but also spatially related. The challenge in spatiotemporal forecasting is in effectively capturing the often complex interactions between the spatial and temporal components. This dual-dependency often leads to high-dimensional data, which can be challenging to analyze due to computational constraints and the risk of overfitting due to the complexity of the task.

Strictly spatial models can be used for spatiotemporal forecasting by setting the input to timestep T and the output to timestep $T+1$. However, incorporating explicit temporal modelling often enhances forecasting ability in deep learning models [81]. The simplest way to create hybrid spatiotemporal models is to combine them sequentially, for instance by first processing the inputs spatially using a CNN, then processing this encoded state using a temporal model such as an LSTM. This method may however not sufficiently exploit the interdependencies between the spatial and temporal dimensions. More sophisticated approaches have been developed to address this challenge, integrating spatial and temporal processing more closely. For example, the Convolutional LSTM (ConvLSTM) [65] combines spatial convolutions within an LSTM network, leading to strong performance in tasks such as precipitation nowcasting or video frame prediction. The core equations of a ConvLSTM are given by

$$\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f * \mathbf{x}_t + \mathbf{U}_f * \mathbf{h}_{t-1} + \mathbf{b}_f) \\
\mathbf{i}_t &= \sigma(\mathbf{W}_i * \mathbf{x}_t + \mathbf{U}_i * \mathbf{h}_{t-1} + \mathbf{b}_i) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o * \mathbf{x}_t + \mathbf{U}_o * \mathbf{h}_{t-1} + \mathbf{b}_o) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_c * \mathbf{x}_t + \mathbf{U}_c * \mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned} \tag{2.33}$$

where $*$ denotes the convolution operation, and the other symbols retain their meanings from the standard LSTM equations [65]. The ConvLSTM can capture complex spatiotemporal correlations effectively, but it can be computationally expensive due to the multiple use of convolution operations. This same approach can be extended to graph-structured data by replacing the convolution operation $*$ with a graph convolution operation $*_\mathcal{G}$. This is the graph convolutional LSTM proposed by Seo et al. (2018) [62], and it is defined as

$$\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f *_\mathcal{G} \mathbf{x}_t + \mathbf{U}_f *_\mathcal{G} \mathbf{h}_{t-1} + \mathbf{b}_f) \\
\mathbf{i}_t &= \sigma(\mathbf{W}_i *_\mathcal{G} \mathbf{x}_t + \mathbf{U}_i *_\mathcal{G} \mathbf{h}_{t-1} + \mathbf{b}_i) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o *_\mathcal{G} \mathbf{x}_t + \mathbf{U}_o *_\mathcal{G} \mathbf{h}_{t-1} + \mathbf{b}_o) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_c *_\mathcal{G} \mathbf{x}_t + \mathbf{U}_c *_\mathcal{G} \mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t).
\end{aligned} \tag{2.34}$$

2.3 Related work

The application of deep learning techniques to sea ice forecasting has gained increasing attention in recent years due to their computational efficiency and generalizability, particularly in the face of a changing climate and increased availability of large training datasets. Early studies applying deep learning to sea ice forecasting were limited to either spatial or temporal modelling. For instance, Chi and Kim (2017) [9] used a long-short term memory (LSTM) module to forecast sea ice on a per-pixel level but did not consider spatial patterns. Kim et al. (2019) [36] later used a deep neural network (DNN) with two fully-connected layers to forecast sea ice concentration considering interactions between pixels through dense layers but did not explicitly account for spatial autocorrelation. Later models based on the convolutional neural network (CNN) were able to leverage spatial patterns. Andersson et al. (2021) [2] used a U-net trained on both climate simulation and observation data to forecast monthly sea ice concentration and was found to out-perform the SEAS5 dynamical model, but did not explicitly model in the temporal dimensions. Spatiotemporal models were then proposed that unify spatial and temporal models. Liu et al. (2021) [45] proposed a model based on the convolutional long-short term memory (ConvLSTM) [65] to perform one-step ahead forecasting of sea ice in the Barents sea which showed promise by outperforming statistical baselines. Asadi et al. (2022) [4] built on this work by proposing a sequence-to-sequence model based on the ConvLSTM to forecast sea ice presence in Hudson Bay. The model generally outperformed the European Centre for Medium-Range Weather Forecasts's (ECMWF) subseasonal-to-seasonal (S2S) ensemble predictions [79].

GNN-based approaches have recently seen some attention in global climate modelling, motivated in part by successes in GNN-based physics simulation models such as MeshGraphNets [54] or graph network simulators [61, 59]. Keisler (2022) [35] first proposed a GNN for forecasting the global climate using an autoregressive encoder-processor-decoder architecture. Gridded reanalysis data was encoded onto an icosohedron graph structure on which a message-passing neural network performed several steps of processing before being decoded back onto the latitude-longitude grid. Results showed that the model is competitive in comparison with state-of-the-art physical models when forecasting geopotential height and temperature over a 6-day rollout with a 6-hour temporal step. Lam et al. (2022) [39] built upon this work with GraphCast, a similar model structure with the most notable difference being the use of multiple icosahedron grids at varying spatial resolution. They demonstrated greater skill than operational state-of-the-art physical models when forecasting global temperature, precipitation, and wind patterns over a 10-day rollout at a 6-hour temporal step. Before climate forecasting, GNNs have had a relatively long history in hydrology [72, 46, 44] and traffic forecasting [6, 31]

Chapter 3

Methodology

3.1 Data

In this thesis, the Hudson Bay systen is taken as the region of interest, and ERA5 reanalysis data is used as atmospheric forcing data to train the models along with oceanographic variables from the GLORYS12 reanalysis product. Sea ice concentration estimates from GLORYS12 are used as the target variable and a proxy for the ground truth.

3.1.1 Study Area

The Hudson Bay system (Figure 3.1) is selected as the study area due to its important role as a major shipping hub, its vital importance to local communities reliant on maritime resupply, and its unique characteristics as an in-land sea largely isolated from the broader Arctic. Hudson Bay serves as a crucial maritime trade route, connecting the Canadian interior to global markets. Accurate sea ice forecasting is essential for the safe and efficient operation of shipping routes in the region. Furthermore, the remote communities along Hudson Bay's coastlines and tributaries depend heavily on maritime transportation for essential goods. This includes the transportation of food, medical supplies, and other necessities, which are critical for their subsistence. The reliability of sea ice forecasts directly impacts the accessibility and regularity of these deliveries, affecting the well-being and resilience of these communities. Hudson Bay's unique physical characteristics as an in-land sea partially enclosed by the Canadian mainland result in distinctive sea ice patterns influenced by local climate, geography, and ocean currents. Sea ice in Hudson Bay exhibits

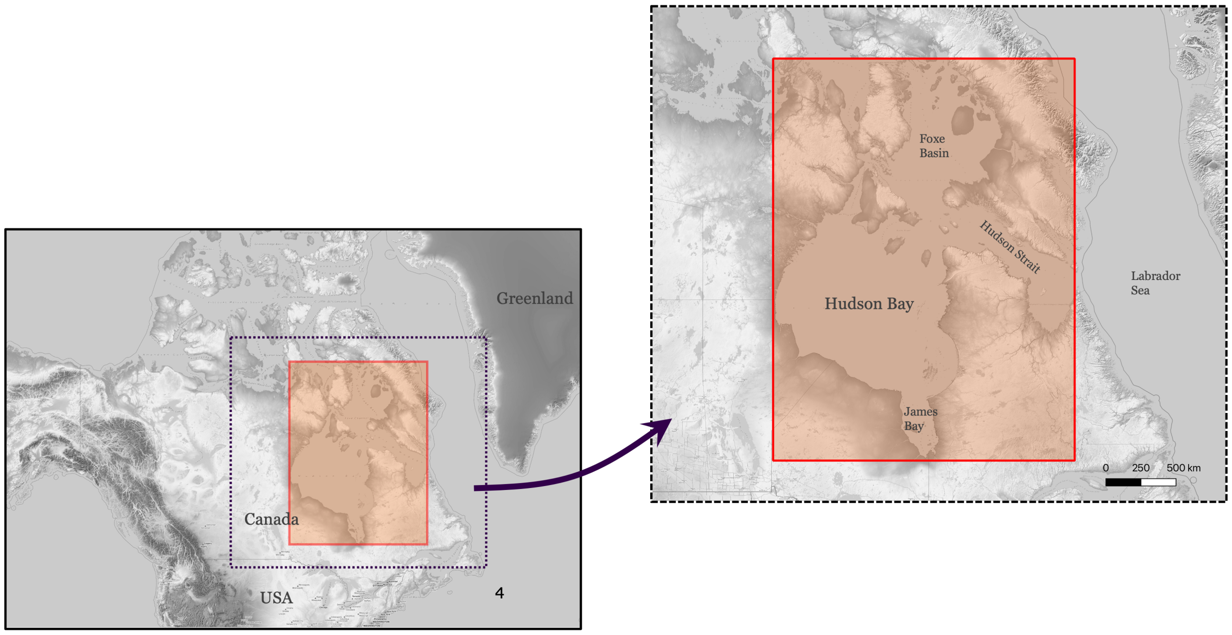


Figure 3.1: Region of interest (orange), including Hudson Bay, Foxe Basin, James Bay, and Hudson Strait. The area covers a total area of roughly 3,300,000km^2.

seasonal variations, with ice covering the entirety of the area during colder months and near complete melt during warmer months. The presence of both fast ice (ice attached to the shoreline) and pack ice (drifting with ocean currents) influences ice dynamics and navigation strategies.

3.1.2 ERA5

ERA5 [24] is a climate reanalysis dataset produced by ECMWF that offers hourly estimates of climatic variables at a spatial resolution of 0.25° from 1979 to present. It is based on the IFS Cycle 41r2 4D-Var data assimilation system and includes a wide range of climatic variables at different pressure levels of the atmosphere. The IFS system assimilates observations from dozens of satellite missions and ground stations to create a physically consistent best representation of atmospheric conditions. Although the model does not have a coupled ocean-atmosphere component, it uses daily passive microwave-derived sea ice concentration estimates from the Ocean and Sea Ice Satellite Application Facilities (OSI-SAF) as the bottom boundary condition for ice-covered seas [24]. This thesis follows previous studies [4, 2] and use 2-meter temperature, 10-meter wind speeds, and surface sensible heat fluxes from ERA5 as input features to our model (see Table 3.1)

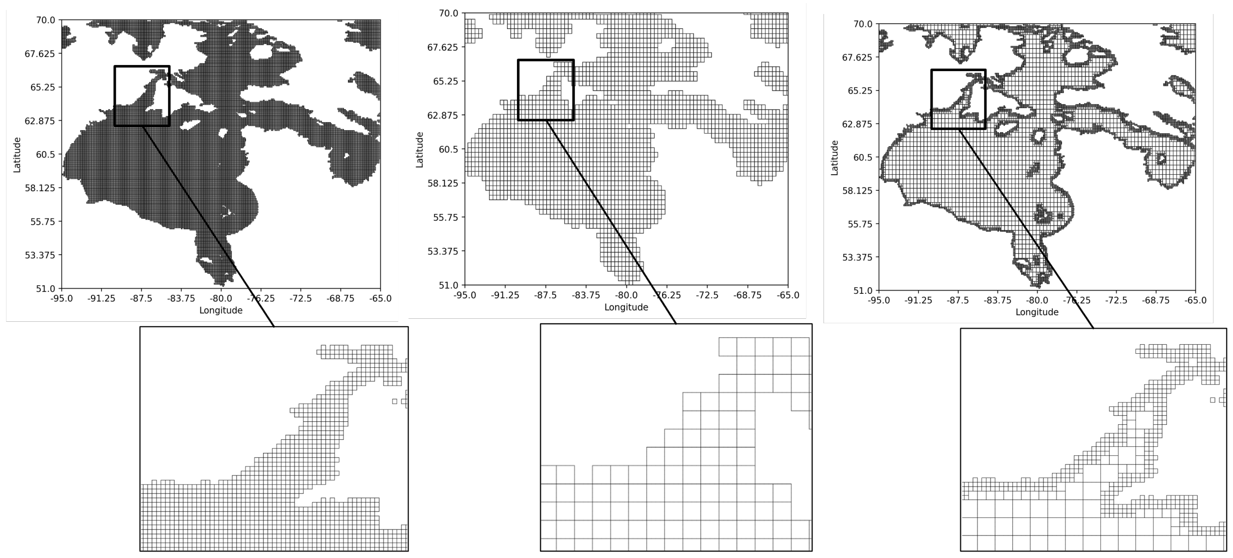
3.1.3 GLORYS12

GLORYS12 [30] is a global ocean and sea ice reanalysis data product developed by the Copernicus Marine Environment Monitoring Service (CMEMS), utilizing the LIM2 EVP NEMO 3.1 platform [48] in the ORCA025 configuration designed by the DRAKKAR consortium. This configuration includes a global sea-ice model with a 1/4° Mercator grid. Atmospheric forcing for the ocean surface model is provided by ECMWF's ERA-Interim [13] reanalysis data until 2019, and ERA5 data thereafter. The spatial resolution of the ocean and ice models is 1/12°. The data assimilation component of GLORYS12 includes in-situ temperature and salinity (T&S) profiles, satellite sea surface temperature (SST), and along track sea-level anomalies derived from satellite altimetry. The assimilation of oceanic observations occurs using a reduced-order Kalman filter, which is based on a singular evolutive extended Kalman (SEEK) filter. The SEEK filter utilizes a three-dimensional multivariate background error covariance matrix and operates on a 7-day assimilation cycle. The system also integrates sea ice concentration observations from IFREMER/CERSAT. Historical records are available from 1993 to present. This work uses GLORYS12 sea ice concentration, thickness, velocities and sea surface temperatures.

3.2 Meshing

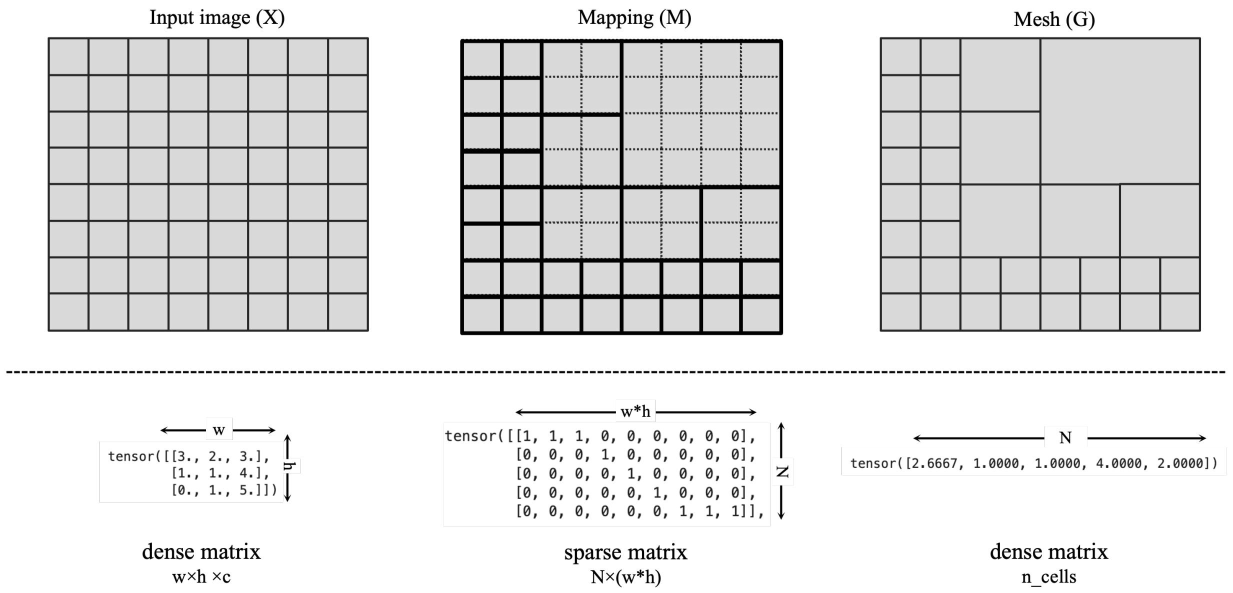
Meshes allow for greater flexibility in defining the model's spatial basis. Unlike two-dimensional convolutional approaches, which require defining a regular two-dimensional grid of pixels over a region, meshes are comprised of cells of abitrary sizes, allowing the modeler to control which areas are modelled in higher resolution (e.g., around ports or passages of interest). Since cells are only defined in regions of interest we also avoid the need to apply a land mask as a post-processing step, unlike in CNN-based approaches which most often model over the whole region before applying a mask to exclude land pixels from the output.

Figure 3.2 shows possible meshes for Hudson Bay using a 1/12 degree grid as the base resolution when trying to balance resolution and computational requirements. The mesh shown in (a) uses the base resolution as a regular mesh, which is computationally heavier with its 32,856 cells, while the mesh in (b) uses a regular four-times coarsened version of the same mesh with 2,425 cells, which may not have sufficient definition. At the shoreline,



(a) Full resolution regular mesh (b) Coarsened regular mesh (c) Irregular mesh

Figure 3.2: Comparison of different mesh definitions for modeling Hudson Bay. (a) A high-resolution regular mesh with 32,856 cells, computationally intensive but highly detailed; (b) a four-times coarsened regular mesh with 2,425 cells lacking sufficient detail along land interfaces; (c) irregular mesh with 9,422 cells, a compromise for both computational efficiency and high resolution at land interfaces. This approach ensures no cell overlaps land while providing high-resolution data for critical regions like ports, passages, and areas of meteorological interest such as the Kivalliq latent heat polynya.



```
tensor([[3., 2., 3.],
        [1., 1., 4.],
        [0., 1., 5.]])
```

```
tensor([[1, 1, 1, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 1, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 1, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 1, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 1, 1, 1]],
```

```
tensor([2.6667, 1.0000, 1.0000, 4.0000, 2.0000])
```

Figure 3.3: Conceptual visualization of the matrices involved in Equation 3.2 and Equation 3.1 (above dotted line) and an small example matrices (below dotted lines) for a single channel C. Note the intermediate matrix Y would simply be the X matrix flattened

this coarse mesh overlaps land but the model does not have the ability to acknowledge this overlapping. For example, a 4×4 cell with only one non-land pixel assigns the sea ice concentration value to the entire cell, possibly undermining the model's ability to reason about volumetric continuity. As a compromise between resolution and computational efficiency, an irregular mesh can be defined with the same four-times coarsened resolution refined near shorelines such that no cell overlaps land. This is shown in (c). This can be done by recursively splitting the cells of the base (coarsened) mesh in four equal parts until no cell overlaps land. The result is a mesh with 9,422 cells. A secondary advantage of this technique is that modelling around shorelines at a higher resolution may be of interest to port operators or local communities. For shipping and freight purposes in Hudson Bay, there is a keen interest in knowing the state of the ice near shipping ports since some operations might required ice free conditions. However, large areas of navigable waters do not require the same high degree of spatial resolution since vessels have the possibility to slightly change their routes, thus a coarser resolution is sufficient.

To convert gridded data from a grid representation $\mathbf{X} \in \mathbb{R}^{w \times h \times c}$ for data with c channels and $w \times h$ spatial dimensions to a mesh representation $\mathbf{G} \in \mathbb{R}^{c \times N}$ with N cells, we first

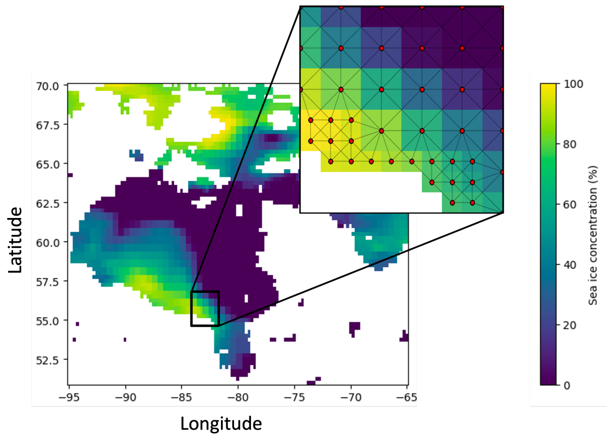


Figure 3.4: Input images are represented as graphs by relating each neighbouring pixel with edges. In this figure, a spatially irregular mesh is used to represent SIC in Hudson Bay, where red dots represent graph nodes and black lines represent edges.

construct a sparse mapping tensor $\mathbf{M} \in \mathbb{R}^{N\times(w*h)}$ where entry (n, p) is assigned 1 if the p^{th} pixel of the flattened grid $\mathbf{Y} \in \mathbb{R}^{c\times(w*h)}$ should be mapped to cell n. We also construct a tensor $\mathbf{P} \in \mathbb{R}^N$ which stores the number of pixels which are mapped to each cell. Then, to convert a sample from a grid to a mesh representation, for each node we find the mean value of each of its constituent pixels with

$$\mathbf{G} = \mathbf{Y}\mathbf{M}^T \oslash \mathbf{P} \tag{3.1}$$

where $\oslash$ represents an element-wise or Hadamard division. G can be converted back to a grid representation by splitting the cells back into its constituent pixels as

$$\hat{\mathbf{Y}} = \mathbf{G}\mathbf{M}. \tag{3.2}$$

A conceptual depiction and example matrices are shown in Figure 3.3. Note that since Equation 3.1 takes the mean of the constituent pixels of each cell, it cannot be perfectly reverted, instead Equation 3.2 simply assigns the cell value to each of its constituent pixels. Formulating these transformations as matrix multiplications allows for greater GPU acceleration which is important if the input meshes are re-meshed dynamically during training, although this is not done in this work.

A graph can then be defined based on this mesh by assigning a node to each cell and placing edges between any two neighboring cell as in Figure 3.4. To preserve spatial awareness, the positions of each node and size of each cell are added as node features, and the length (normalized distance between nodes) and angle (normalized bearing between nodes) of the edges are stored as edge features. The edges are therefore considered to be directed edges as the edge features are direction-dependent, that is, for two nodes u and v, the edge from u to v (e_{uv}) is not equivalent to the edge from v to u (e_{vu}).

3.3 Model Architecture

The proposed model uses graph convolutional long-short term memory (GCLSTM) modules within a sequence-to-sequence architecture. The GCLSTM module and the overall architecture are shown in Figure 3.5 and Figure 3.6, and described in the subsections below.

3.4 GCLSTM

The graph convolutional long-short term memory (GCLSTM) module used in this work is a modified version of the model from Seo et al. (2018) [62], which is in turn inspired by the ConvLSTM first proposed by Shi et al. (2015) [65]. The GCLSTM equations are given by

$$\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f *_{\mathcal{G}} \mathbf{x}_t + \mathbf{U}_f *_{\mathcal{G}} \mathbf{h}_{t-1} + \mathbf{V}_f \cdot \mathbf{c}_{t-1} + \mathbf{b}_f) \\
\mathbf{i}_t &= \sigma(\mathbf{W}_i *_{\mathcal{G}} \mathbf{x}_t + \mathbf{U}_i *_{\mathcal{G}} \mathbf{h}_{t-1} + \mathbf{V}_i \cdot \mathbf{c}_{t-1} + \mathbf{b}_i) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o *_{\mathcal{G}} \mathbf{x}_t + \mathbf{U}_o *_{\mathcal{G}} \mathbf{h}_{t-1} + \mathbf{V}_o \cdot \mathbf{c}_{t-1} + \mathbf{b}_o) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_c *_{\mathcal{G}} \mathbf{x}_t + \mathbf{U}_c *_{\mathcal{G}} \mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t).
\end{aligned} \tag{3.3}$$

where σ represents the sigmoid activation function, $\odot$ denotes element-wise multiplication, tanh is the hyperbolic tangent activation function, $*_{\mathcal{G}}$ is a graph convolution module, and $\mathbf{h}_t$ is the hidden state at timestep t. The matrices $\mathbf{W}$, $\mathbf{U}$, $\mathbf{V}$ and the vectors $\mathbf{b}$ are the weights and biases associated with the forget ($\mathbf{f}_t$), input ($\mathbf{i}_t$), output ($\mathbf{o}_t$), and cell state update ($\mathbf{c}_t$) at timestep t. The module is also presented visually in Figure 3.6, and closely resembles the peephole LSTM introduced by [18], with the only modification being the addition of graph convolution operators over the hidden and input states at each of the input, forget, cell and output gates in the place of weight matrices. This is represented as the $*_{\mathcal{G}}$ block in the figure. The graph convolution operators allow information exchange between nodes through the directed edges. The model proposed by Seo et al. (2018) [62] uses a single Chebyshev graph convolution [23] that has limited spatial expressivity since a

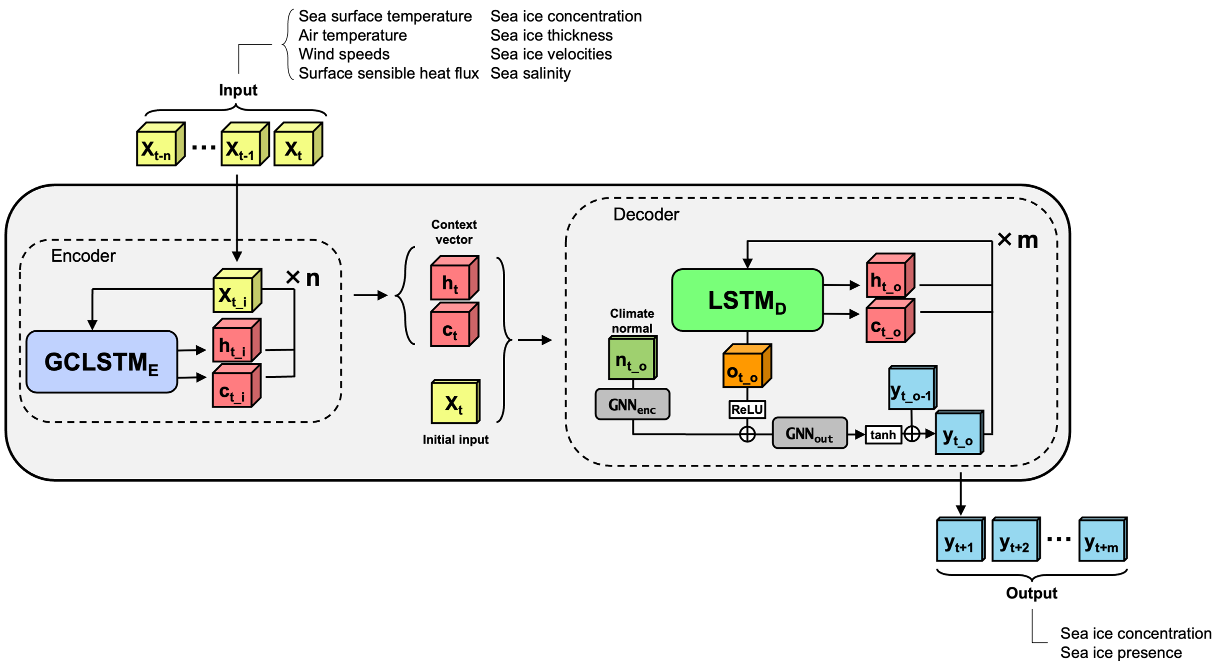


Figure 3.5: Overall model architecture. The last hidden (h_t) and cell (c_t) states of the encoder act as the context vectors and are used as the initial states of the decoder. The encoder learns features from the n input timesteps, and the last hidden (h_t) and cell (c_t) states are retained as the context vector used to initiate the decoder, which unrolls over the fixed m desired output timesteps. The initial input to the decoder X_t is the ice channel of the last input timestep. GNN_{enc} and GNN_{out}, used to encode climatology at each output timestep (n_{t_o}) and reduce the dimensionality of the output (o_{t_o}), respectively, are stacked spatial convolutions with leaky ReLU activations. $\bigoplus$ represents element-wise addition.

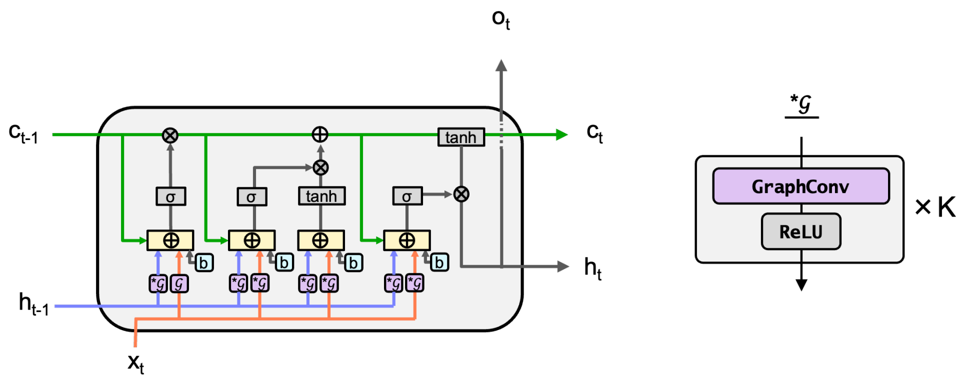


Figure 3.6: Graph convolutional long-short term memory (GCLSTM) module. The module is based on the peephole LSTM [18], with the addition of K stacked graph convolutions applied to both the hidden states and input. $\bigoplus$ represents element-wise addition, and $\bigotimes$ represents element-wise multiplication.

single convolution can only exchange information between immediate neighbors. Since the processes dominating ice formation and break-up are physical processes occurring across space, we wish to increase the model's ability to recognize spatial patterns, and therefore use K stacked convolutions followed by leaky ReLU activations, which provides information exchange over K hops. The peephole variant of the LSTM is used here as it has been shown to outperform the vanilla LSTM [32], particularly for video understanding [68]. The convolution operator taking the place of GraphConv in Figure 3.6 can be arbitrarily selected from the myriad graph convolution operators that have been proposed. In this work, the graph transformer convolution from Shi et al. (2021) [66], and the more basic Graph Convolutional Network (GCN) first proposed by Kipf and Welling [38] are evaluated.

In the graph transformer convolution, the feature vector of a given node v, $\mathbf{h}_v$, is updated by aggregating information from its neighbors $u \in \mathcal{N}(v)$, and the node itself, using edge features from u to v, $\mathbf{e}_{uv}$. The governing equation for the graph transformer convolution is

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\mathbf{W}_1^{(l)}\mathbf{h}_v^{(l)} + \sum_{u\in\mathcal{N}(v)\cup\{v\}} \alpha_{uv}^{(l)}(\mathbf{W}_2^{(l)}\mathbf{h}_u^{(l)} + \mathbf{W}_3^{(l)}\mathbf{e}_{uv})\right) \tag{3.4}$$

where $\mathcal{N}(i)$ denotes the neighbors of node i, $\mathbf{W}_{1,2,3,4}^{(l)}$ are weight matrices for layer l and the attention coefficients α_{uv} for layer l are given by

$$\alpha_{uv}^{(l)} = softmax\left(\frac{(\mathbf{W}_4^{(l)}\mathbf{h}_v^{(l)})^T(\mathbf{W}_4^{(l)}\mathbf{h}_u^{(l)} + \mathbf{W}_3^{(l)}\mathbf{e}_{uv})}{\sqrt{d}}\right) \tag{3.5}$$

where d is the dimensionality of the features. The attention weights allow the model to selectively attend to a given node's neighbors based on their node and edge feature vectors. The inclusion of edge features and an edge specific weight matrix allows the model to learn to relate the edge features to better reflect anisotropic evolution of the model state.

This transformer convolution is compared with the Graph Convolutional Network (GCN) proposed by Kipf and Welling (2017) [38], as it is a commonly used and simpler convolution operator. The GCN operator is defined by the equation

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\sum_{u\in\mathcal{N}(v)\cup\{v\}} \frac{e_{uv}}{\sqrt{\hat{d}_v\hat{d}_u}}\mathbf{h}_u^{(l)}\mathbf{W}^{(l)}\right) \tag{3.6}$$

where $\hat{d}_v$ is the degree terms from the diagonal degree matrix $\hat{\mathbf{D}}$ of the adjacency matrix with self-loops $\hat{\mathbf{A}}$ for node v and e_{uv} is a scalar edge weight between nodes u and v. Since e_{uv} must be a scalar, instead of the length and angle of the edges used in the transformer convolution, in the GCN only the length is used as the edge weights. Note that this limits the spatial awareness of the model as it does not receive information about the nodes' relative positions, unlike the transformer convolution.

3.5 Sequence-to-Sequence Architecture

The GCLSTM module is used within a sequence-to-sequence encoder-decoder structure to learn features from the inputs and evolve the sea ice state forward in time. The overall architecture is shown in Figure 3.5. Since navigation and offshore operations are affected at various degree by the presence and concentration of sea ice, our model forecasts both SIC and SIP as a multi-task learning approach. Although sea ice presence—defined as any pixel where SIC is greater than 15%—can be derived from the forecasted SIC values, a model trained without the secondary SIP forecasts would not be optimized for this 15% threshold. It was also found through experimentation that including SIP as a secondary task improved SIC forecasts in the break-up and freeze-up seasons.

The encoder is responsible for learning rich spatiotemporal features from the input sequence while the decoder is responsible for evolving the state forward in time from these learned features. The encoder therefore acts as an information bottleneck, meaning it is crucial that the encoder is sophisticated enough to distill the inputs into a context vector with sufficient information for the decoder to use in the unrolling process. Given a sufficiently rich context vector, the decoder does not necessarily need to learn additional spatial features within the context vector, nor during the unrolling process. Therefore, in this work a spatiotemporal GCLSTM module is used in the encoder block, and a simple LSTM in the decoder block. Although the decoder block also contains graph convolutions (e.g., in GNN_{out}), the distinction between the two is that the GCLSTM in the encoder block integrates graph convolutions within the temporal model allowing for simultaneous spatial and temporal modelling, while the decoder block models temporal and spatial dynamics separately, with GNN_{out} being used mainly for dimensionality reduction. Using an LSTM rather than a GCLSTM module in the decoder block also greatly reduces training time in the case where there are fewer input timesteps than output timesteps. Note that experiments with a GCLSTM in the decoder were also run but showed no improvements over using an LSTM.

The encoder processes each input timestep sequentially, updating the hidden and cell states at each timestep with layer normalization [5] applied to the hidden and cell states after each timestep to increase model stability. The final hidden and cell states are the high-dimensional vectors that are taken as the context vectors that contain the learned features from the input and are used to initialize the hidden and cell state of the decoder. The last input ice state is used as the initial input to the decoder (or start token) since we wish to evolve the state forward from this initial state. The decoder is run recurrently for the desired number of output timesteps in a similar fashion to the encoder but using the last step's prediction (y_{t-1}) as the input for the current step (y_t).

Since sea ice is highly seasonal, the model is susceptible to a form of modal collapse wherein the model converges to a local minimum, predicting only the average sea ice conditions for a particular day of the year. These daily averages are known as the climate normals or climatology. For long-term forecasting of climatological variables, climatology can perform reasonably well compared to dynamic or statistical models due to strong seasonality. Since we wish to outperform climatology and expect the model to learn to use it as a heuristic, it is included as an input such that model can focus on learning departures from normal conditions. This was shown to be beneficial for sea ice forecasting in a previous study [4]. Climate normals are calculated as the mean ice concentration values for each day of the year over the entire training set and are encoded into latent space using a shallow multi-layer GNN before being combined with the decoder output by element-wise addition. The result is then fed through a multi-layer GNN with leaky ReLU activations to reduce the dimensionality to two, and finally through a hyperbolic tangent activation to map the values between -1 and 1. This output represents the change in sea ice conditions and is added to the last timestep's prediction. Since both SIC and SIP should be bound between 0 and 1, the output is passed through a sigmoid layer that produces the final predictions.

3.6 Experimental set-up

3.6.1 Mesh Definition

To illustrate the advantage of using graph networks, experiments were designed to demonstrate the ability to produce forecasts over an irregular mesh. To this end, experiments were run on an irregular mesh as well as the coarsened regular mesh described in section 3.2 and shown in Figure 3.2b and Figure 3.2c. The irregular mesh is refined to a higher resolution at the land edges by splitting the base 1/3° mesh if a cell intersects a one-cell buffer around land. This buffer is used since near-shore dynamics can be particularly complex. By extending high-resolution meshing slightly beyond the immediate land-water interface, the model may be better equipped to capture these complex dynamics occurring in these more critical regions. The resulting irregular mesh contains 1/12°, 1/6° and 1/3° sized cells. To show that the complexities introduced by this irregular mesh is not a detriment to the model, a separate experiment is conducted by training the same model over the regular 1/3° mesh. This should be an easier task than the irregular mesh, therefore showing similar performance over either meshes is sufficient to demonstrate that the model is resolution-agnostic.

3.6.2 Data Partitioning

The Hudson Bay region, including Hudson Strait, James Bay and Foxe Basin, undergoes a cyclical transformation in its ice cover characterized by complete freezing during the winter months and total melt in the summer, with some multi-year ice possible in Foxe Basin. This seasonal cycle is subject to considerable inter-annual variability, both in terms of the rate at which these processes occur and the timing of these transitions. Figure 3.7 illustrates this variability by showing monthly SIC anomalies between 1993 and 2020. These anomalies are computed as the mean differences between observed SIC and the long-term average concentration for each corresponding month. The data reveals distinct periods of anomalous behavior in SIC. Specifically, the years 1993 to 1997 were marked

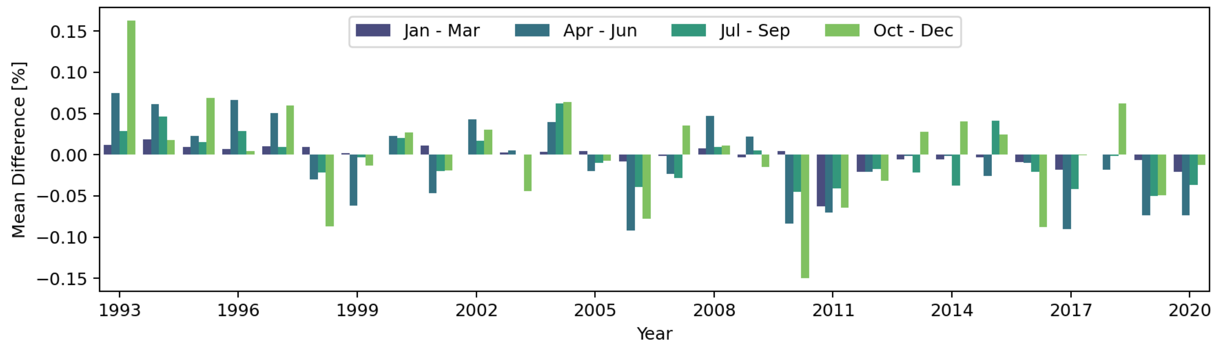


Figure 3.7: Monthly sea ice concentration anomalies in Hudson Bay from 1993-2020. Highlights periods of higher and lower-than-average sea ice concentrations.

by higher-than-average SIC, indicating that during these years, Hudson Bay experienced an earlier freeze-up and a delayed break-up season. In contrast, the period from 2010 to 2012 exhibited anomalously low SIC, characterized by a late onset of freeze-up and an earlier melting season. Including data from both these anomalous periods along with years that exhibit more typical ice conditions is critical for enhancing model robustness in the face of varying environmental conditions. This is particularly important in the context of climate change, where shifts in temperature and weather patterns could further exacerbate the variability in sea ice conditions. The data is therefore partitioned into a sequential 20-year, 3-year, 3-year split, wherein data from 1993-2013 is used for training, 2013-2016 is used for validation, and 2016-2019 is used for testing. Note however that the test period only includes years with normal or lower-than-usual ice conditions. Although this bias may not be optimal, lower-than-normal ice conditions may be more representative of future ice conditions in the Hudson region [70]

One model is trained for each month of the year, each denoted as a ‘monthly model’. Each monthly model was trained using data from the respective month with a 15-day buffer before and after the beginning and end of the month respectively. For example, the April model is trained with input data for each day between March 16 and May 15 over all training years. A longer buffer of one month was tested but did not lead to significant improvements in model performance. In inference mode, each model is used only to produce a forecast with inputs from its respective month. For example, to generate 90 day forecasts for April, a 90 day forecast is launched for each day between April 1 and April 30. Training separate model for each month of the year was done since it is expected that the dynamics that must be learned for one time of the year to be sufficiently different from other times of the year such that each model will have greater accuracy by concentrating efforts in learning specific ice dynamics [4]. As a secondary benefit, this also allows training to be carried out more efficiently as each monthly model can be trained in parallel.

3.6.3 Input Features

Sea ice concentration data from GLORYS12 serve as the target variable, while atmospheric variables from ERA5, combined with oceanographic variables from GLORYS12, are used as input features. Sea ice dynamics are primarily influenced by factors such as air and sea temperature [80], wind [69], heat fluxes [29], and ocean salinity [87], and are therefore used as input variables. The 10 chosen input variables are listed in Table 3.1, along with the rationale for their selection. It should be noted that ERA5 hourly variables are re-gridded from their original 0.25° grid to match the GLORYS12 1/12° grid, and resampled to match the GLORYS12 daily temporal resolution. This is achieved through spatial linear interpolation and aggregation from an hourly to a daily resolution using a simple mean. The input sequence length is 10 days and the spatial domain as a grid is 229×361. Since the model operates over the mesh domain rather than the grid domain, the dimensionality of the inputs to the encoder as (input steps, number of nodes, input features) is $10 \times 9,422 \times 10$ for the irregular mesh and $10 \times 2,425 \times 10$ for the regular mesh. The input to the decoder is the context vectors provided by the encoder as well as the climatology for each forecast day. The output dimensionality is $90 \times 9,422 \times 2$ for the irregular mesh, and $90 \times 2,425 \times 2$ for the regular mesh.

Table 3.1: Selected input variables to the encoder, data source and rationale for inclusion.

Short Name	Full Name	Source	Rationale for Inclusion
`sic`	Sea ice concentration	GLORYS12	Direct measure of what is being forecasted; crucial for temporal dynamics and initial conditions.
`sit`	Sea ice thickness	GLORYS12	Provides insights into the resiliency and robustness of the ice, affecting its likelihood to melt or deform.
`siuv`	Sea ice velocities	GLORYS12	Indicates the direction and speed of sea ice movement.
`so`	Sea water salinity	GLORYS12	Salinity affects the freezing point of sea water and is crucial in the dynamics of ice formation and melt.
`sst`	Sea surface temperature	GLORYS12	The temperature of surrounding sea water directly affects ice melt and formation rates.
`t2m`	2-meter temperature	ERA5	Air temperature can provide additional context for the thermal conditions affecting the sea ice surface.
`u10/v10`	10-meter wind velocity	ERA5	Influences the motion and deformation of sea ice.
`sshf`	Surface sensible heat flux	ERA5	Surface sensible heat flux is an indicator of the heat exchange between the atmosphere and the sea surface, affecting ice melt and formation.
`x`	x-position of each node	—	Provides the latitudinal spatial context for each data point.
`y`	y-position of each node	—	Provides the longitudinal spatial context for each data point.
`doy`	Day of the year	—	Provides temporal context.
`csize`	Cell size	—	Provides the relative size of the area covered by each cell for additional spatial context.

3.6.4 Baseline Model

As a baseline model with which to compare the model, a combination of two common statistical baselines is used: persistence (P) and climatology (C). Persistence refers to persisting the most recent sea ice conditions and tends to perform well at very short forecast lengths particularly outside of the freezing and melting seasons. Climatology refers to the pixel-wise average SIC for each day of the year where the average is taken over the historical period of interest. Climatology tends to perform best relative to forecast models at longer lead times. For forecasts produced over a seasonal scale, a stronger baseline than either persistence and climatology can be derived by combining the two using a weighted average with the relative weights varying by lead time, where more weight is given to persistence than climatology at short lead times and more weight is given to climatology than persistence at long lead times. The form chosen for the baseline model is

$$F = (1 - \gamma)P + \gamma C, \tag{3.7}$$

where

$$\gamma(t) = \gamma_0 \times e^{-\lambda t}. \tag{3.8}$$

γ_0 is set to 1 since we know persistence to be a strong predictor at short lead times, and λ is optimized by minimizing the mean squared error over the training dataset for each month. The resulting weights are shown as a heatmap in Figure 3.8.

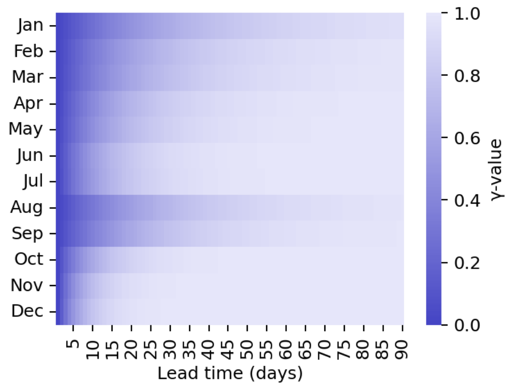


Figure 3.8: Gamma values for the baseline model (Equation 3.7) showing the balance between persistence and climatology by launch date month and lead time. Gamma values near 0 favor persistence while values near 1 favor climatology. Less variable ice seasons such as Jan/Feb and Aug/Sep rely more on persistence for longer lead times.

3.6.5 Model Hyperparameters and Implementation

This thesis evaluates three distinct models, listed in Table 3.2. The primary focus is the GraphSIFNet-Att model, which incorporates three TransformerConv spatial convolutions in the GCLSTM block and is trained on the irregular mesh described in section 3.2 for 35 epochs. That is, in Figure 3.6, $*\mathcal{G}$ uses the TransformerConv as the GraphConv block with $K = 3$. For comparison, the GraphSIFNet-Att-Reg model is examined, which is identical in architecture but trained on the coarsened regular mesh from section 3.2 for 35 epochs. Additionally, these are contrasted with the GraphSIFNet-GCN model, which employs six GCN convolutions within the GCLSTM module, that is, the GraphConv block is the GCN with $K = 6$. GraphSIFNet-GCN is trained over the irregular mesh for 45 epochs. Each of these models have the same number of parameters (approximately 123,000).

Each model uses a 10-day input sequence to predict the subsequent 90 days. A hidden dimension size of 32 is used for each of the hidden state and cell state of the encoder and decoder LSTMs, as well as in all graph convolutional layers. The GNN used to encode climatology (GNN_{enc}) is comprised of a single graph convolution layer, and the output GNN (GNN_{out}) is comprised of 3 stacked convolution layers with leaky ReLU activations. The hidden size, number of spatial convolutions and number of GCLSTM/LSTM layers were chosen based on small-scale experiments which aimed to keep the model simple yet effective. The optimizer is the Adam optimizer with an initial learning rate of 0.001 reducing by 10% every 5 epochs. An L2 regularization value of 0.01 is applied to the weights reduce the risk of overfitting, and gradient clipping with a value of 1.0 is applied to mitigate the risk of gradient explosion due to the extended forecast length. Early stopping was used if no improvement in the validation loss was observed for 10 epochs. Since the model produces two outputs, a custom loss function was used that combines a mean square error (MSE) loss from the continuous SIC prediction and binary cross-entropy (BCE) loss from the probabilistic SIP prediction. The BCE loss is scaled by a factor of 0.1 and added to the MSE loss before back-propagation. Since losses are calculated over a mesh with cells of varying physical sizes, the losses are also scaled by the size of each cell. This prevents the model from over-valuing correct predictions in areas of higher spatial resolution. The models are implemented in Pytorch using the pytorch-geometric [16] package and trained on a single Tesla V100 GPU hosted by the Digital Research Alliance of Canada. A summary of models tested and training times is given in Table 3.2.

Table 3.2: Summary of developed model configurations. The models differ in their spatial convolutions and their underlying meshes, with the aim of contrasting the attention-based transformer convolution with the graph convolutional network, as well as demonstrating the model's ability to model over an irregular mesh.

Name	**Convolution (# stacked layers)**	**Mesh**	**Approximate training time**
`GraphSIFNet-Att`	TransformerConv (3)	Irregular ($1/12°$ - $1/3°$)	10h (30 epochs)
`GraphSIFNet-Att-Reg`	TransformerConv (3)	Regular ($1/3°$)	8h (30 epochs)
`GraphSIFNet-GCN`	GCN (6)	Irregular ($1/12°$ - $1/3°$)	10h (45 epochs)
`Baseline (Equation 3.7)`	N/A	N/A	N/A

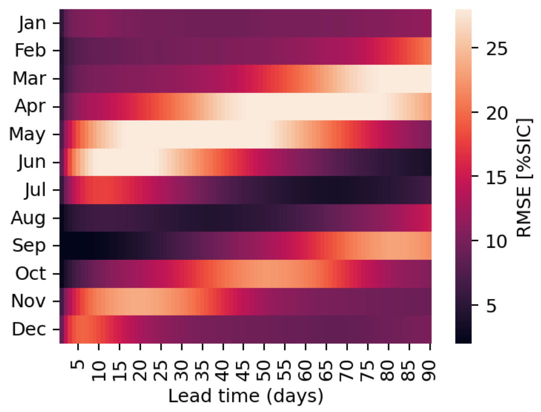
Chapter 4

Results

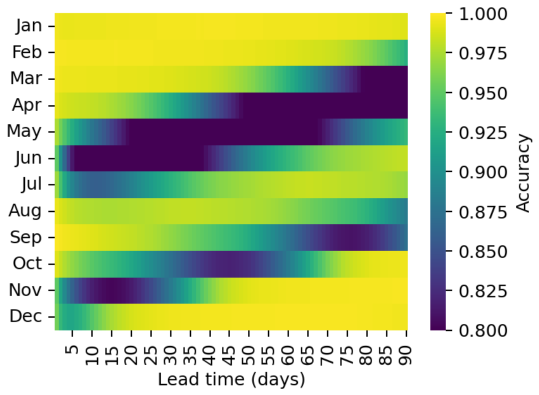
In this section, the GraphSIFNet-Att model is evaluated by comparing its performance with the statistical baseline and contrasting with the two other configurations: GraphSIFNet-Att-Reg and GraphSIFNet-GCN. Using GraphSIFNet-Att, insights from the attention weights, the results of a variable importance experiment, and an evaluation of its ability to predict break-up and freeze-up dates are also presented.

4.1 Baseline Performance

The performance of the baseline statistical model defined by Equation 3.7 for both the SIC and SIP forecasting task is shown in Figure 4.1a and Figure 4.1b, respectively. These heatmaps are generated by calculating the spatial average of the root mean squared error (RMSE) over the domain using only the test years (2016-2019). The errors are grouped by the month of the launch dates and lead times. For instance, the value in the top right corner of the error heatmaps (January, 90-day lead time) indicates the mean RMSE for all 90-day forecasts launched in January, that is, forecasts for dates spanning April 1st to May 1st. The two clearly visible bands of higher RMSE values correspond to the break-up and freeze-up seasons, the former normally spanning from the beginning of May to mid-July and the latter normally spanning from the beginning of November to the end of December. These seasons are the most difficult to forecast as the timing and pattern of the break-up and freeze-up vary between years. Conversely, August to beginning of October are largely ice-free, thus the errors are near zero. In the winter months, that is, mid-December to the beginning of April, ice is present throughout the Hudson Bay system though some open water can sporadically be found around shorelines, for example due to offshore winds, thus SIC RMSE values during the winter months are small but not zero.



(a) Baseline SIC performance



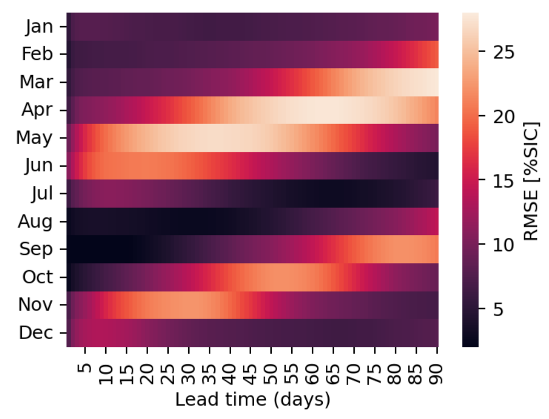
(b) Baseline SIP performance

Figure 4.1: Performance of the baseline statistical model on SIC (a) and SIP (b) over the test years aggregated by the month of the launch date and lead time.

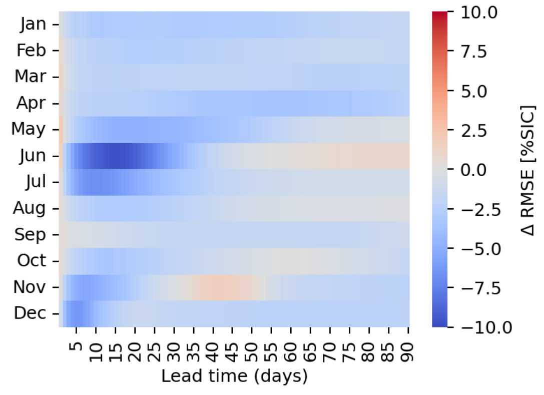
4.2 GraphSIFNet-Att Performance

The performance of GraphSIFNet-Att model and the difference in performance relative to the baseline model is shown in Figure 4.2 and Figure 4.3 for SIC and SIP forecasts, respectively. Since persistence and climatology are usually used as baselines seperately, the difference in performance relative to both are shown in Appendix A. Models are evaluated against GLORYS12 SIC and SIP on the full-resolution 1/12° GLORYS12 grid.

For the majority of the months and lead times, the GraphSIFNet-Att model exhibits improvements in SIC forecasts over the baseline, with minor exceptions. The model exhibits the largest improvements over the baseline in its short- to medium-term forecasts of the break-up season (lead times 5 to 45 launched in May to July). These show up to a 10% improvement over the baseline. At longer timesteps, the improvements over the baseline during the break-up period (launched in March and April) are less pronounced, hovering around 2-3%. However at these long lead times even small improvements demonstrate forecast skill and can provide value to users of the system. During the winter months when the region is almost entirely frozen, the model still exhibits a 2-3% improvement over the baseline at all lead times. This suggests that the model may be able to better capture

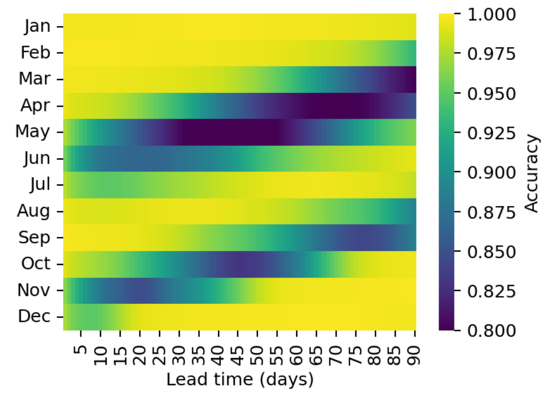


(a) GraphSIFNet-Att

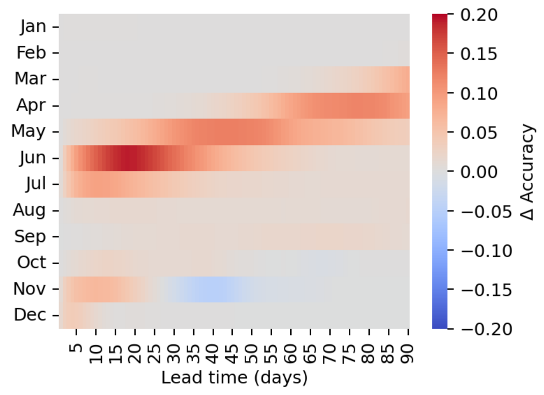


(b) Δ(GraphSIFNet-Att, Baseline)

Figure 4.2: RMSE heatmaps for the SIC forecasting task by month and lead time for the GraphSIFNet-Att model (a), and the RMSE differences between GraphSIFNet-Att and the baseline (b) where negative values (blue) indicate a reduction in model error relative to the baseline.



(a) GraphSIFNet-Att



(b) Δ(GraphSIFNet-Att, Baseline)

Figure 4.3: Accuracy heatmaps for the SIP forecasting task by month and lead time for the GraphSIFNet-Att model (a), and the difference between GraphSIFNet-Att and the baseline (b) where positive values (red) in the difference plots indicate an increase in model accuracy relative to the baseline.

the effects of off-shore winds mechanically creating open water regions along the shoreline. During freeze-up, the model only shows skill over the baseline at short lead times from

0 to 25 days. Longer forecasts beyond 25 days perform on par with the baseline or only marginally better. Forecasts launched in November with a 30 to 55 day lead time perform worse than the baseline, indicating difficulty in capturing the final stages of ice formation.

The SIP accuracy heatmaps in Figure 4.3 show similar patterns, with increases in accuracy of up to 20% from the GraphSIFNet-Att model over the baseline during the break-up process, and more modest increases during the freeze-up process. Notably, however, GraphSIFNet-Att outperforms quite significantly (> 10%) even at long lead times. This indicates that although the model may struggle to forecast the precise SIC at these lead times, it still has skill in forecasting the point at which the ice will completely melt or break up (i.e. SIC dropping below 15%).

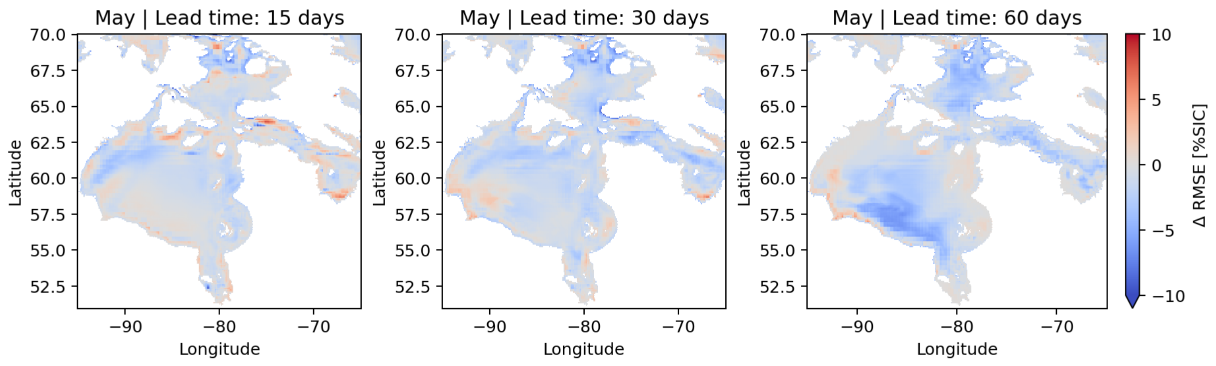
4.3 Comparison Between Model Configurations

Differences in both SIC RMSE and SIP accuracy between the GraphSIFNet model configurations, averaged across all timesteps for each month, are shown in Figure 4.4. The GraphSIFNet-GCN and GraphSIFNet-Att-Reg models demonstrate comparable performance relative to GraphSIFNet-Att, with differences being largely insignificant when aggregated across the entire region. To better understand the differences in their capabilities, spatial monthly SIC RMSE maps for the 15-, 30-, and 60-day lead times for forecasts launched in May and November are presented in Figure 4.5 and 4.6. These correspond to parts of the break-up and freeze-up periods, respectively.

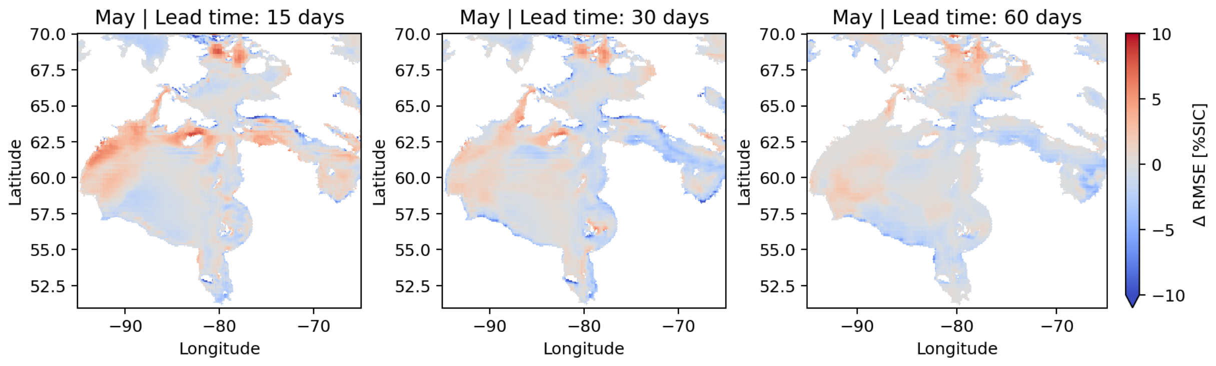
Task	Model	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Avg.
SIC	GraphSIFNet-GCN	0.29	0.19	0.02	0.11	0.67	-0.33	0.18	0.22	0.00	0.11	-0.37	0.40	0.12
	GraphSIFNet-Att-Reg	0.43	0.12	-0.16	0.30	-0.19	-0.71	0.22	0.03	-0.08	1.19	-0.79	0.51	0.07
SIP	GraphSIFNet-GCN	0.00	-0.01	0.05	0.14	-0.54	0.08	-0.06	-0.07	0.37	0.13	-0.38	-0.03	-0.03
	GraphSIFNet-Att-Reg	0.00	-0.03	0.08	-0.22	-0.10	0.25	-0.10	0.02	0.18	-0.21	0.83	-0.15	0.05

Figure 4.4: Difference in monthly SIC RMSE [%SIC] and SIP [%]accuracy between GraphSIFNet-Att-Reg and GraphSIFNet-GCN relative to GraphSIFNet-Att averaged over all 90 day forecasts launched in the given month. Negative RMSE differences and positive accuracy differences indicate better performance on the part of GraphSIFNet-Att relative to the other models.

Early (15-day) forecasts in the Northwest region of Hudson Bay, launched in May, are best captured by GraphSIFNet-Att-Reg. This region is characterized by a latent heat



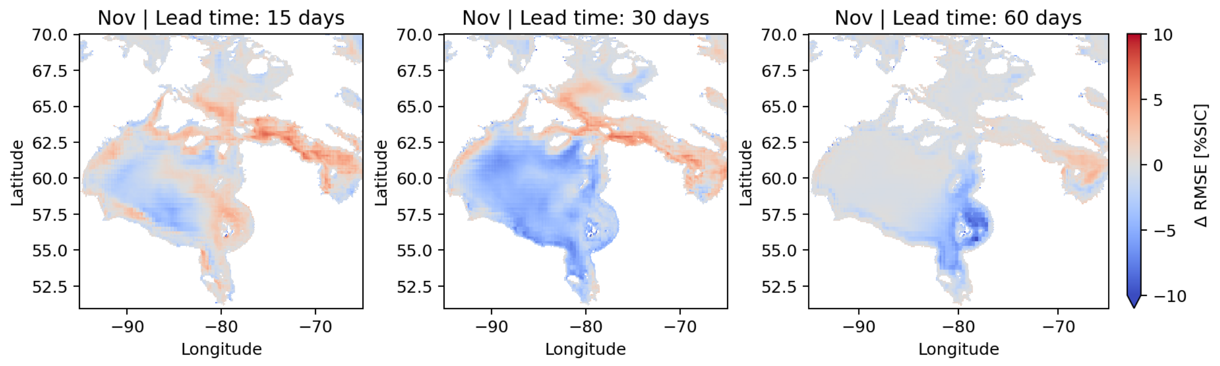
(a) May — Δ(GraphSIFNet-Att, GraphSIFNet-GCN)



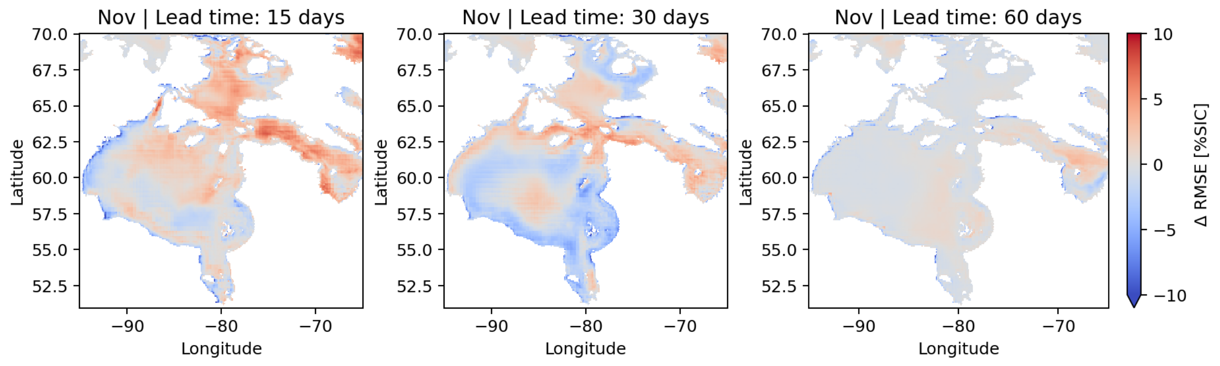
(b) May — Δ(GraphSIFNet-Att, GraphSIFNet-Att-Reg)

Figure 4.5: Comparison of SIC RMSE for GraphSIFNet-Att, GraphSIFNet-Att-Reg, and GraphSIFNet-GCN models at 15-, 30-, and 60-day forecast lead times, initiated in May (break-up). The figure shows the difference in RMSE between GraphSIFNet-Att and both GraphSIFNet-Att-Reg and GraphSIFNet-GCN. Subplot (a) shows the impact of the different graph convolution, while subplot (b) shows the impact of the different meshes. Negative values indicate a reduction in error in the GraphSIFNet-Att relative to the other indicated model.

polynya, suggesting that the coarser uniform resolution mesh may aid the model in forecasting the formation and behavior of the polynya. Using a finer resolution mesh in this region might cause the model to overemphasize local variations in sea ice concentration and thickness, potentially obscuring the broader spatial patterns crucial for accurate polynya forecasting. Both GraphSIFNet-GCN and GraphSIFNet-Att-Reg outperform GraphSIFNet-Att in the 15- and 30-day forecasts launched in November in Hudson Strait. The freeze-up



(a) November — Δ(GraphSIFNet-Att, GraphSIFNet-GCN)



(b) November — Δ(GraphSIFNet-Att, GraphSIFNet-Att-Reg)

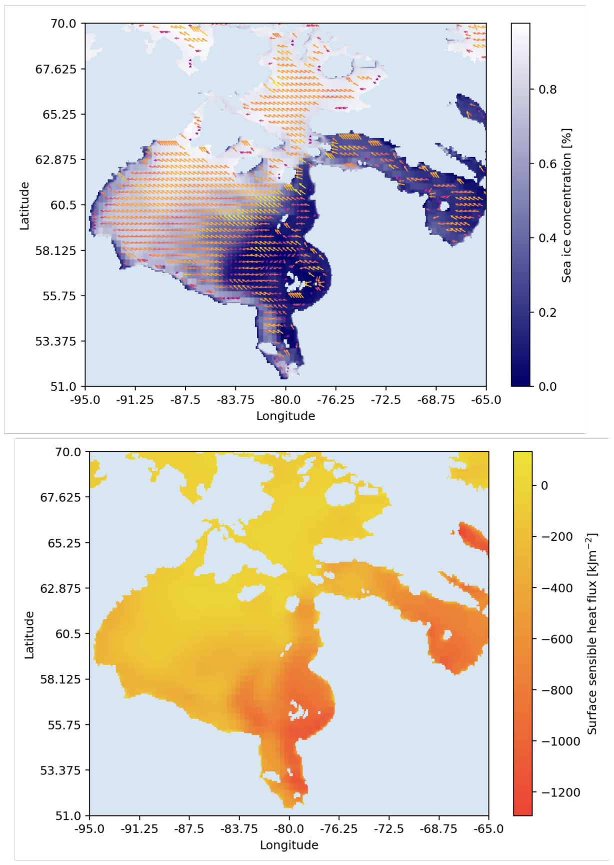
Figure 4.6: Comparison of SIC RMSE for GraphSIFNet-Att, GraphSIFNet-Att-Reg, and GraphSIFNet-GCN models at 15-, 30-, and 60-day forecast lead times, initiated in November (freeze-up). The figure shows the difference in RMSE between GraphSIFNet-Att and both GraphSIFNet-Att-Reg and GraphSIFNet-GCN. Subplot (a) shows the impact of the different graph convolution, while subplot (b) shows the impact of the different meshes. Negative values indicate a reduction in error in the GraphSIFNet-Att relative to the other indicated model.

in Hudson Strait is characterized by rapid changes in ice formation and movement influenced by strong ocean currents. These conditions create a highly dynamic and challenging environment for sea ice prediction. Since all three models exhibit similar performance, the additional interpretability granted by the attention weights in GraphSIFNet-Att motivates the use of GraphSIFNet-Att over the others.

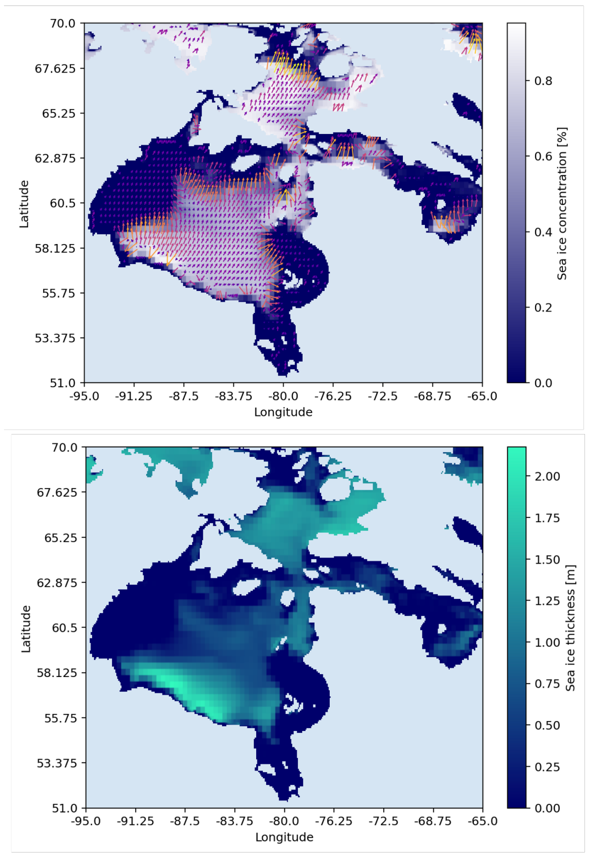
4.4 Attention Maps

The use of transformer convolution in the model enhances its interpretability. By examining the attention weights in the encoder's first layer of graph convolutions, insights can be gleaned into how the model encodes the input data. According to Equation 3.4 and Equation 3.5, each node is assigned attention weights for its neighboring nodes based on learned weight matrices in each transformer layer. The softmax function ensures that the sum of all attention weights for a given node's neighbors equals 1. Consequently, the node is updated using a weighted average of its neighbors' features, which are projected into a latent space. Due to the large number of edges, visualizing these weights on a simple map is challenging. A simpler approach for visualization involves calculating the primary direction from which each node is updated. This can be done by summing the attention weights as vectors (α values in Equation 3.4 with the direction of their respective edges) for each node. These can be represented by arrows, the magnitude of which is proportional to the difference in weights. For example, a node with evenly distributed attention weights among eight neighbors would be represented as a single dot (indicating the node does not preferentially attend to any given neighbour), whereas a node with a dominant westward neighbor would have a large arrow pointing westward. These arrows can be interpreted as indicating the direction of information flow through the graph as the model processes the input maps.

Figure 4.7 provides examples of attention weights of the input gate for a single input image during both freeze-up (Figure 4.8a) and melting (Figure 4.8b) seasons. Although the attention mechanism is applied to the hidden and input tensors at each of the LSTM gates, it is most informative to visualize the weights that are applied to the inputs since the inputs are physically interpretable. Note that attention weights at land interfaces are omitted for visual clarity, as they are numerous and the lack of nodes on land means the dominant direction is always away from the shore. In the freeze-up condition, the model directs information flow generally from the southeast to the northwest. This suggests that the model learns the importance of understanding the sea ice and atmospheric conditions of nodes to the northwest, aligning with the direction of freezing. It it is logical that a node that contains water should know the condition of its 3-hop neighbor to the northwest, as if this neighbor is frozen, it is likely that this node will freeze in the near future. Conversely, during the melting season, arrows point towards open water, indicating that nodes with icy conditions but with water-containing neighbors should consider these neighbors important as they indicate the node is likely to melt soon. Notably, the magnitude of the arrows is larger at the ice edge and nearly zero in the consolidated ice region, which could reflect the localized nature of the break-up process compared to the more gradual freeze-up. That is,


Latitude
Longitude
Sea ice concentration [%]
Surface sensible heat flux [kJm^{-2}]

(a) December 1, 2014



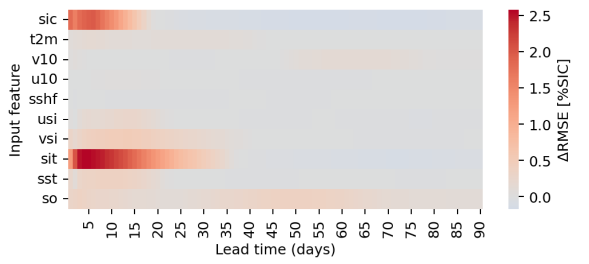
(b) July 1, 2014

Figure 4.7: Visualization of attention weights of the input gate applied to the input tensors during the freeze-up (a) and melting (b) seasons overlaid on the sea ice concentration input. Arrows indicate the primary direction and magnitude of information flow based on the learned attention weights. Attention weights at the land interfaces are omitted for clarity. The attention weights appear to be largely influenced by sea ice concentration, but other input variables also influence the weights, for example surface sensible heat flux in (a), and sea ice thickness in (b).

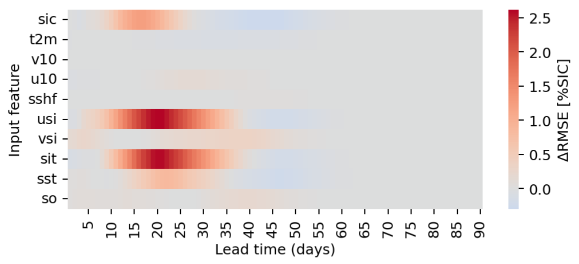
the break-up process is largely confined to the ice edge, while freeze-up gradually occurs across the region, as seen by changes in sea ice concentration. Nodes in the open water region during the melting season are less likely to change and, therefore, do not require attention to specific neighbors. Note that although the weights are visualized on the sea ice concentration inputs, they apply indiscriminately to all input features. Interestingly, the model appears to prioritize sea ice thickness over concentration, evidenced by the larger attention weights where thickness drops more dramatically than concentration in Figure 4.8b. This is logical given the importance of thickness in determining the rate at which the ice will melt or break up. Additionally, the attention weights in the open-water region during the freeze-up condition appear to be influenced by surface sensible heat flux, suggesting its significance as an input feature.

4.5 Variable Importance

The models are trained with a number of input variables (refer to Table 3.1), which were anticipated to be used by the model to make its predictions. However, these variables may not contribute equally to the resulting predictions. In this section, the significance of each feature is explored by feature ablation through omission [17]. Specifically, forecasts are produced using the trained GraphSIFNet-Att model by substituting each input variable, one at a time, with white Gaussian noise generated using the mean and standard deviation of the real inputs. By replacing these inputs without re-training the model, we can assess the degree of the model's dependence on each respective input. Little or no change in performance after noise injection suggests the model does not consider the input when producing forecasts, while a large change would suggest the inverse. Figure 4.8 shows the resulting difference in RMSE when re-generating predictions on the test years using the June and December models when each variable is replaced with noise.



(a) June model feature importance



(b) November model feature importance

Figure 4.8: Feature ablation with noise injection for the June and November GraphSIFNet-Att models. Positive values indicate an increase in RMSE when each respective variable is replaced with noise.

During the break-up process (June model), the model largely relies on the input sea ice concentration and sea ice thickness to make its predictions, but also considers the ice velocities, sea surface temperature and sea salinity to a smaller degree. Other variables do not significantly affect the resulting predictions. The model appears to use sea ice concentrations to inform near-term forecasts (days 0 through 20), and sea ice thickness to inform its medium-term forecasts (days 0 through 35). This makes intuitive sense as thickness is an indicator of the ice cover's longevity making it relevant at longer forecast steps, while sea ice concentration is more important for immediate predictions since lower ice concentrations are normally associated with ice parcels that are already breaking up. Note that at forecast steps larger than 35 days, forecasts launched in June are largely forecasting periods where Hudson Bay is fully open water, thus none of the input features contribute to the resulting forecasts.

Similarly, during the freeze-up process, the model relies on sea ice thickness, sea ice concentration, sea ice velocity and sea surface temperature to make its predictions. Again, the model largely considers sea ice concentration to make its shorter term forecasts (days 10 through 25), while considering ice velocity and thickness for medium-term forecasts (days 15 through 40). Ice velocity may be indicating areas where ice migrates, thereby creating space for new ice formation. The difference between the vertical and horizontal ice velocity component (*usi* and *vsi*) may indicate that they offer redundant information, thus it is sufficient for the model to consider one of the components. Again, November forecasts at larger than 40 days are largely forecasting periods of full ice cover, therefore omitting input features does not impact the scores. It is also worth noting that in both cases, the model does not appear to consider the variables originating from ERA5. This could point to a mismatch between ERA5 and GLORYS12, which would be unsurprising as GLORYS12 uses ERA-Interim as model forcing at the surface. Since the target variables are derived from GLORYS12, the models therefore prioritize input features originating from GLORYS12.

To illustrate the impact of these variables on the predictions, a sample GraphSIFNet-Att forecast is shown in Figure 4.9, along with the same forecast when replacing sea ice concentration and sea ice thickness (SIT) with noise as described above. Replacing either SIC or SIT with noise does not significantly affect the 1-day forecast, suggesting the model uses persistence as a heuristic at very short lead times. Beyond the 10-day forecast, predictions are affected by the noise injections, with the model forecasting a quicker melt when sea ice thickness is replaced with noise, consistent with the theory that thickness is used as a signal of ice longevity. When SIC is replaced with noise, the model persists more of the ice in the 20-day forecast, suggesting that SIC is also important for ice integrity.

Although this technique offers some insight into feature importance, it should be noted that since the models are not re-trained, the observed changes in performance due to feature omission may not perfectly reflect the true importance of each feature. This is because the

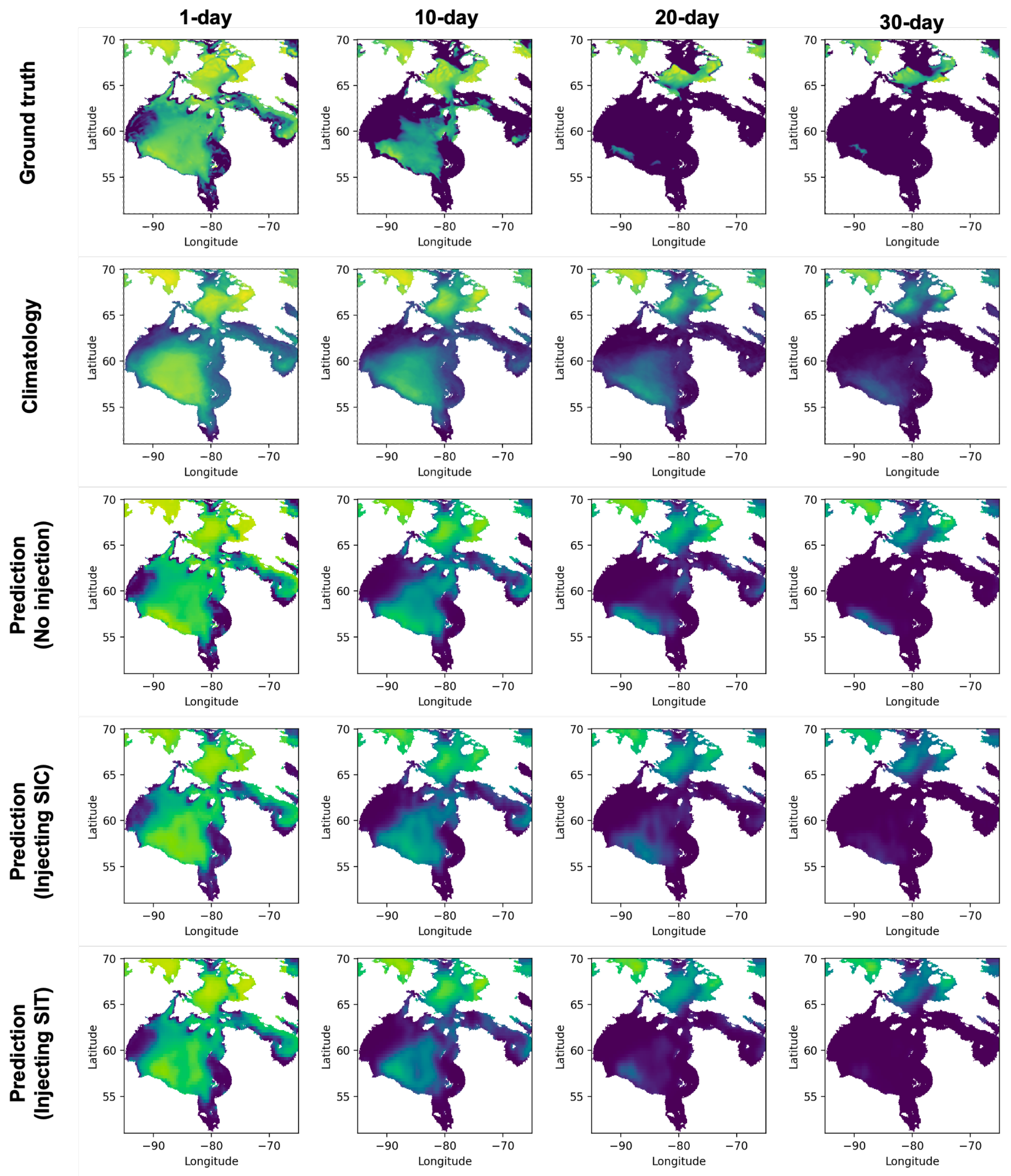


Figure 4.9: Sample 1-, 10-, 20-, and 30-day forecasts from GraphSIFNet-Att launched on June 15, 2014. The climatology for each forecast day is shown for reference, and the results of running inference after replacing sea ice concentration (SIC) and sea ice thickness (SIT) with noise is shown.

model has been optimized to make predictions based on the full set of features, therefore the omission of any one feature changes the input space in a way that the model was not specifically trained to handle. Moreover, the interdependencies between features are not accounted for in this single-feature ablation approach. Variables in the dataset may interact in complex, non-linear ways that are not captured by examining each variable in isolation. Despite these limitations, this feature ablation technique provides useful insights into the relative importance of the different input features used in these particular trained models [17]. Since we know which features the models are using, we know which input variables should be more closely monitored.

4.6 Estimating Break-up and Freeze-up Dates

A potential use-case for sea ice prediction in Hudson Bay is the estimation of break-up and freeze-up dates in key locations, as these dates have significant implications for maritime navigation and local communities. The GraphSIFNet-Att model's performance in estimating the freeze-up date at three key ports in Hudson Bay is evaluated: the ports of Churchill, Quaqtaq and Inukjuak. The port of Churchill is mostly used to export grain while the ports of Quaqtaq and Inukjuak are regularly used for community resupply. These three ports were chosen as their locations are representative of the varying sea ice conditions found in the Hudson Bay region. In this thesis, the validation and test years (2014 to 2019) serve as the period for assessing the predicted break-up and freeze-up dates. These dates are determined using the same criteria as the previous study, which follows the definition given by the Canadian Ice Service (CIS). That is, the freeze-up date at a given site is defined as the initial day when open water persists for 15 consecutive days, with open water being defined as a SIC of less than 15%. Conversely, the break-up date is defined as first day at which SIC exceeds 15% for 15 consecutive days. The 30-day and 60-day predicted break-up and freeze-up dates are determined using the same criteria, but with open water and ice conditions being defined as a sea ice presence probability less than and greater than 50%, respectively. For each port, the mean pixel value of a 3×3 window around the nearest pixel to the port locations is taken.

Figure 4.10 and Figure 4.11 display the predicted dates of freeze-up/breakup at the three ports with 30 and 60 days of lead time compared to the actual observed dates for the validation and test years along with the mean absolute error. Predicted dates falling within 7 days of the observed dates are considered correct, visualized by the pink shaded area. This definition of a correct forecast is in line with a previous study [4]. The 30-day forecasted break-up and freeze-up dates for Churchill are noticeably inferior to the other

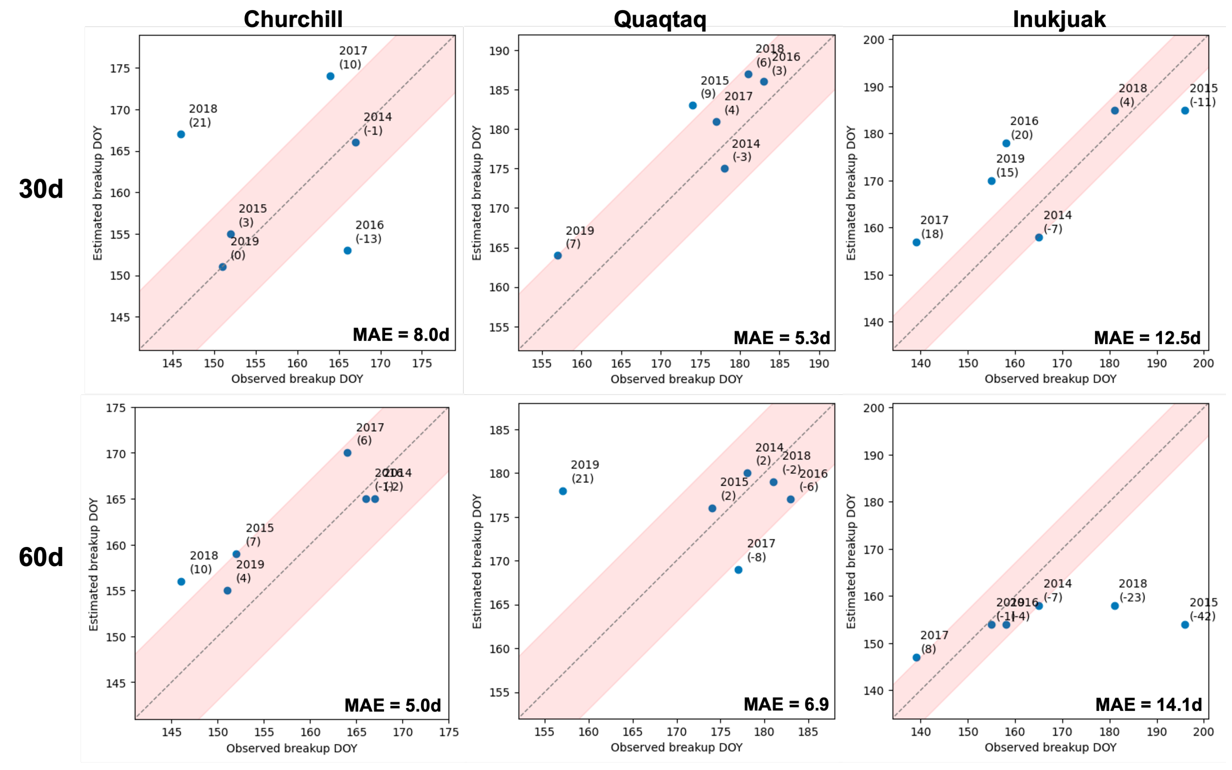


Figure 4.10: Break-up dates predicted by GraphSIFNet-Att at Churchill, Inukjuak, and Quaqtaq ports for lead times of 30 and 60 days for the years 2014 to 2019 compared to the observed dates from GLORYS12. The pink shaded area represents a 7-day buffer around a perfect forecast. Samples which fall within this buffer are deemed correct forecasts. The annotated numbers in parentheses are the error for each year.

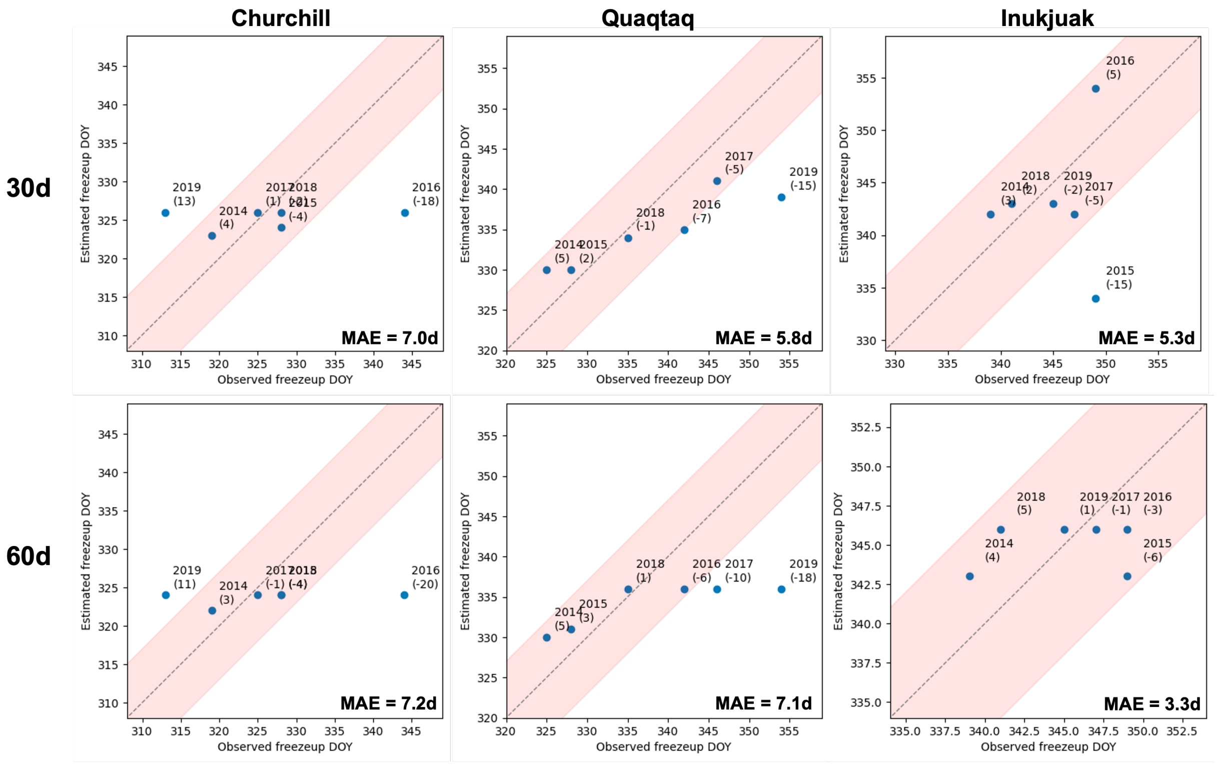
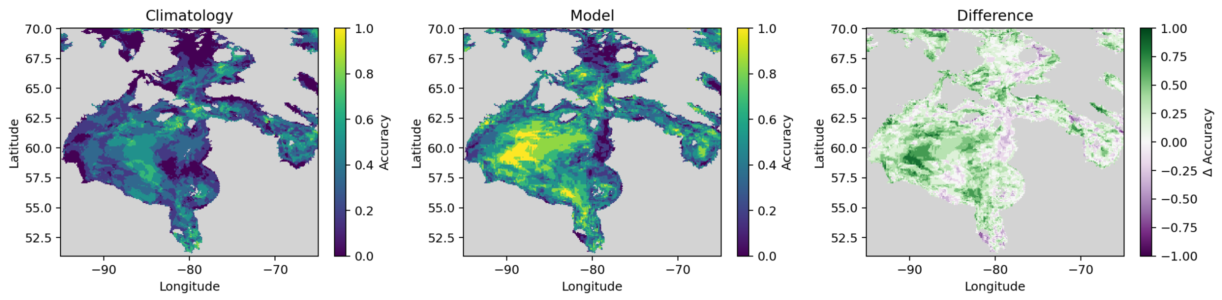


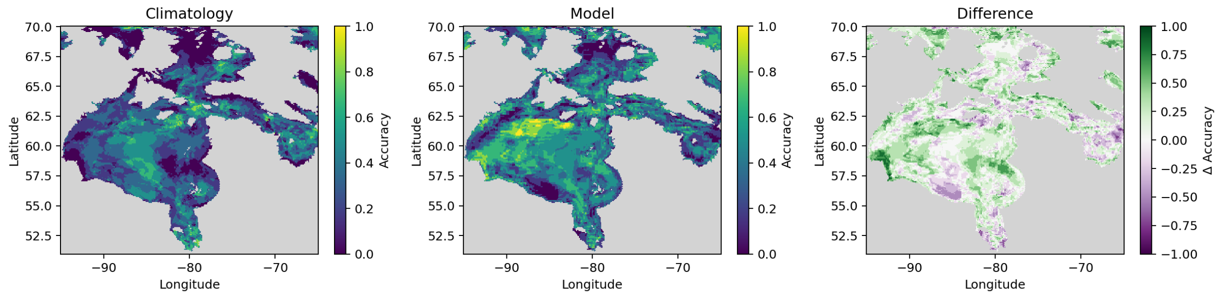
Figure 4.11: Freeze-up dates predicted by GraphSIFNet-Att at Churchill, Inukjuak, and Quaqtaq ports for lead times of 30 and 60 days for the years 2014 to 2019 compared to the observed dates from GLORYS12. The pink shaded area represents a 7-day buffer around a perfect forecast. Samples which fall within this buffer are deemed correct forecasts. The annotated numbers in parentheses are the error for each year.

two ports, likely due to challenges presented by the latent heat polynya in the Northwest of Hudson Bay. The uniform forecasts of freeze-up dates at Churchill can be interpreted as an admission that the model does not have skill here and resorts to forecasting the same mean day every year. Break-up predictions at Inukjuak also pose a challenge for the model, likely due to freshwater inflows from the James Bay area affecting the timing and rate of melt. Quaqtaq sees the most successful predictions, with all freeze-up dates falling within 7 days of the observed date.

In Figure 4.12 and Figure 4.13, the break-up and freeze-up accuracies are shown spatially for the entire region. These accuracies are calculated as the proportion of years with predicted break-up or freeze-up dates within 7 days of the observed date. These are compared to predictions made using the climate normals. The model performs equally or better than climatology for most of the region in predicting break-up dates at both 30-days and 60-days of lead time. However, there is a strong pattern in the freeze-up maps where the model performs worse than climatology in the western half of the bay but still outperforms climatology in the eastern half and in Hudson Strait. This is unsurprising as Hudson Bay begins its freeze-up process in the northwest corner of the bay, thus the onset of that initial freezing is difficult to predict. Once the bay has begun freezing over, the model can better predict the timing of the rest of the bay. Although we might expect the model to use atmospheric conditions such as temperature to predict the onset of freeze-up, the model only has access those atmospheric conditions 30 or 60 days prior to the forecast date. There may not be a strong enough signal in those initial conditions to allow the model to accurately predict how quickly the temperatures will drop.

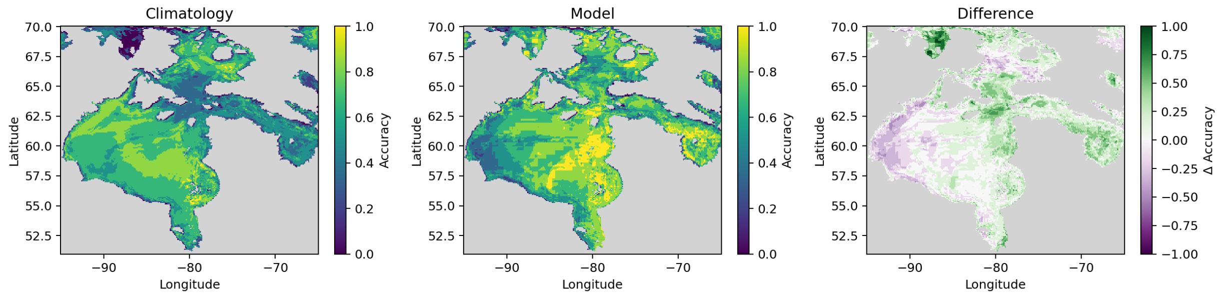


(a) 30-day break-up date estimate map

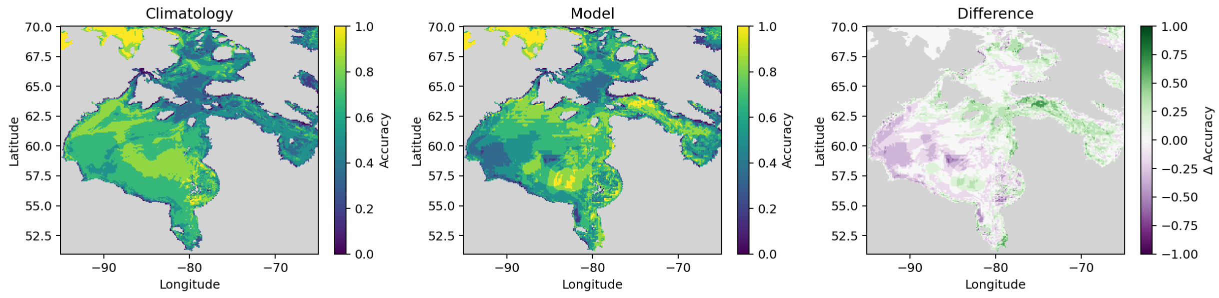


(b) 60-day break-up date estimate map

Figure 4.12: Break-up date estimate maps from the climatological baseline (left), GraphSIFNet-Att model predictions (middle), and the difference between the two (right). Positive values in the difference plots indicate an increase in accuracy from the model relative to the baseline, where accuracy is defined as the proportion of predictions falling within 7 days of the observed date for the years 2014 to 2019.



(a) 30-day freeze-up date estimate map



(b) 60-day freeze-up date estimate map

Figure 4.13: Freeze-up date estimate maps from the climatological baseline (left), GraphSIFNet-Att model predictions (middle), and the difference between the two (right). Positive values in the difference plots indicate an increase in accuracy from the model relative to the baseline, where accuracy is defined as the proportion of predictions falling within 7 days of the observed date for the years 2014 to 2019.

Chapter 5

Conclusion

This thesis demonstrated the effectiveness of using a GNN-based spatiotemporal forecasting model for predicting daily sea ice concentration and sea ice presence in Hudson Bay over a 90-day time horizon. To demonstrate the ability of GNNs to handle spatially irregular meshes, models were trained on both a uniform regular mesh and an irregular mesh with higher resolution near shorelines. The proposed model uses an attention-based transformer spatial convolution to learn spatial features from the input, which was shown to have similar performance compared to the more basic graph convolutional network. The attention-based convolution however has the additional benefit of increasing the model's interpretability, motivating its use.

Results from this work highlighted the model's skill in predicting sea ice dynamics, with particular success noted in short- to medium-term forecasts during the break-up season when compared to a linear combination of persistence and climatology as a statistical baseline. The model performed as well or better on the irregular mesh as on the regular mesh, with the exception of some difficulty capturing the initial freeze-up in the Northwest region of Hudson Bay as well as the polynya formation at longer lead times. This suggests that improvements could be made in refining the model's sensitivity to complex spatial features associated with irregular meshes, particularly in areas where ice dynamics are highly variable. This could involve more sophisticated positional and spatial encoding, perhaps by projecting the positional, cell size, distance and angle encodings into higher dimensional latent space. The model showed similar overall performance between the model using the transformer convolution and the GCN within the GCLSTM module, with some differences in performance in certain regions such as Hudson Strait. This suggested potential overfitting in the model using the spatial transformer convolution.

The attention mechanism within the transformer convolution offered interpretability by highlighting the primary direction and magnitude of information flow in the encoder, which aligned with known physical processes such as the direction of freezing and melting. A feature ablation experiment indicated the trained model's reliance on sea ice concentration, thickness and velocities to inform its predictions. Other variables did not contribute significantlty to the resulting forecasts, which could explain the model's poor performance in forecasting the Kivalliq latent heat polynya. A evaluation of the model's ability to predict freeze-up and break-up dates was conducted, revealing the model's limited ability to forecast the onset of freeze-up in Hudson Bay, as well as the onset of break-up in the Northwest region which is influenced by the polynya. The model however still showed skill over the statistical baseline in these tasks.

5.1 Future Work

This work demonstrated the effectiveness of temporal GNNs in forecasting sea ice. However, several potential avenues exist to extend or improve upon this work.

Dynamic Re-Meshing

The experiments conducted in this thesis were done using a static spatial mesh. However, the graph-based model presented here theoretically allows for modelling over a mesh that is dynamic in time, evolving as the underlying data changes (e.g., as the ice conditions evolve). Since each node has its own hidden and cell states, cells can be combined by averaging the states or split by duplicating the states. Thus, the latent representation of a given snapshot can be re-meshed by using Equation 3.2 and Equation 3.1 given an initial mesh defined by the mapping tensor M_1 and a target mapping tensor M_2 (along with the corresponding tensors P_1 and P_2. These operations being entirely formulated as matrix operations allows for GPU acceleration and for the operations to be more easily differentiable for backpropagation.

For example, one could define a dynamic mesh which has a higher resolution at the ice edge where the ice conditions are known to be more dynamic. As the ice conditions evolve, so too would the underlying mesh. The advantages are two-fold. First, it allows for a reduction in data volume with minimal information loss, contrary to the static mesh used in this work which has information loss where the data has high spatial variance. Second, the dynamic mesh could potentially help the model learn more sophisticated dynamics and is more consistent with some physical simulation software. The idea of dynamically re-meshed GNNs for physical simulations was explored in [54].

Training over such a dynamic mesh was initially explored as a part of this thesis, but was ultimately removed as the computation required proved infeasible on the available computational resources when training over the full training set. Further investigation of this technique to improve its efficiency could yield better results, however. This could consist of investigating the computational graph more thoroughly to find inefficiencies, re-structuring the training pipeline to use multiple GPUs, or simplifying the model further.

To demonstrate the feasibility of dynamic re-meshing in a spatiotemporal forecasting model, experiments were conducted on a toy dataset which yielded promising results. The results of this experiment are shown in Appendix B.

Physics-informed Learning

Deep learning models can often benefit from a more direct incorporation of known physical laws when modeling physical systems. While the expectation is that a deep learning model will inherently learn the necessary physical processes, training a model using the standard backpropagation procedure does not guarantee this. Furthermore, verifying what the model has learned remains challenging. Incorporating physical equations, either within the network architecture or as part of the training process, can help generate more physically realistic forecasts.

Take, for instance, the simple ice growth model given by

$$h_{\text{ice}} = \frac{Q_{\text{net}} \cdot \Delta t}{\rho_{\text{ice}} \cdot L_f} \tag{5.1}$$

where h_{ice} is the change in ice thickness, Q_{net} represents the net energy flux, ρ_{ice} denotes the density of sea ice, L_f is the latent heat of fusion, and Δt is the time step length (here, one day). Re-arranged to isolate Q_net, this becomes

$$Q_{\text{net}} = \frac{h_{\text{ice}} \cdot \rho_{\text{ice}} \cdot L_f}{\Delta t}. \tag{5.2}$$

This equation could be used to estimate the net energy flux required to produce the change in ice thickness forecasted by the model, and ensure that it falls within a reasonable range, which could be determined using historical records from ERA5 for each time of the year. If the required net flux is found to be unreasonable, a penalty term could be added to the loss. This is just one example of a physical equation that could be used to constrain the model outputs in different ways. Many other equations could be used to constrain different parts of the system (for instance, conservation of mass using ice velocity fields, or more sophisticated thermodynamic equations).

Alternatively, physical equations could be incorporated by directly building them into the neural network architecture. For instance, sea ice models are sometimes evaluated by calculating an "ice budget" over the domain to verify that the net budget is near zero [52]. Take the equation for sea ice intensification [26]

$$\frac{\partial C}{\partial t} + \nabla \cdot (uC) = f_c - r \tag{5.3}$$

where $\frac{\partial C}{\partial t}$ representations ice intensification (the change in sea ice concentration over time), C represents sea ice concentration, f_c denotes thermodynamic freezing or melt, and r denotes mechanical redistribution (ice parcels ridging or rafting). The $\nabla \cdot (uC)$ can be separated into its advection and divergence terms and re-written as

$$\frac{\partial C}{\partial t} = -u \cdot \nabla C - C \nabla \cdot u + \text{residual} = A + D + residual \tag{5.4}$$

where A and D are the advection and divergence terms, respectively, and the residual term combine both f_c and r. This decomposition can be used directly within a deep learning model by having the model predict all of the three constituent terms (advection, convergence, and residual) and calculating the loss with respect to these three terms. To generate a sea ice concentration forecast, the three terms are summed, and added to the sea ice concentrations of the previous day. Note that to accomplish this, the three terms above need to be estimated from some sea ice dataset. This can be done using sea ice velocity fields in the GLORYS12 dataset used in this work on a pixel-wise basis using the equations:

$$A_{ij} = -v^x_{i,j} \cdot \left[\frac{(C_{i+1,j} - C_{i-1,j})}{2 \cdot (x_{i+1,j} - x_{i-1,j})}\right] - v^y_{i,j} \cdot \left[\frac{(C_{i,j+1} - C_{i,j-1})}{2 \cdot (y_{i,j+1} - y_{i,j-1})}\right] \tag{5.5}$$

$$D_{ij} = -C^x_{i,j} \cdot \left[\frac{(v_{i+1,j} - v_{i-1,j})}{2 \cdot (x_{i+1,j} - x_{i-1,j})}\right] - C^y_{i,j} \cdot \left[\frac{(v_{i,j+1} - v_{i,j-1})}{2 \cdot (y_{i,j+1} - y_{i,j-1})}\right] \tag{5.6}$$

for given pixel indices (i, j), where C represents sea ice concentration, and v^x and v^y are the eastward and northward ice velocities. While this training strategy was not

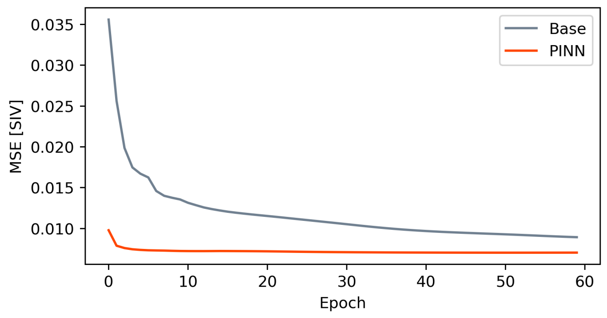


Figure 5.1: Comparing the test loss curve when training the April model to predict sea ice volume directly (Base) versus predicting the advection, divergence and residual terms separately (PINN).

implemented in this study, preliminary small-scale experiments forecasting sea ice volume (the product of sea ice concentration and thickness) using this strategy yielded promising results, showing faster convergence and a lower error (see Figure 5.1).

Improving the Ice Edge Definition

Probabilistic spatial ice models such as the one employed in this work often lead to the generation of blurred edges in the modeled ice distributions. This effect arises because these models are optimized to minimize error across observed and predicted sea ice concentrations. In the process of error minimization, the model inherently averages out the sharp transitions between ice and open water, resulting in a smoothed gradient rather than a distinct boundary. Real-world sea ice edges, however, are often sharp and clearly defined due to the physical processes governing ice formation and melt. A possible remedy to this issue is the use a perceptual loss function that would penalize forecasts that *look* different from the ground truth due to differences in texture or structure. For example, the structural similarity index measure (SSIM) quantifies the difference in luminance, contrast, and structure between images. For two images x and y, SSIM is given by the equation

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{5.7}$$

where μ_x and μ_y are the average intensities of x and y, σ_x^2 and σ_y^2 are the variances of x and y, σ_{xy} is the covariance of x and y. $c_1 = (k_1 L)^2$ and $c_2 = (k_2 L)^2$ are factors used for numerical stability with L being the dynamic range of the images, and k_1 and k_2 are parameters typically set to 0.01 and 0.03, respectively. Incorporating SSIM or similar measures into the model's loss function could encourage the preservation of distinct ice-water boundaries, thereby generating more physically feasible forecasts, though perhaps reducing the accuracies calculated by metrics such as MSE.

Alternatively, different modelling techniques such as GANs, which are trained with the goal of generating predictions which are indistinguishable from real observations. This will inherently generate more feasible ice edges as blurred edges would be an obvious sign that a sample is a prediction, easily identifiable by the discriminator.

Efficiency

In this thesis, a model was trained over Hudson Bay, a relatively small domain compared to the broader pan-Arctic domain. The main computational limitation was the memory usage, as during each update step, the entire graph is processed at once. For a larger domain at the same high resolution, it may not be feasible to train the model in the same way. Fortunately, in a GNN, each node is processed as a separate sample by the network. This means that one input sample $Y \in \mathbb{R}^{N \times C}$ does not necessarily need to be processed fully at once, instead, nodes could also be sampled in batches sequentially until the full sample has been processed. Alternatively, each epoch could randomly sample a limited number of nodes for training, though this risks divergence during training if the samples are too dissimilar.

Other Improvements

Another avenue for future work could be a deeper investigation of the adjacency matrix. In this work, edges were placed between any two directly spatially adjacent cells. However, edges could also be placed between distant cells thereby widening the receptive field without adding convolutions. This could be investigated by transforming the adjacency matrix into a learnable matrix optimized during training. Limits would likely need to be imposed, either with a maximum number of edges or a maximum edge length, to ensure computational feasibility.

Incorporating long-term weather forecasts from third party sources such as the Canadian Global Ice Ocean Prediction System (GIOPS) could also be beneficial, particularly in forecasting freeze-up. These forecasts could be used, for example, to inform the model as to whether it is likely to be a warmer or cooler winter season, or whether the timing of sub-zero temperatures in the fall is likely to be earlier or later than usual. Integrating these forecasts would require careful consideration to ensure the forecasts are being effectively considered, without dominating over the model itself.

Lastly, multi-resolution modelling either through an ensemble of models operating over meshes of different resolution or using multiple meshes of varying resolutions within a single model could be explored. This may help the model better capture both large-scale and small-scale phenomena.

References

[1] Sahara Ali, Yiyi Huang, Xin Huang, and Jianwu Wang. Sea Ice Forecasting using Attention-based Ensemble LSTM. *arXiv:2108.00853 [physics]*, February 2022. URL http://arxiv.org/abs/2108.00853. arXiv: 2108.00853.

[2] Tom R. Andersson, J. Scott Hosking, María Pérez-Ortiz, Brooks Paige, Andrew Elliott, Chris Russell, Stephen Law, Daniel C. Jones, Jeremy Wilkinson, Tony Phillips, James Byrne, Steffen Tietsche, Beena Balan Sarojini, Eduardo Blanchard-Wrigglesworth, Yevgeny Aksenov, Rod Downie, and Emily Shuckburgh. Seasonal Arctic sea ice forecasting with probabilistic deep learning. *Nature Communications*, 12(1):5124, August 2021. ISSN 2041-1723. doi: 10.1038/s41467-021-25257-4. URL https://www.nature.com/articles/s41467-021-25257-4. Number: 1 Publisher: Nature Publishing Group.

[3] Luc Anselin. A Typology of Spatial Econometric Models. In Luc Anselin, editor, *Spatial Econometrics: Methods and Models*, Studies in Operational Regional Science, pages 32–40. Springer Netherlands, Dordrecht, 1988. ISBN 978-94-015-7799-1. doi: 10.1007/978-94-015-7799-1_4. URL https://doi.org/10.1007/978-94-015-7799-1_4.

[4] Nazanin Asadi, Philippe Lamontagne, Matthew King, Martin Richard, and K. Andrea Scott. Probabilistic spatiotemporal seasonal sea ice presence forecasting using sequence-to-sequence learning and ERA5 data in the Hudson Bay region. *The Cryosphere*, 16(9):3753–3773, September 2022. ISSN 1994-0424. doi: 10.5194/tc-16-3753-2022. URL https://tc.copernicus.org/articles/16/3753/2022/.

[5] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization, July 2016. URL http://arxiv.org/abs/1607.06450. arXiv:1607.06450 [cs, stat].

[6] Khac-Hoai Nam Bui, Jiho Cho, and Hongsuk Yi. Spatial-temporal graph neural network for traffic forecasting: An overview and open research issues. *Applied Intelligence*,

52(3):2763–2774, February 2022. ISSN 1573-7497. doi: 10.1007/s10489-021-02587-w. URL https://doi.org/10.1007/s10489-021-02587-w.

[7] Roberto Cahuantzi, Xinye Chen, and Stefan Güttel. A Comparison of LSTM and GRU Networks for Learning Symbolic Sequences. In Kohei Arai, editor, *Intelligent Computing*, volume 739, pages 771–785. Springer Nature Switzerland, Cham, 2023. ISBN 978-3-031-37962-8 978-3-031-37963-5. doi: 10.1007/978-3-031-37963-5_53. URL https://link.springer.com/10.1007/978-3-031-37963-5_53. Series Title: Lecture Notes in Networks and Systems.

[8] D. J. Cavalieri and C. L. Parkinson. Arctic sea ice variability and trends, 1979–2010. *The Cryosphere*, 6(4):881–889, August 2012. ISSN 1994-0416. doi: 10.5194/tc-6-881-2012. URL https://tc.copernicus.org/articles/6/881/2012/. Publisher: Copernicus GmbH.

[9] Junhwa Chi and Hyun-choel Kim. Prediction of Arctic Sea Ice Concentration Using a Fully Data Driven Deep Neural Network. *Remote Sensing*, 9(12):1305, December 2017. ISSN 2072-4292. doi: 10.3390/rs9121305. URL https://www.mdpi.com/2072-4292/9/12/1305. Number: 12 Publisher: Multidisciplinary Digital Publishing Institute.

[10] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches, October 2014. URL http://arxiv.org/abs/1409.1259. arXiv:1409.1259 [cs, stat].

[11] K. Ch. Das. The Laplacian spectrum of a graph. *Computers & Mathematics with Applications*, 48(5):715–724, September 2004. ISSN 0898-1221. doi: 10.1016/j.camwa.2004.05.005. URL https://www.sciencedirect.com/science/article/pii/S0898122104003074.

[12] Richard A. Davis, Pengfei Zang, and Tian Zheng. Sparse Vector Autoregressive Modeling, July 2012. URL http://arxiv.org/abs/1207.0520. arXiv:1207.0520 [stat].

[13] D. P. Dee, S. M. Uppala, A. J. Simmons, P. Berrisford, P. Poli, S. Kobayashi, U. Andrae, M. A. Balmaseda, G. Balsamo, P. Bauer, P. Bechtold, A. C. M. Beljaars, L. van de Berg, J. Bidlot, N. Bormann, C. Delsol, R. Dragani, M. Fuentes, A. J. Geer, L. Haimberger, S. B. Healy, H. Hersbach, E. V. Hólm, L. Isaksen, P. Kållberg, M. Köhler, M. Matricardi, A. P. McNally, B. M. Monge-Sanz, J.-J. Morcrette, B.-K. Park, C. Peubey, P. de Rosnay, C. Tavolato, J.-N. Thépaut, and F. Vitart. The ERA-Interim reanalysis: configuration and performance of the data assimilation system. *Quarterly Journal of the Royal Meteorological Society*, 137(656):553–597, 2011. ISSN 1477-870X. doi: 10.1002/qj.

828. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/qj.828. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/qj.828.

[14] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/hash/04df4d434d481c5bb723be1b6df1ee65-Abstract.html.

[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, June 2021. URL http://arxiv.org/abs/2010.11929. arXiv:2010.11929 [cs].

[16] Matthias Fey and Jan Eric Lenssen. Fast Graph Representation Learning with PyTorch Geometric, April 2019. URL http://arxiv.org/abs/1903.02428. arXiv:1903.02428 [cs, stat].

[17] Ruth C. Fong and Andrea Vedaldi. Interpretable Explanations of Black Boxes by Meaningful Perturbation. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3449–3457, Venice, October 2017. IEEE. ISBN 978-1-5386-1032-9. doi: 10.1109/ICCV.2017.371. URL http://ieeexplore.ieee.org/document/8237633/.

[18] Felix Gers, Nicol Schraudolph, and Jürgen Schmidhuber. Learning Precise Timing with LSTM Recurrent Networks. *Journal of Machine Learning Research*, 3:115–143, January 2002. doi: 10.1162/153244303768966139.

[19] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5): 602–610, July 2005. ISSN 0893-6080. doi: 10.1016/j.neunet.2005.06.042. URL https://www.sciencedirect.com/science/article/pii/S0893608005001206.

[20] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html.

[21] William L. Hamilton. The Graph Neural Network Model. In William L. Hamilton, editor, *Graph Representation Learning*, Synthesis Lectures on Artificial Intelligence

and Machine Learning, pages 51–70. Springer International Publishing, Cham, 2020. ISBN 978-3-031-01588-5. doi: 10.1007/978-3-031-01588-5_5. URL https://doi.org/10.1007/978-3-031-01588-5_5.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Las Vegas, NV, USA, June 2016. IEEE. ISBN 978-1-4673-8851-1. doi: 10.1109/CVPR.2016.90. URL http://ieeexplore.ieee.org/document/7780459/.

[23] Mingguo He, Zhewei Wei, and Ji-Rong Wen. Convolutional Neural Networks on Graphs with Chebyshev Approximation, Revisited, December 2022. URL http://arxiv.org/abs/2202.03580. arXiv:2202.03580 [cs].

[24] Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz Sabater, Julien Nicolas, Carole Peubey, Raluca Radu, Dinand Schepers, Adrian Simmons, Cornel Soci, Saleh Abdalla, Xavier Abellan, Gianpaolo Balsamo, Peter Bechtold, Gionata Biavati, Jean Bidlot, Massimo Bonavita, Giovanna Chiara, Per Dahlgren, Dick Dee, Michail Diamantakis, Rossana Dragani, Johannes Flemming, Richard Forbes, Manuel Fuentes, Alan Geer, Leo Haimberger, Sean Healy, Robin J. Hogan, Elías Hólm, Marta Janisková, Sarah Keeley, Patrick Laloyaux, Philippe Lopez, Cristina Lupu, Gabor Radnoti, Patricia Rosnay, Iryna Rozum, Freja Vamborg, Sebastien Villaume, and Jean Noel Thépaut. The ERA5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049, July 2020. ISSN 0035-9009, 1477-870X. doi: 10.1002/qj.3803. URL https://onlinelibrary.wiley.com/doi/10.1002/qj.3803.

[25] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-term Memory. *Neural computation*, 9:1735–80, December 1997. doi: 10.1162/neco.1997.9.8.1735.

[26] Paul R. Holland. The seasonality of Antarctic sea ice trends. *Geophysical Research Letters*, 41(12):4230–4237, 2014. ISSN 1944-8007. doi: 10.1002/2014GL060172. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/2014GL060172. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/2014GL060172.

[27] Elizabeth Hunke and W. Lipscomb. CICE: The Los Alamos sea ice model documentation and software user's manual version 4.0 LA-CC-06-012. *Tech. Rep. LA-CC-06-012*, January 2010.

[28] Elizabeth Hunke, Richard Allard, Philippe Blain, Ed Blockley, Daniel Feltham, Thierry Fichefet, Gilles Garric, Robert Grumbine, Jean-François Lemieux, Till Rasmussen, Mads Ribergaard, Andrew Roberts, Axel Schweiger, Steffen Tietsche, Bruno Tremblay, Martin Vancoppenolle, and Jinlun Zhang. Should Sea-Ice Modeling Tools Designed for Climate Research Be Used for Short-Term Forecasting? *Current Climate Change Reports*, 6(4):121–136, December 2020. ISSN 2198-6061. doi: 10.1007/s40641-020-00162-y. URL https://doi.org/10.1007/s40641-020-00162-y.

[29] Vladimir V. Ivanov, Vladimir A. Alexeev, Irina Repina, Nikolay V. Koldunov, and Alexander Smirnov. Tracing Atlantic Water Signature in the Arctic Sea Ice Cover East of Svalbard. *Advances in Meteorology*, 2012:e201818, August 2012. ISSN 1687-9309. doi: 10.1155/2012/201818. URL https://www.hindawi.com/journals/amete/2012/201818/. Publisher: Hindawi.

[30] Lellouche Jean-Michel, Greiner Eric, Bourdallé-Badie Romain, Garric Gilles, Melet Angélique, Drévillon Marie, Bricaud Clément, Hamon Mathieu, Le Galloudec Olivier, Regnier Charly, Candela Tony, Testut Charles-Emmanuel, Gasparin Florent, Ruggiero Giovanni, Benkiran Mounir, Drillet Yann, and Le Traon Pierre-Yves. The Copernicus Global 1/12° Oceanic and Sea Ice GLORYS12 Reanalysis. *Frontiers in Earth Science*, 9, 2021. ISSN 2296-6463. URL https://www.frontiersin.org/articles/10.3389/feart.2021.698876.

[31] Weiwei Jiang and Jiayun Luo. Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications*, 207:117921, November 2022. ISSN 0957-4174. doi: 10.1016/j.eswa.2022.117921. URL https://www.sciencedirect.com/science/article/pii/S0957417422011654.

[32] Atharva Joshi, Pradeep K. Deshmukh, and Jay Lohokare. Comparative analysis of Vanilla LSTM and Peephole LSTM for stock market price prediction. In *2022 International Conference on Computing, Communication, Security and Intelligent Systems (IC3SIS)*, pages 1–6, June 2022. doi: 10.1109/IC3SIS54991.2022.9885528.

[33] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, March 1960. ISSN 0021-9223. doi: 10.1115/1.3662552. URL https://doi.org/10.1115/1.3662552.

[34] Ibrahem Kandel and Mauro Castelli. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT Express*, 6(4):312–315, December 2020. ISSN 2405-9595. doi: 10.1016/j.icte.2020.04.010. URL https://www.sciencedirect.com/science/article/pii/S2405959519303455.

[35] Ryan Keisler. Forecasting Global Weather with Graph Neural Networks, February 2022. URL http://arxiv.org/abs/2202.07575. Number: arXiv:2202.07575 arXiv:2202.07575 [physics].

[36] Jiwon Kim, Kwangjin Kim, Jaeil Cho, Yong Q. Kang, Hong-Joo Yoon, and Yang-Won Lee. Satellite-Based Prediction of Arctic Sea Ice Concentration Using a Deep Neural Network with Multi-Model Ensemble. *Remote Sensing*, 11(1):19, January 2019. ISSN 2072-4292. doi: 10.3390/rs11010019. URL https://www.mdpi.com/2072-4292/11/1/19. Number: 1 Publisher: Multidisciplinary Digital Publishing Institute.

[37] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, January 2017. URL http://arxiv.org/abs/1412.6980. arXiv:1412.6980 [cs].

[38] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks, February 2017. URL http://arxiv.org/abs/1609.02907. arXiv:1609.02907 [cs, stat].

[39] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Alexander Pritzel, Suman Ravuri, Timo Ewalds, Ferran Alet, Zach Eaton-Rosen, Weihua Hu, Alexander Merose, Stephan Hoyer, George Holland, Jacklynn Stott, Oriol Vinyals, Shakir Mohamed, and Peter Battaglia. GraphCast: Learning skillful medium-range global weather forecasting, December 2022. URL http://arxiv.org/abs/2212.12794. arXiv:2212.12794 [physics].

[40] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998. ISSN 1558-2256. doi: 10.1109/5.726791. URL https://ieeexplore.ieee.org/document/726791. Conference Name: Proceedings of the IEEE.

[41] Ming Li, Ren Zhang, and Kefeng Liu. Machine Learning Incorporated With Causal Analysis for Short-Term Prediction of Sea Ice. *Frontiers in Marine Science*, 8, 2021. ISSN 2296-7745. URL https://www.frontiersin.org/articles/10.3389/fmars.2021.649378.

[42] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12):6999–7019, December 2022. ISSN 2162-2388. doi: 10.1109/TNNLS.2021.3084827. URL https://ieeexplore.ieee.org/abstract/document/9451544. Conference Name: IEEE Transactions on Neural Networks and Learning Systems.

[43] Zachary C. Lipton, John Berkowitz, and Charles Elkan. A Critical Review of Recurrent Neural Networks for Sequence Learning, October 2015. URL http://arxiv.org/abs/1506.00019. arXiv:1506.00019 [cs].

[44] Guanjun Liu, Shuo Ouyang, Hui Qin, Shuai Liu, Qin Shen, Yuhua Qu, Zhiwei Zheng, Huaiwei Sun, and Jianzhong Zhou. Assessing spatial connectivity effects on daily streamflow forecasting using Bayesian-based graph neural network. *Science of The Total Environment*, 855:158968, January 2023. ISSN 0048-9697. doi: 10.1016/j.scitotenv.2022.158968. URL https://www.sciencedirect.com/science/article/pii/S0048969722060673.

[45] Yang Liu, Laurens Bogaardt, Jisk Attema, and Wilco Hazeleger. Extended Range Arctic Sea Ice Forecast with Convolutional Long-Short Term Memory Networks. *Monthly Weather Review*, 149, March 2021. doi: 10.1175/MWR-D-20-0113.1.

[46] Yongqi Liu, Guibing Hou, Feng Huang, Hui Qin, Baohua Wang, and Ling Yi. Directed graph deep neural network for multi-step daily streamflow forecasting. *Journal of Hydrology*, 607:127515, April 2022. ISSN 0022-1694. doi: 10.1016/j.jhydrol.2022.127515. URL https://www.sciencedirect.com/science/article/pii/S0022169422000907.

[47] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the Effective Receptive Field in Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/hash/c8067ad1937f728f51288b3eb986afaa-Abstract.html.

[48] Gurvan Madec. NEMO ocean engine.

[49] Walter N. Meier, Greta K. Hovelsrud, Bob E.H. van Oort, Jeffrey R. Key, Kit M. Kovacs, Christine Michel, Christian Haas, Mats A. Granskog, Sebastian Gerland, Donald K. Perovich, Alexander Makshtas, and James D. Reist. Arctic sea ice in transformation: A review of recent observed changes and impacts on biology and human activity. *Reviews of Geophysics*, 52 (3):185–217, 2014. ISSN 1944-9208. doi: 10.1002/2013RG000431. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/2013RG000431. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/2013RG000431.

[50] Twila A. Moon, Irina Overeem, Matt Druckenmiller, Marika Holland, Henry Huntington, George Kling, Amy Lauren Lovecraft, Gifford Miller, Ted Scambos, Christina

Schädel, Edward A. G. Schuur, Erin Trochim, Francis Wiese, Dee Williams, and Gifford Wong. The Expanding Footprint of Rapid Arctic Change. *Earth's Future*, 7(3):212–218, 2019. ISSN 2328-4277. doi: 10.1029/2018EF001088. URL https://onlinelibrary.wiley.com/doi/abs/10.1029/2018EF001088. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2018EF001088.

[51] Lawrence Mudryk, Jackie Dawson, Stephen Howell, Chris Derksen, Thomas Zagon, and Mike Brady. Impact of 1, 2 and 4 °C of global warming on ship navigation in the Canadian Arctic. *Nature Climate Change*, 11:1–7, August 2021. doi: 10.1038/s41558-021-01087-6.

[52] Yafei Nie, Petteri Uotila, Bin Cheng, François Massonnet, Noriaki Kimura, Andrea Cipollone, and Xianqing Lv. Southern Ocean sea ice concentration budgets of five ocean-sea ice reanalyses. *Climate Dynamics*, April 2022. ISSN 0930-7575, 1432-0894. doi: 10.1007/s00382-022-06260-x. URL https://link.springer.com/10.1007/s00382-022-06260-x.

[53] Sergiu Oprea, Pablo Martinez-Gonzalez, Alberto Garcia-Garcia, John Alejandro Castro-Vargas, Sergio Orts-Escolano, Jose Garcia-Rodriguez, and Antonis Argyros. A Review on Deep Learning Techniques for Video Prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(6):2806–2826, June 2022. ISSN 1939-3539. doi: 10.1109/TPAMI.2020.3045007. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[54] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning Mesh-Based Simulation with Graph Networks. October 2020. URL https://openreview.net/forum?id=roNqYL0_XP.

[55] Phillip E. Pfeifer and Stuart Jay Deutsch. A STARIMA Model-Building Procedure with Application to Description and Regional Forecasting. *Transactions of the Institute of British Geographers*, 5(3):330, 1980. ISSN 00202754. doi: 10.2307/621846. URL https://www.jstor.org/stable/621846?origin=crossref.

[56] Eric Post, Uma S. Bhatt, Cecilia M. Bitz, Jedediah F. Brodie, Tara L. Fulton, Mark Hebblewhite, Jeffrey Kerby, Susan J. Kutz, Ian Stirling, and Donald A. Walker. Ecological Consequences of Sea-Ice Decline. *Science*, 341(6145):519–524, August 2013. doi: 10.1126/science.1235225. URL https://www.science.org/doi/full/10.1126/science.1235225. Publisher: American Association for the Advancement of Science.

[57] Pei Quan, Yong Shi, Minglong Lei, Jiaxu Leng, Tianlin Zhang, and Lingfeng Niu. A Brief Review of Receptive Fields in Graph Convolutional Networks. In *IEEE/WIC/ACM International Conference on Web Intelligence - Companion Volume*, WI '19 Companion, pages 106–110, New York, NY, USA, October 2019. Association for Computing Machinery. ISBN 978-1-4503-6988-6. doi: 10.1145/3358695.3360934. URL https://doi.org/10.1145/3358695.3360934.

[58] Pierre Rampal, Sylvain Bouillon, Einar Ólason, and Mathieu Morlighem. neXtSIM: a new Lagrangian sea ice model. *The Cryosphere*, 10(3):1055–1073, May 2016. ISSN 1994-0416. doi: 10.5194/tc-10-1055-2016. URL https://tc.copernicus.org/articles/10/1055/2016/. Publisher: Copernicus GmbH.

[59] Yulia Rubanova, Alvaro Sanchez-Gonzalez, Tobias Pfaff, and Peter Battaglia. Constraint-based graph network simulator. In *Proceedings of the 39th International Conference on Machine Learning*, pages 18844–18870. PMLR, June 2022. URL https://proceedings.mlr.press/v162/rubanova22a.html. ISSN: 2640-3498.

[60] Sebastian Ruder. An overview of gradient descent optimization algorithms, June 2017. URL http://arxiv.org/abs/1609.04747. arXiv:1609.04747 [cs].

[61] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to Simulate Complex Physics with Graph Networks. In *Proceedings of the 37th International Conference on Machine Learning*, pages 8459–8468. PMLR, November 2020. URL https://proceedings.mlr.press/v119/sanchez-gonzalez20a.html. ISSN: 2640-3498.

[62] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Structured Sequence Modeling with Graph Convolutional Recurrent Networks. In Long Cheng, Andrew Chi Sing Leung, and Seiichi Ozawa, editors, *Neural Information Processing*, Lecture Notes in Computer Science, pages 362–373, Cham, 2018. Springer International Publishing. ISBN 978-3-030-04167-0. doi: 10.1007/978-3-030-04167-0_33.

[63] Mark C. Serreze and Walter N. Meier. The Arctic's sea ice cover: trends, variability, predictability, and comparisons to the Antarctic. *Annals of the New York Academy of Sciences*, 1436(1):36–53, 2019. ISSN 1749-6632. doi: 10.1111/nyas.13856. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/nyas.13856. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/nyas.13856.

[64] Alex Sherstinsky. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *Physica D: Nonlinear Phenomena*, 404:

132306, March 2020. ISSN 0167-2789. doi: 10.1016/j.physd.2019.132306. URL https://www.sciencedirect.com/science/article/pii/S0167278919305974.

[65] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun WOO. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/hash/07563a3fe3bbe7e3ba84431ad9d055af-Abstract.html.

[66] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun. Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 1548–1554, Montreal, Canada, August 2021. International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-9-6. doi: 10.24963/ijcai.2021/214. URL https://www.ijcai.org/proceedings/2021/214.

[67] Christopher A. Sims. Macroeconomics and Reality. *Econometrica*, 48(1):1–48, 1980. ISSN 0012-9682. doi: 10.2307/1912017. URL https://www.jstor.org/stable/1912017. Publisher: [Wiley, Econometric Society].

[68] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised Learning of Video Representations using LSTMs. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 843–852. PMLR, June 2015. URL https://proceedings.mlr.press/v37/srivastava15.html. ISSN: 1938-7228.

[69] S. E. Stammerjohn, M. R. Drinkwater, R. C. Smith, and X. Liu. Ice-atmosphere interactions during sea-ice advance and retreat in the western Antarctic Peninsula region. *Journal of Geophysical Research: Oceans*, 108(C10), 2003. ISSN 2156-2202. doi: 10.1029/2002JC001543. URL https://onlinelibrary.wiley.com/doi/abs/10.1029/2002JC001543. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2002JC001543.

[70] Julienne Stroeve and Dirk Notz. Changing state of Arctic sea ice across all seasons. *Environmental Research Letters*, 13(10):103001, September 2018. ISSN 1748-9326. doi: 10.1088/1748-9326/aade56. URL https://dx.doi.org/10.1088/1748-9326/aade56. Publisher: IOP Publishing.

[71] Julienne C. Stroeve, Mark C. Serreze, Marika M. Holland, Jennifer E. Kay, James Malanik, and Andrew P. Barrett. The Arctic's rapidly shrinking sea ice cover: a research synthesis. *Climatic Change*, 110(3):1005–1027, February 2012. ISSN

1573-1480. doi: 10.1007/s10584-011-0101-1. URL https://doi.org/10.1007/s10584-011-0101-1.

[72] Alexander Y. Sun, Peishi Jiang, Zong-Liang Yang, Yangxinyu Xie, and Xingyuan Chen. A graph neural network (GNN) approach to basin-scale river network learning: the role of physics-based connectivity and data fusion. *Hydrology and Earth System Sciences*, 26(19):5163–5184, October 2022. ISSN 1027-5606. doi: 10.5194/hess-26-5163-2022. URL https://hess.copernicus.org/articles/26/5163/2022/. Publisher: Copernicus GmbH.

[73] Chengcheng Sun, Chenhao Li, Xiang Lin, Tianji Zheng, Fanrong Meng, Xiaobin Rui, and Zhixiao Wang. Attention-based graph neural networks: a survey. *Artificial Intelligence Review*, 56(2):2263–2310, November 2023. ISSN 1573-7462. doi: 10.1007/s10462-023-10577-2. URL https://doi.org/10.1007/s10462-023-10577-2.

[74] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://papers.nips.cc/paper_files/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html.

[75] Ah Chung Tsoi. Recurrent neural network architectures: An overview. In C. Lee Giles and Marco Gori, editors, *Adaptive Processing of Sequences and Data Structures: International Summer School on Neural Networks "E.R. Caianiello" Vietri sul Mare, Salerno, Italy September 6–13, 1997 Tutorial Lectures*, Lecture Notes in Computer Science, pages 1–26. Springer, Berlin, Heidelberg, 1998. ISBN 978-3-540-69752-7. doi: 10.1007/BFb0053993. URL https://doi.org/10.1007/BFb0053993.

[76] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[77] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks, February 2018. URL http://arxiv.org/abs/1710.10903. arXiv:1710.10903 [cs, stat].

[78] Timo Vihma. Effects of Arctic Sea Ice Decline on Weather and Climate: A Review. *Surveys in Geophysics*, 35(5):1175–1214, September 2014. ISSN 1573-0956. doi: 10.1007/s10712-014-9284-0. URL https://doi.org/10.1007/s10712-014-9284-0.

[79] Frédéric Vitart and Andrew W. Robertson. The sub-seasonal to seasonal prediction project (S2S) and the prediction of extreme events. *npj Climate and Atmospheric Science*, 1(1):1–7, March 2018. ISSN 2397-3722. doi: 10.1038/s41612-018-0013-0. URL https://www.nature.com/articles/s41612-018-0013-0. Number: 1 Publisher: Nature Publishing Group.

[80] Caixin Wang, Robert M. Graham, Keguang Wang, Sebastian Gerland, and Mats A. Granskog. Comparison of ERA5 and ERA-Interim near-surface air temperature, snowfall and precipitation over Arctic sea ice: effects on sea ice thermodynamics and evolution. *The Cryosphere*, 13(6):1661–1679, June 2019. ISSN 1994-0416. doi: 10.5194/tc-13-1661-2019. URL https://tc.copernicus.org/articles/13/1661/2019/. Publisher: Copernicus GmbH.

[81] Senzhang Wang, Jiannong Cao, and Philip S. Yu. Deep Learning for Spatio-Temporal Data Mining: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 34(08):3681–3700, August 2022. ISSN 1041-4347. doi: 10.1109/TKDE.2020.3025580. URL https://www.computer.org/csdl/journal/tk/2022/08/09204396/1nkyTK35x1S. Publisher: IEEE Computer Society.

[82] Timothy Williams, Anton Korosov, Pierre Rampal, and Einar Ólason. Presentation and evaluation of the Arctic sea ice forecasting system neXtSIM-F. *The Cryosphere*, 15(7):3207–3227, July 2021. ISSN 1994-0416. doi: 10.5194/tc-15-3207-2021. URL https://tc.copernicus.org/articles/15/3207/2021/. Publisher: Copernicus GmbH.

[83] K. Worden, George Tsialiamanis, E. Cross, and Timothy Rogers. Artificial Neural Networks. pages 85–119. October 2023. ISBN 978-3-031-36643-7. doi: 10.1007/978-3-031-36644-4_2.

[84] Zhanghao Wu, Paras Jain, Matthew A. Wright, Azalia Mirhoseini, Joseph E. Gonzalez, and Ion Stoica. Representing Long-Range Context for Graph Neural Networks with Global Attention, January 2022. URL http://arxiv.org/abs/2201.08821. arXiv:2201.08821 [cs].

[85] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, January 2021. ISSN 2162-237X, 2162-2388. doi: 10.1109/TNNLS.2020.2978386. URL http://arxiv.org/abs/1901.00596. arXiv:1901.00596 [cs, stat].

[86] Xingbai Xu and Lung-fei Lee. A spatial autoregressive model with a nonlinear transformation of the dependent variable. *Journal of Econometrics*, 186(1):1–18, May 2015. ISSN 0304-4076. doi: 10.1016/j.jeconom.2014.12.005. URL https://www.sciencedirect.com/science/article/pii/S0304407614002954.

[87] T. Yao, C. L. Tang, and I. K. Peterson. Modeling the seasonal variation of sea ice in the Labrador Sea with a coupled multicategory ice model and the Princeton ocean model. *Journal of Geophysical Research: Oceans*, 105 (C1):1153–1165, 2000. ISSN 2156-2202. doi: 10.1029/1999JC900264. URL https://onlinelibrary.wiley.com/doi/abs/10.1029/1999JC900264. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/1999JC900264.

[88] Fisher Yu and Vladlen Koltun. Multi-Scale Context Aggregation by Dilated Convolutions, April 2016. URL http://arxiv.org/abs/1511.07122. arXiv:1511.07122 [cs].

[89] Jinlun Zhang and DA Rothrock. Modeling Global Sea Ice with a Thickness and Enthalpy Distribution Model in Generalized Curvilinear Coordinates. *Monthly Weather Review - MON WEATHER REV*, 131, May 2003. doi: 10.1175/1520-0493(2003) 131⟨0845:MGSIWA⟩2.0.CO;2.

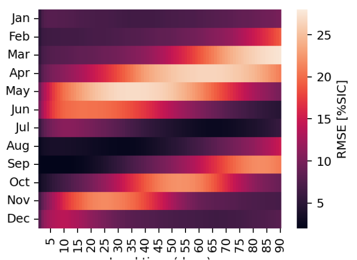
[90] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An End-to-End Deep Learning Architecture for Graph Classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), April 2018. ISSN 2374-3468, 2159-5399. doi: 10.1609/aaai.v32i1.11782. URL https://ojs.aaai.org/index.php/AAAI/article/view/11782.

[91] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, January 2020. ISSN 2666-6510. doi: 10.1016/j.aiopen.2021.01.001. URL https://www.sciencedirect.com/science/article/pii/S2666651021000012.

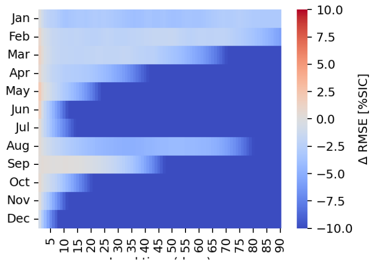
APPENDICES

Appendix A

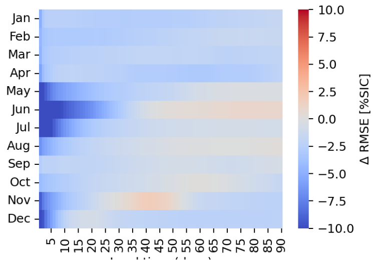
Additional RMSE Heatmaps



(a) Model

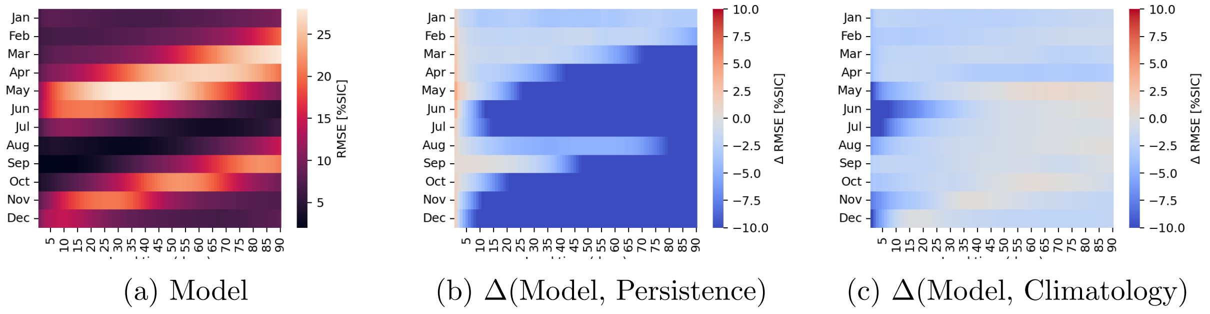


(b) Δ(Model, Persistence)



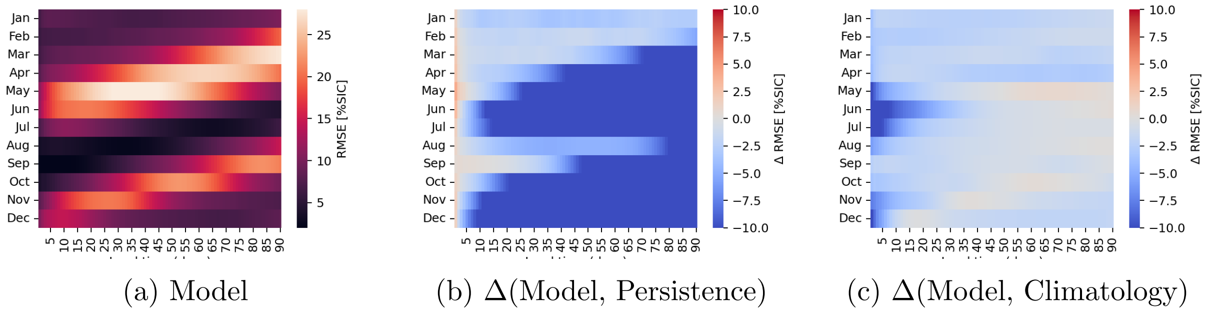
(c) Δ(Model, Climatology)

Figure A.1: RMSE heatmaps for the SIC forecasting task by month and lead time for the GraphSIFNet-Att model (a), and the RMSE differences between GraphSIFNet-Att and persistence (b) and climatology (c) where negative values (blue) indicate a reduction in model error relative to the baseline.



(a) Model (b) Δ(Model, Persistence) (c) Δ(Model, Climatology)

Figure A.2: RMSE heatmaps for the SIC forecasting task by month and lead time for the GraphSIFNet-Att-Reg model (a), and the RMSE differences between GraphSIFNet-Att-Reg and persistence (b) and climatology (c) where negative values (blue) indicate a reduction in model error relative to the baseline.



(a) Model (b) Δ(Model, Persistence) (c) Δ(Model, Climatology)

Figure A.3: RMSE heatmaps for the SIC forecasting task by month and lead time for the GraphSIFNet-Att-Reg model (a), and the RMSE differences between GraphSIFNet-GCN and persistence (b) and climatology (c) where negative values (blue) indicate a reduction in model error relative to the baseline.

Appendix B

Dynamic Re-Meshing Experiment on MovingMNIST

The following stand-alone paper written for the SYDE 672 course demonstrates initial experiments using a model similar to that used in this work for predicting frames of the MovingMNIST dataset. The model differs from GraphSIFNet as it uses an autoregressive model rather than a sequence-to-sequence model, and the spatial and temporal convolution steps are done in sequence rather than together. Nevertheless, it demonstrates the feasibility of evolving the underlying mesh during the unrolling process, and this method could potentially be combined with GraphSIFNet. This method is however computationally intensive, and optimizations would need to be made to train such a model in a reasonable amount of time.

Next-Frame Video Prediction on Sparse Image Sequences Using Graph Neural Networks

Zacharie Gousseau
Systems Design Engineering
University of Waterloo
Waterloo, Ontario
Email: zgoussea@uwaterloo.ca

Abstract—**In this study, I propose a reduction of basis technique for sequences of sparse images in the context of next-frame video prediction. Using a quadtree decomposition on a toy dataset of sparse images, we effectively reduce the dimensionality and transform each image into mesh-like spatially heterogeneous graph structures. A basic spatiotemporal graph neural network (GNN) model is introduced to perform next-frame prediction, which shows that despite the significant data reduction from the quadtree decomposition, the GNN performs slightly better using the reduced basis when the number of training epochs is fixed, in addition to significantly reducing training time. These results suggest that temporal graph networks have potential for handling mesh-like sequence data, though further investigation into more flexible networks that can model temporally heterogeneous meshes is recommended to improve generalizability.**

1. Introduction

Next-frame video prediction is a challenging problem that has largely been addressed using spatiotemporal model based on the convolutional neural network (CNN). These models are particularly well-suited to image data with a regular structure, such as photographs or video frames as they allow for learning rich features that are present in real-world images [1]. However, some image sequences, such as freeform 2D sketches or point clouds, are sparse in nature, meaning that the features of interest occupy a small subset of the image space. While CNN-based methods may still produce strong predictions on these types of images, they are not optimal since much memory and computation is wasted convolving over image regions containing little or no relevant information [2]. Some examples in which spatial sparsity may arise in sequential image data are:

- Sequences of handwritten or hand-drawn images, such as handwritten text or freeform sketches
- Satellite image sequences in which we are interested in change detection in some dynamic region surrounded by static features (e.g. forecasting the position of the ice edge over an ocean region during freeze-up)
- Sequences of images created by projecting 3D point clouds of objects onto a 2D plane
- Simulation meshes, such as a flag blowing in the wind, where the cell values correspond to physical distortions in the flag.
- Surveillance or security camera footage, which may contain large areas of background with only a few objects of interest present

To leverage the redundancy in sparse images, images can be decomposed into heterogeneous meshes with increased resolution in regions of high information density, and lower resolution elsewhere. The definition of *information density* is of course problem dependent and needs to be carefully considered as any reduction technique will lead to some degree of information loss. Here, a simple quadtree decomposition scheme is proposed to reduce redundant information and transform image sequences into sequences of spatially heterogeneous meshes. These meshes can then be thought of as semi-structured graphs with undirected edges between neighbouring mesh cells, and a spatiotemporal graph neural network (GNN) is then proposed to learn spatial relationships between nodes, as well as temporal dependences between consecutive frames.

Temporal GNN approaches have already been proposed for timeseries modelling on unstructured data. The temporal graph network (TGN) [3] was proposed as a generic temporal GNN framework for dynamic graphs where nodes and edges may be added or removed. The TGN retains a memory vector for each node in the network, which is updated at each timestep, including the node's deletion and insertion. In the update step, new embeddings for each node are computed using a message passing function, as well as the probablility of edges between any two nodes. This is a highly versatile framework ideal for highly unstructured, dynamic graphs. Images are of course highly structured, thus such a flexible architecture is not required. Several temporal GNN architectures have been proposed, largely in the traffic forecasting domain. These include recurrent networks [4] [5], attention-based models [6] [7], and simple message-passing networks [8] [9].

Little work has been done in studying the application of these techniques in next-frame video prediction. This is

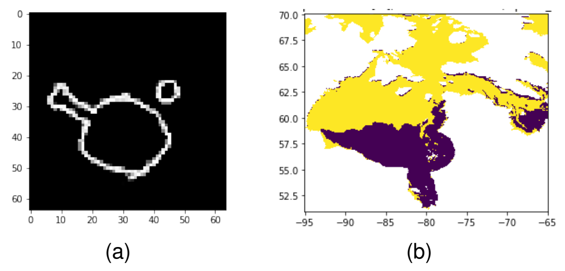


Figure 1. Two examples of sparse images. (a) Hand-drawn outline of a ping-pong paddle and ball; (b) Ice presence map over Hudson Bay.

unsurprising as graph-based approaches are most useful on unstructured data, and models based on convolutions have generally been accepted as the default framework for regularly gridded data. The closest related work is that of [10] in which an extension of the ConvLSTM introduced by [11] is proposed, where the 2-dimensional convolution operations is simply replaced by a graph convolution. They demonstrate strong performance on the MovingMNIST challenge, outperforming the ConvLSTM which they attribute to the lower number of parameters required to achieve the same receptive field size. They also demonstrate the rotation invariance intrinsic to graph networks by use of a modified MovingMNIST with rotating digits. This study investigates a similar technique, with the addition of a data reduction technique to increase efficiency on sparse image sequences.

2. Methodology

The following sections provide details on the toy dataset considered in this study, the reduction of basis technique, proposed model and experimental set-up.

2.1. Toy problem

In this work, a variation of the MovingMNIST dataset [11] is considered, in which a single digit moves about a canvas in a deterministic fashion. The object's initial position and velocity are randomly initiated, after which it moves in a straight line, bouncing off the boundaries of the canvas. Noise can be added by randomly nudging the x and y velocities during the trajectory, and adding white noise to the pixel intensities. In this toy dataset, a single 12×12 digit is used over a 32×32 canvas, illustrated in Figure 2.

2.2. Reduction of basis

Sparse images contain redundant information which can be discarded with minimal information loss. Here, I propose an irregular grid structure which reduces the image resolution in areas of low information density (i.e. empty regions in the toy dataset). This is achieved using a quadtree decomposition algorithm in which we start with a single cell covering the entire image, which is recursively split in four if the some condition is met. This condition could be the

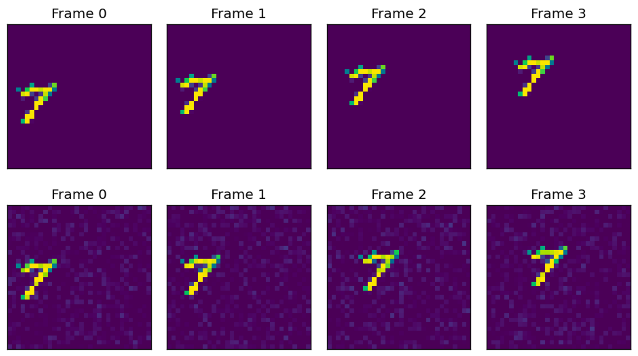


Figure 2. Four frames of the toy dataset without noise (top), and with added white noise in pixel intensity and object velocity (bottom). Both are identically initiated, but the velocity noise causes the digit to move more sporadically in the noisy frames.

maximum, sum, or variance of pixel values within the cell meeting some threshold. Here, a threshold on the variance is selected as it is regions with high variance (edges, gradients) which contain the most information. Some many-to-one mapping function needs to be specified to map the original gridded pixels to the mesh representation. This could be a learned mapping such as an MLP, or a simple aggregate. A mean over of the pixels is chosen here given that the cells are split on a variance threshold which ensures that each cell contains values within a relatively narrow band. Pseudo-code for the quadtree decomposition algorithm is given in Algorithm 1.

This can be viewed as a reduction in basis on an input image z, in which we could specify the transformation as:

$$\underline{\underline{z}} = F\underline{z} \qquad (1) \qquad\qquad \underline{z} = S\underline{\underline{z}} \qquad (2)$$

where F specifies the transformation from pixels $\underline{z}$ to nodes $\underline{\underline{z}}$, and S inverts the transformation. Clearly, some information is lost during these transformations such that $SF \neq I$ and therefore $\underline{z} - SF\underline{z} \neq \underline{0}$. The amount of information loss can be quantified as the mean squared error between the original and tranformed-untransformed images, which can be used to determine an appropriate value for the variance threshold used in the quadtree decomposition such that sufficient reduction is achieved while maintaining an acceptable level of information loss. In this work, an arbitrary variance threshold value of 0.05 was selected through trial-and-error. This is appropriate since a given region of a MovingMNIST frame either contains part of the digit or not, with no in-between. Thus, there is a single best threshold which can be found and optimization is not required.

2.3. A Graph-based Approach

CNN-based algorithms cannot be applied due to the irregular structure of the quadtree decomposed images. However, these structures can be viewed as meshes or graphs, which are the natural domain of graph-based techniques. In

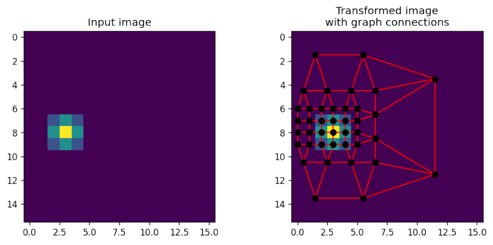


Figure 3. Example transformation of one frame to its graph representation. Black circles represent nodes and red lines are the (undirected) vertices connecting nodes. The graph is has a greater density of nodes over the object and a sparser structure elsewhere. The transformation has no effect on the image itself as it retains full resolution over the object while the rest of the image is occupied by zeros.

Algorithm 1 Quadtree decomposition

```
1:  procedure DECOMPOSE($image, x, y, s$)
2:      $thresh \leftarrow 0.05$
3:      $window \leftarrow image[x : x + s][y : y + s]$
4:      if $var(window) > thresh$ then
5:          for each $i \in I$ do
6:              for each $j \in J$ do
7:                  $image[i][j] \leftarrow mean(window)$
8:              end for
9:          end for
10:     else
11:         $s_1 \leftarrow s/2$
12:         DECOMPOSE($image, x, y, s_1$)
13:         DECOMPOSE($image, x + s_1, y, s_1$)
14:         DECOMPOSE($image, x, y + s_1, s_1$)
15:         DECOMPOSE($image, x + s_1, y + s_1, s_1$)
16:     end if
17: end procedure
```

particular, graph neural networks (GNNs) can easily handle such irregular meshes without special consideration. The decomposed images can be transformed into sets of nodes and vertices, where the centroid of each cell are the nodes, and vertices are inserted between any two adjacent cells. To retain spatial awareness, the position and size of each cell (node) is stored as attributes of the nodes, while distances between nodes are stored as edge weights. The edge weights ensure that nodes which are spatially distant from some node have less effect during the update step.

To model temporal dependencies, the model should be made to accept multiple frames as input for each prediction step. Decomposing each input image separately would require learning patterns on a dynamic set of graphs, which is a relatively complex task. To avoid this complexity, the input frames can be superimposed by using the maximum pixel intensity from all input images, and a graph created from the superimposed image. In this way, a single static graph can be used to represent all input frames. Furthermore, since we expect the object to move somewhere outside of the region of high node density in the next predicted frame, a two-pixel buffer is added. This scheme limits the amount of basis reduction, but greatly simplifies the modelling process. Figure 4 shows this procedure applied to three input frames.

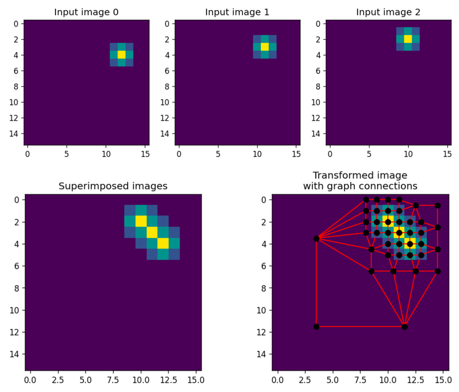


Figure 4. Example transformation of three input images to its graph representation using superimposition. Circles represent nodes and red lines are the (undirected) vertices connecting nodes. The resulting graph can adequately represent all three input frames.

2.4. Proposed Model

To model both spatial and temporal patterns, a hybrid message-passing graph neural network (MPGNN) and long-short term memory (LSTM) architecture (MPGNN-LSTM) is proposed. This model first encodes each input mesh separately using N stacked graph convolution layers with shared weights. ReLU activation and batch normalization is applied after ever graph convolution. The encoded states are then used as input into a set of M LSTM layers. Only the final state of the LSTM layers is retained and concatenated with the original inputs features. Finally, the embeddings are fed through a final fully-connected layer which produce the final output. A high-level depiction of the model is illustrated in Figure 5, and a more detailed description of the components are given in the following sections.

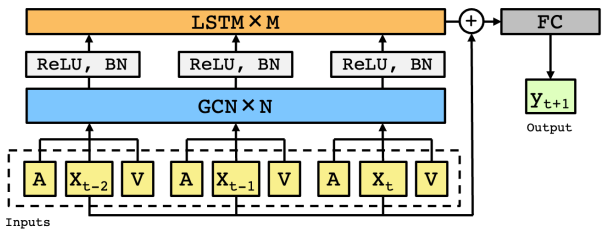


Figure 5. Overall architecture of the MPGNN-LSTM network. X_t is the node input feature matrix for timestep t, while A and V are the adjacency matrix and edge weights, respectively. The graph structure specified by A and V is shared among the inputs, hence the omission of subscripts.

2.4.1. Graph Convolutions. The model learns dependencies between connected nodes through graph convolutions. For a graph with N nodes, a graph convolution layer with feature dimensionality D takes the general form

$$H^{(l+1)} = f(H^{(l)}, A) \tag{3}$$

for a network with L layers, where $H^l \in \mathbb{R}^{N \times D}$ is the hidden state of the l^{th}, $f(\cdot)$ is some aggregation function, and A is the adjacency matrix for the given graph. The adjacency matrix $A \in \mathbb{R}^{N \times N}$ specifies the connections between nodes and has entries

$$A_{ij} = \begin{cases} 1 & \text{if } N_i \text{ and } N_j \text{ are connected} \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

A simple yet effective aggregation function is the function

$$f(H^{(l)}, A) = \sigma(AH^{(l)}W^{(l)}) \tag{5}$$

where $W^{(l)} \in \mathbb{R}^{N \times N}$ is the weight matrix of the l^{th} layer, and σ is some non-linear activation function. For a given node, we would like to update (rather than replace) its state using its neighbor's states, therefore it is useful to add self-loops into the adjacency matrix, i.e. connecting each node to itself. This can be done by simply adding the identity matrix to the adjacency matrix $\hat{A} = A + I$. To avoid exploding the values in the hidden states during the weighted multiplication, we can also normalize the adjacency matrix using the diagonal node degree matrix $\hat{D}$ of $\hat{A}$, which ensures each row sums to one. Kipf et al. suggest using a symmetric normalization, such that the adjacency normalization becomes $\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}$ rather than $\hat{D}^{-1}\hat{A}$. The graph convolution operation can therefore be written as

$$f(H^{(l)}, A) = \sigma(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}H^{(l)}W^{(l)}) \tag{6}$$

This is the form of the basic graph convolution proposed by Kipf et al. [12] which is used in this work through the PyTorch-Geometric implementation [13]. By stacking several graph convolution layers, the receptive field increases as each node receives messages from its neighbors, which themselves have received messages from their neighbors, and so on.

2.4.2. Long-Short Term Memory (LSTM). Temporal patterns in this proposed model are learned through the long-short term memory module, which learns patterns in sequential data through non-linear control gates which control the flow of information between subsequent LSTM cells. A given cell maintains a cell state $\mathbf{c}_t$ and hidden state $\mathbf{h}_t$ which accumulates information at each step through an input gate $\mathbf{i}_t$ which allows information to be accepted into memory, a forget gate $\mathbf{f}_t$ which decides which information is discarded, and the output gate $\mathbf{o}_t$ which controls the information propagated to the final output state. This allows each cell to track gradients thereby mitigating the vanishing gradient issue during back-propagation through time which plagues the vanilla recurrent neural network [14]. The following are the governing update equations of the LSTM:

$$\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f \cdot \mathbf{x}_t + \mathbf{U}_f \cdot \mathbf{h}_{t-1} + \mathbf{b}_f) \\
\mathbf{i}_t &= \sigma(\mathbf{W}_i \cdot \mathbf{x}_t + \mathbf{U}_i \cdot \mathbf{h}_{t-1} + \mathbf{b}_i) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o \cdot \mathbf{x}_t + \mathbf{U}_o \cdot \mathbf{h}_{t-1} + \mathbf{b}_o) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_c \cdot \mathbf{x}_t + \mathbf{U}_c \cdot \mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned} \tag{7}$$

where σ is the sigmoid function, and $\odot$ represents element-wise multiplication.

TABLE 1. DATASET CHARACTERISTICS

Characteristic	Value
Input frames	3
Image size	32x32
Digit size	12x12
White noise - pixel intensity	$N(0, 0.05)$
White noise - velocity	$N(0, 0.5)$
Num. samples	1000 (train), 100 (test)

TABLE 2. MODEL AND TRAINING CHARACTERISTICS

	Characteristic	Value
Training	Epochs	200
	Loss function	Mean squared error (MSE)
	Learning rate	0.01, decay by $\times 0.9$ every 5 epochs
	Optimizer	SGD, momentum=0.9
Model	LSTM layers	2
	CGN layers	4
	FC layers	2
	Hidden size	64
	Dropout	0.1

2.5. Experimental Set-up

Two experiments are conducted. In the first experiment, images are transformed into graphs using the input images' pixels as nodes directly. This is the base case in which no reduction of basis is performed, and the aim is to show that this problem can be adequately modelled using the proposed temporal GNN. In the second experiment, input images are decomposed using the quadtree decomposition as described in subsection 2.3. In the following section, the experiment without the quadtree decomposition reduction of basis is referred to as using the *full basis*, while the experiment with the reduction of basis is referred to as using the *reduced basis*. The same training set-up is used in both cases, as the aim is to show that similar results can be achieved with and without the reduction of basis.

The dataset consist of 1000 training samples and 100 test samples, each of which is a sequence of four images, where the first three are used as input, and the fourth is target. Each sample randomly selects a digit from the MNIST database [15] which is resampled to 12×12 and initiated at a random location on the canvas for the first frame, before being set along a random trajectory for following three frames. White noise is added to the pixel intensities and velocity fields. The dataset characteristics are summarized in Table 1

The model configuration and training set-up is also kept constant between the two experiments. Parameters such as the number of layers, hidden state size and dropout were set following trial-and-error, as were hyperparameters such as the learning rate and number of epochs. The configuration used in the experiments are listed in Table 2. The models were trained on a single NVIDIA Tesla V100 GPU.

TABLE 3. Summary of Experimental Results

	Num. Nodes	Test MSE	Training time (min)
Full Basis	1024	0.0154	89.2
Reduced Basis	100-200	0.0147	51.3
Percent Change	-80-90%	-4.5%	-42%

3. Results

A summary of the training results is shown in Table 3. The quadtree decomposition reduces the number of nodes from $32 \times 32 = 1024$ to 100-200 depending on the position of the digit on the canvas, representing an 80-90% reduction in data size. This corresponds to a reduction in training time by approximately 42%. This non-linearly scaling between data size and training time indicates that there is some overhead cost which remains constant between the two techniques. The mean squared error on the test set is lower after the basis reduction wihch may be surprising, however the loss curves in Figure 6 suggests a slower convergence using the full basis, and that additional epochs may have further reduced the loss. Note that the erratic test loss is likely a result of the small batch size (here, 1) which updates the model weights more often than with larger batch sizes.

Examples of predictions on a hold-out set using both models is shown in Figure 7. In both cases, the models adequately identify the general region in which the digit is likely to be in the next frame. The blurry edges are expected as the velocity noise adds stochasticity to the digit's next position, thus the models cannot be certain of the location. Predictions using the reduced basis better reproduce the shape of the digit and are more confident, while predictions using the full basis are more diffuse. Although it is likely that further training the full basis model would result in similar or better results than with the reduced basis, the improvement at the same number of epochs and with almost half the training time is noteworthy.

4. Discussion

The quadtree decomposition imposes a heuristic on the problem – regions of high variance (as a proxy for information density) will persist into the subsequent frames. Conversely, we also impose the heuristic that regions without high variance are uninteresting and should be modelled at a low resolution. In the MovingMNIST problem, these heuristics makes sense – digits move in a semi-deterministic manner, and never does a new digit appear elsewhere on the canvas. However, these heuristics could be viewed as too strong and problem-specific. For instance, the model could simply learn predict higher values over small cells and lower values over large cells without learning anything else about the dynamics of the problem and still achieve reasonable results. This limits the generalizability of the solution.

Furthermore, in this limited toy problem, we've ensured no information loss as all image regions that do not contain any part of the digit are left unsplit, and since the noise

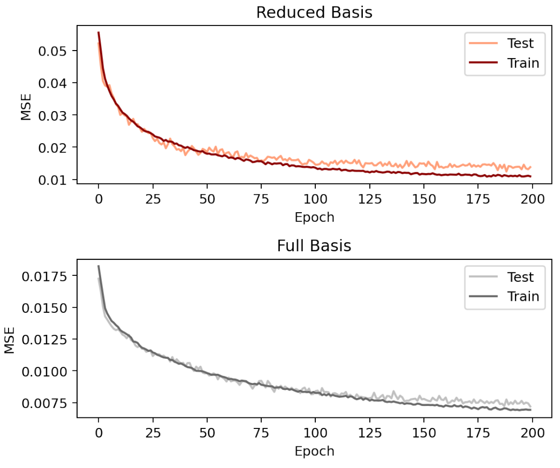


Figure 6. Loss curves for the reduced basis (top) and full basis (bottom). The difference in scale between the two plots is a result of the loss calculation which is performed on the graph representation, thus the MSE values cannot be directly be compared.

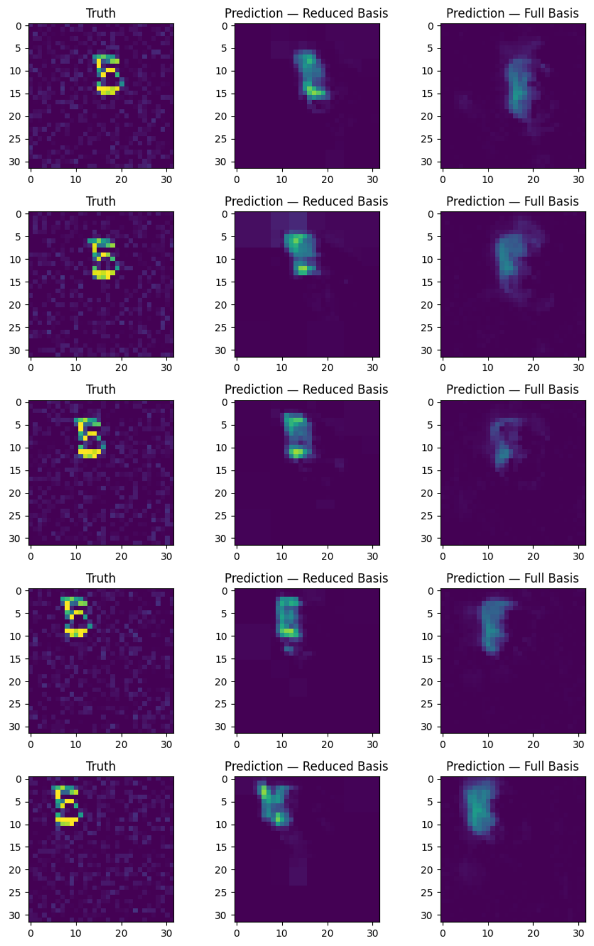


Figure 7. Five consecutive predictions using the reduced basis (middle), and the full basis (right), and the ground truth (left).

is white, the resulting mean value for each unsplit cell is close to zero. It is therefore essentially a noise reduction technique. In real-world datasets, the noise may not be white, and the decomposition may lead to artifacts which the model would need to learn to ignore.

5. Conclusion

This work shows promise in using quadtree decomposition for reducing the dimensionality of sparse image sequences, allowing for efficient next-frame video prediction using spatiotemporal graph neural networks. The results show that the GNN performed slightly better using the reduced basis when the number of training epochs was kept constant, while also significantly reduced training time. Although the focus in this work was on highly sparse images, the data reduction and modelling technique proposed here is not exclusive to these types of images. Quadtree decomposition could be applied to any image sequence dataset, however the amount of data reduction may be limited on more dynamic scenes. In these cases, CNN-based techniques may offer better overall performance.

5.1. Future Work

Several limitations exist in this study, and some suggestions for future work is provided here.

First, the toy problem used here is limited in scope, and should be expanded upon to better understand the method's generalizability. The results in this work appear to show the model memorizing the form of the digit which remains unchanged within image sequences. In principle, the model should be able to learn patterns in morphing or rotation of the digit, though this is not demonstrated here. Furthermore, the addition of non-white noise to the pixel values such as gradients or larger artifacts would better show whether the model can generalize to real-world datasets.

Furthermore, a heuristic was used in this work to ensure a single graph structure could be used on all input frames. That is, we asserted *a priori* that the location of regions of high information density would not change drastically between frames, thus superimposing the input images and adding a buffer to the resulting graph should adequately represent the information in the input frames as well as the next predicted frame. This heuristic does not necessarily apply generally, and a model which accepts graphs which are heterogeneous both in space and time would better generalize. One method of accepting temporally heterogeneous graphs would be to add temporal vertices between any two temporally adjacent cells. Careful consideration would need to be taken to model the temporal dependence in this structure. Graph convolutions could be used along the temporal axis such that the model is agnostic to whether a dimension is spatial or temporal, only distinguished by attributes of the vertices. In this case, edge features would need to be used and updated similar to node features, rather than using simple edge weights. Alternatively, the LSTM could still be used along each temporal trajectory, that is, every possible set of temporal vertices connecting the first and last input frame. Depending on the complexity of the graph structures, however, this could get prohibitively expensive.

Lastly, to fairly benchmark against other next-frame prediction models, the model should be trained and tested on the standard MovingMNIST using the standard metrics. It may also be useful to compare the performance and efficiency of both GNN models here against a baseline CNN-based model.

References

[1] Sergiu Oprea, Pablo Martinez-Gonzalez, Alberto Garcia-Garcia, John Alejandro Castro-Vargas, Sergio Orts-Escolano, Jose Garcia-Rodriguez, and Antonis Argyros. A Review on Deep Learning Techniques for Video Prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(6):2806–2826, June 2022. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[2] Jianning Li, Christina Gsaxner, Antonio Pepe, Dieter Schmalstieg, Jens Kleesiek, and Jan Egger. Sparse Convolutional Neural Networks for Medical Image Analysis, February 2022.

[3] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal Graph Networks for Deep Learning on Dynamic Graphs, October 2020. arXiv:2006.10637 [cs, stat].

[4] Jinyin Chen, Xueke Wang, and Xuanheng Xu. GC-LSTM: Graph Convolution Embedded LSTM for Dynamic Link Prediction, October 2021. arXiv:1812.04206 [physics].

[5] Aynaz Taheri and Tanya Berger-Wolf. Predictive Temporal Embedding of Dynamic Graphs. In *2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 57–64, August 2019. ISSN: 2473-991X.

[6] Jiandong Bai, Jiawei Zhu, Yujiao Song, Ling Zhao, Zhixiang Hou, Ronghua Du, and Haifeng Li. A3T-GCN: Attention Temporal Graph Convolutional Network for Traffic Forecasting. *ISPRS International Journal of Geo-Information*, 10(485):485, July 2021. Publisher: MDPI AG.

[7] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. Attention Based Spatial-Temporal Graph Convolutional Networks for Traffic Flow Forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):922–929, July 2019. Number: 01.

[8] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning Mesh-Based Simulation with Graph Networks, June 2021. arXiv:2010.03409 [cs].

[9] Ryan Keisler. Forecasting Global Weather with Graph Neural Networks, February 2022. Number: arXiv:2202.07575 arXiv:2202.07575 [physics].

[10] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Structured Sequence Modeling with Graph Convolutional Recurrent Networks, December 2016. arXiv:1612.07659 [cs, stat] version: 1.

[11] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. *arXiv:1506.04214 [cs]*, September 2015. arXiv: 1506.04214 version: 2.

[12] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks, February 2017. arXiv:1609.02907 [cs, stat].

[13] Matthias Fey and Jan Eric Lenssen. Fast Graph Representation Learning with PyTorch Geometric, April 2019. arXiv:1903.02428 [cs, stat].

[14] Alex Sherstinsky. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *Physica D: Nonlinear Phenomena*, 404:132306, March 2020.

[15] Li Deng. The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. *IEEE Signal Processing Magazine*, 29(6):141–142, November 2012. Conference Name: IEEE Signal Processing Magazine.