Sampling without stratification: end-to-end methods for farm-scale soil carbon monitoring

by

Jenny Moralejo

S.B. in Computer Science and Engineering
and Mathematical Economics
Massachusetts Institute of Technology (2024)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2024

Authored by: Jenny Moralejo
Department of Electrical Engineering and Computer Science
August 23, 2024

Certified by: Sherrie Wang
Assistant Professor of Mechanical Engineering
Thesis Supervisor

Accepted by: Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Sampling without stratification: end-to-end methods for farm-scale soil carbon monitoring

by

Jenny Moralejo

Submitted to the Department of Electrical Engineering and Computer Science
on August 23, 2024, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Computer Science and Engineering

Abstract

Reversing the effects of climate change requires drawing gigatons of carbon dioxide out of the atmosphere and sequestering it in chemical, biological, or mineralogical forms. Within the Earth system, soils represent the largest terrestrial stock of carbon [6], and monitoring soil carbon is essential both for soil conservation and for efforts to reduce atmospheric carbon dioxide levels. Agricultural soil carbon sequestration has been proposed as an intervention to mitigate atmospheric carbon levels. However, in order for soil carbon sequestration to be viable both scientifically and economically, e.g. in a carbon crediting system, reliable ways of monitoring, validating, and predicting soil carbon must be developed. Current measurement methodologies are prohibitively costly to apply at global scale [23]. The thesis simulates and analyzes simulated field-level soil carbon data to train machine learning models which perform 3 tasks in an end-to-end fashion: (1) determining soil carbon stocks in unsampled areas based on sparse measurement data, (2) leveraging historical data to predict future changes to soil carbon levels, and (3) determining the optimal number and siting of samples for the next measurement cycle, to maximize the predictive power achieved per dollar spent on measurement.

Thesis Supervisor: Sherrie Wang
Title: Assistant Professor of Mechanical Engineering

Acknowledgments

First and foremost, I would like to thank my advisor Professor Sherrie Wang and my mentor Dr. Evan Coleman for our numerous meetings and mentorship. I'm incredibly grateful to have learned from you both and I've taken away so much through your expertise and passion for the space! Additionally, I am fortunate to have had amazing mentors and lab mates. I would like to thank everyone in the Earth Intelligence Lab who have given me invaluable guidance and feedback and have made every day in lab so fun and exciting: Dr. Jonathan Giezendanner and Dr. Yuhao Nie; future Drs. (!) Kerri Lu, Qidong Yang, Jordi Laguarta Soler, and Chenhui Zhang; fellow M.Engs Zitong Chen and Stephen Campbell; and Eldar Urkumbayev. I am very lucky to have had teachers who I've learned so much from and have continued to help mentor me throughout the ups and downs of undergrad and M.Eng. Thank you Dr. Christina Ji and future Dr. Fankeng Sun! I would like to thank the MIT EECS office, especially Jessica Zdon-Smith for the guidance, perspective, and advice throughout this past year. Finally, I would like to thank my family and friends for their constant support.

The only reason I am able to get here is because of the wonderful people in my corner who have helped me throughout my life. Thank you all very much!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Atmospheric carbon directly contributes to climate change with increasing carbon levels in the atmosphere being associated with a rise in global temperatures. In order to combat the effects of climate change, various methods of sequestering carbon have been proposed. Soil contains and is capable of holding the largest terrestrial stock of carbon [6] and as such soil management has become increasingly important to aid in sequestration efforts [10]. In order to provide a financial incentive for this management and sequestration, trading carbon offsets as part of a carbon credits system has become increasingly relevant. There exist both voluntary and compliance carbon markets and as these markets become more widespread, accurately accounting for sequestered carbon is imperative for a well-functioning system. However, soil carbon measurements are extremely expensive and time intensive to collect. As estimated by CSIRO in 2013, the cost of analyzing a single sample in lab (disregarding labor and equipment costs) is $AUD 30 [1]. Accounting methods generally rely on sampling soil carbon at at a discrete number of points within a region and then extrapolating these measurements to obtain the full stock of carbon in the soil. A common recommendation in Australia is to use a 25m $\times$ 25m plot of land as a sampling unit [14]. Over farms that span hundreds of acres (where 1 acre is around 4046 square meters), thousands of samples would need to be taken. The laboratory costs alone, which by

themselves are a small fraction of the overall cost, are neither scalable nor feasible. The possibility of accurately predicting soil carbon levels with selective and limited sampling to improve cost and allow for a more viable carbon market will be explored in this paper.

Uniform random sampling is an unbiased and common way to converge on and estimate some variable's true distribution, but it is both costly and unfeasible to implement over large farms. A common approach due to the spatial correlation between areas of farms, is to split the farm into strata, grouping areas of land that have similar characteristics to each other and theoretically similar carbon distributions as suggested by the patent filed by Wiseman et al. [29]. Uniform random sampling is shown to have worse results than stratified random sampling when appropriate strata can be found [3]. Thus, more recently developed sampling strategies rely on stratifying farmlands into different sections usually via geographical features, minimizing sample variance, or other machine learning techniques such as k-means [4]. After stratification, kriging or gaussian process regression is often applied to the individual strata to create a dense soil carbon map. However, techniques to stratify farms are limited and currently do not have a large amount of data to inform the stratification. Having more data can help to inform stratification which can in turn increase prediction accuracy. Developing sampling techniques informed by data can improve prediction accuracy while minimizing costs.

1.2 Related Work

Various organizations have developed their own methods for predicting soil organic carbon levels at local levels through basic methods such as linear regression and k-means clustering. Of these methods, the best linear unbiased predictor of interpolated carbon from samples is kriging or Gaussian process regression. The benefit of kriging over methods like linear regression is that spatial correlations between points are used to interpolate values. Multiple types of kriging exist and have been applied to this problem such as ordinary kriging [28] and regression kriging (including

more covariate features) [22]. While the results are promising, kriging relies on the assumption of second-order stationarity and other environmental factors can impact the spatial covariance between two sampling points. Additionally, kriging becomes quite computationally expensive with higher dimensions and more data points.

However, the relationship between soil carbon and other variables are not always linear [11]. In order to account for these effects, deep learning approaches have been developed. Convolutional neural networks have been used to predict a variety of soil properties including soil carbon [32]. Other ensemble machine learning algorithms such as Random Forest, Gradient boosted decision trees, and Extreme gradient boosting on a combination of remote sensing data such as satellite imagery and spectral information have also been used [30].

The above approaches rely solely on satellite data in specific areas and are limited in their performance due to limited spectral and spatial resolution, limited quantity of data, and challenges with spectral decomposition to accurately separate out the signals of soil organic carbon.

Machine learning models that rely on sampled carbon data points as well as remote sensing have been developed [15]; however these studies also suffer from a lack of data and focus on specific areas, thus the models do not generalize well.

Few studies employ data from previous time steps to inform future predictions - something that is essential to understanding soil carbon dynamics. Changes that occur in soil year by year are small but manifest in greater ways in the future. In order to accurately predict soil carbon levels and their changes, establishing and accounting for changes over time is essential.

One study by Kakhani et al. utilized an LSTM to make predictions over time using a base CNN with a spatial attention mechanism which outperformed the traditional Random Forest approach to this problem [9]. This approach combines satellite imagery with climate temporal information, however the satellite imagery was not temporal and only the climate input impacted the LSTM module. Thus while they were able to predict soil carbon well by incorporating some temporal information, more work remains to be done to predict changes in carbon across time.

Ultimately, these models are do not have enough sampled data to inform their predictions and this data is expensive to gather.

Different sampling strategies have been developed to improve this process which rely on stratifying farmlands either by features or by distance and sampling based on these groupings. The recommended soil carbon sampling practice in Australia is to divide the farm into 25m x 25m squares [14], while the NSW DECCW recommends to choose a 25m x 25m area, divide it into 10x10 quadrants and randomly sample 10 times. Dividing a farm into equal areas leads to biased extrapolations and choosing a specific quadrant incentivizes dishonest sequestration practices for overestimating carbon stocks [4].

Multiple approaches have been taken to address these issues by stratifying farms in more informed ways to reduce sampling variance and capture a more certain estimate of the change in carbon content. Ospats is a method that optimizes stratified random sampling by assigning discretized points on a farm to a stratum by examining the pairwise differences between predictions and the covariances of prediction errors. The predictions and prediction errors are drawn from an existing carbon map which is then stratified by this metric [4]. While this method is able to to optimally allocate strata based on prediction error, it relies on data being bootstrapped directly from an existing carbon map with prediction errors. This carbon map is often generated by linear regression run on previously collected data, which is also limited in scope and thus accuracy. This is the case for many methods to determine strata and carbon stock such as K-means [13] and multivariate regression [22]. While carbon maps can also be determined from the cited deep learning approaches above (which account for non linearities), the performance of these models is limited. Additionally, while accurate stratum determination can decrease the number of samples needed for an accurate audit, the proposed sampling methodology doesn't take labor and travel costs into account. More work remains to be done to analyze how mechanizable a sampling methodology is [4].

These approaches rely on explicitly structuring predicted carbon maps into strata through various methods to inform sampling. Other methods which rely less on stra-

tum determination such as balanced sampling [18] have also been evaluated on this problem. Compared with stratified random sampling and uniform random sampling, doubly balanced sampling performed significantly better, reducing the required sample size by 32% [18], however it is not possible to have an accurate quantification of its uncertainty [7].

Rather than using a specific algorithm to determine stratum explicitly, we propose a method to recommend sampling points that minimize a cost, which is a function of the number of points as well as prediction error.

1.3 Statement of Contributions

Technically, this work hopes to accomplish four things. First, to simulate data in a better-understood soil environment to address the lack of data available for a proof-of-concept. Second, to develop a machine learning model to estimate soil carbon levels at the given timestep as well as over time. Third, to predict soil carbon levels over time. Finally, this model recommends various sampling points to be fed into the model for enhanced future prediction. The goal is to develop a sampling approach that gets close to the performance of stratified sampling in a way that is economically and physically feasible. This approach does not rely on the stratification of farms through the methods discussed above, but rather an implicit stratification, and the model outputs a sample allocation based off of this implicit stratification.

At a higher level, the contribution of this work is to introduce a framework of prediction and sample allocation across time which, due to the lack of temporal data availability, has not been thoroughly researched. This work hopes to offer a proof of concept for what is possible within the field and encourage further ground-truth data collection based on expensive but highly informative laboratory methods.

The rest of this thesis is organized as follows: In Chapter 2, the simulated dataset, simulation methods, and sampling strategies on this dataset are explained. Chapter 3 introduces the stock estimator model for creating a dense carbon map from samples, including prior work, comparisons to kriging baselines, and overall results. Chapter

4 extends on the model introduced in Chapter 3 to produce dense carbon maps from a sequence of samples collected across time, and predict the carbon distribution for future timesteps. Chapter 5 extends upon this model further to output recommended sampling locations that maximize the information gain of the previous models and compares the output to established sampling strategies in the field and analyzes the results for this particular dataset. Finally, Chapter 6 offers concluding remarks and future research directions.

Chapter 2

Data

2.1 Peatland Model

High resolution data of soil carbon levels at different points of land in farms does not exist publicly at a large scale. Available data is often scattered across various publications and publicly available soil databases such as the Rapid Carbon Assessment database (RaCA) [24] in the United States and the Land Use and Coverage Area frame Survey (LUCAS) [16] in Europe. This data unfortunately hasn't been collected regularly. In the case of LUCAS, there is only data for the years 2009, 2012, 2015, and 2022. The geographic distribution of this data also varies depending on the year. While a large number of samples has been taken, these are insufficient to establish the ground truth required to train a temporal model. In order to compensate for this lack of data, synthetic data was generated. All analyses conducted in the sections to follow were performed on this synthetic data.

While soil dynamics are not completely understood at a fine-grained level, the carbon accrual in peatlands is better understood than in other systems. Carbon accrual in peatlands is tied directly to the shape and volume of a given peat bog relative to the height of the water table within the bog [2]. Thus, we are able to use the volume of peat and average carbon density of a bog to estimate the carbon stock. Using the model by Cobb et al. [2] which estimates change in peat bog shape over time, artificial data on soil carbon stocks in peatlands over time under the assumption

of a uniform water table was generated.

This model developed describes the stable morphology of peat using a partial differential equation (Equation 7 in [2]) and is written below:

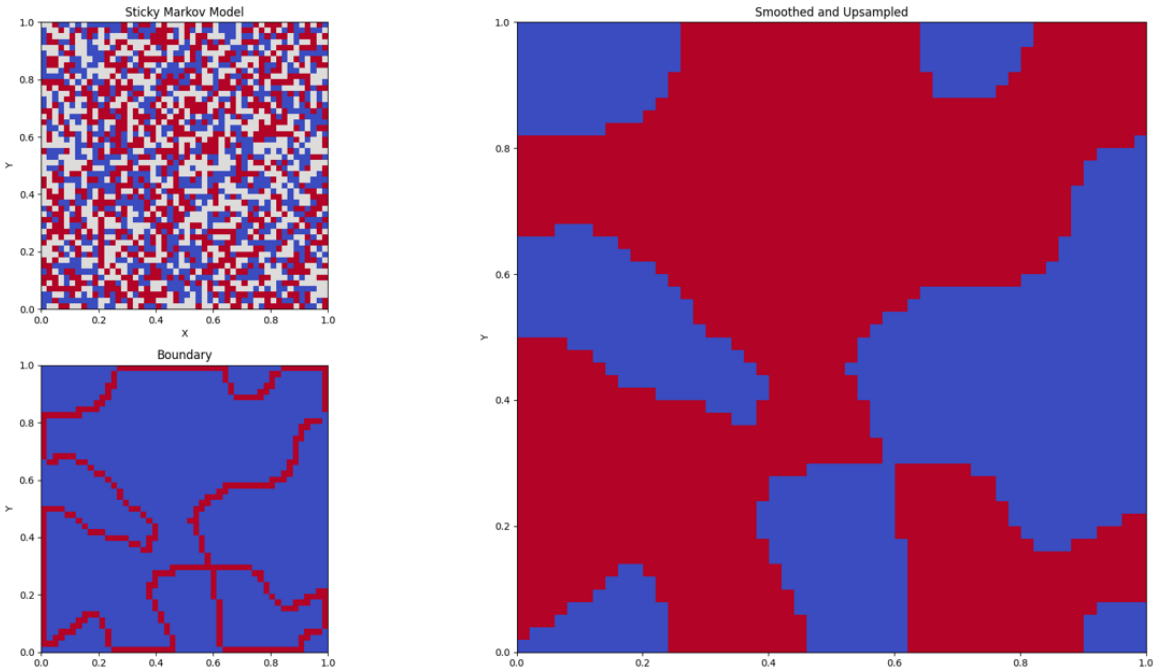
$$\nabla^2 p_\infty = -\frac{\langle q_n \rangle}{\langle T \rangle} \tag{2.1}$$

The variable $\langle q_n \rangle$ is the average net precipitation and $\langle T \rangle$ is the average transmissivity.

There is a lack of data on transmissivity and net precipitation, especially for the particular environment our simulation is based on. There are currently different approaches to generate this data, including using an existing model such as the one used in the Cobb et al.'s approach or training a new model using available datasets such as the ERA5 surface air temperature, precipitation and soil moisture dataset [21]. However, using an existing model would require further field-specific data. Thus the decision was made to set $-\langle q_n \rangle / \langle T \rangle = 100$ in the simulated data as the focus of the paper is sampling methodology rather than simulation. Using an existing dataset such as ERA5 could help model more realistic and varied conditions instead of just a singular rainfall regime, and is a potential avenue for further exploration as more granular data is collected.

An important note is that peatlands are a diverse system and their functionality and carbon accrual can vary depending on their surrounding ecosystem. The majority of peatlands are in temperate and boreal zones [20], but the model of [2] only suggests a method to predict carbon flux for those in tropical zones. While such a simulation may thus be limited, the proposed end-to-end system provides a proof of concept of what is possible. Due to regulatory requirements such as those developed to integrate with the European Union Emissions Trading System [27], it is expected a large dataset will be collected in the future, and this simulation can in principle be replaced by the collection of super-resolution empirical data.

Figure 2-1: Bog Generation Process



Top left: the sticky markov model used to randomly generate bog shapes; Bottom left: the boundary of the peat bogs; Right: The smoothed model denoting the shape of the peat bogs where the red areas denote the bog.

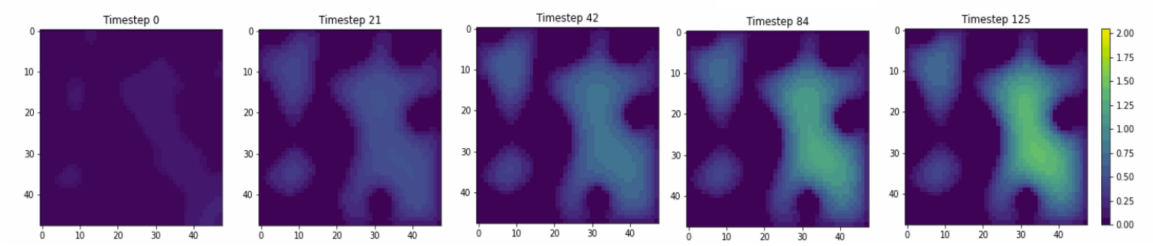
2.2 Dataset Creation

2.2.1 Ground Truth Simulation

In order to simulate peatland initial states, a $K \times K$, $(K = 48)$, grid filled with a random integer in the interval [-1, 1] is created. This grid is smoothed by separating the grid into $\lfloor K/C \rfloor$ blocks where C represents the coarse graining factor. The C used was 4, resulting in 12 4x4 blocks in each dimension (width and height). Each block is averaged over downsampling to a smaller grid of size $\lfloor K/C \rfloor \times \lfloor K/C \rfloor$. In order to slightly smooth and upsample back to the original shape, the Kronecker product is taken between the downsampled grid and a same sized grid full of ones and additional padding is added as needed. Finally a Gaussian filter is applied to create rounded bog shapes as shown in Figure 2-1.

Given the initial state and boundary mask, the finite difference method was used to incrementally solve for the solution to the stable peatland morphology within the

Figure 2-2: Peatland Evolution Across Time



Left to right: Bog morphology at timestep 0, 21, 42, 84, and 125 after applying the finite difference method to solve the equation governing peat shape evolution.

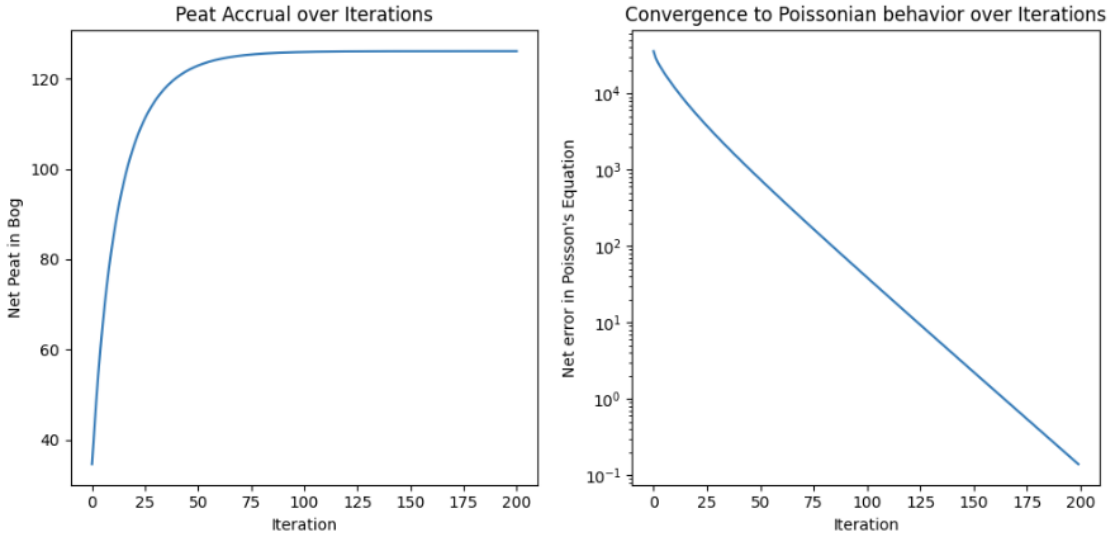
boundary where the $-\langle q_n \rangle / \langle T \rangle$ constant was set to 100. At each timestep, each point within the bog boundary is updated by taking the average of the neighboring points adjusted by the source term constant driving peatland development. The finite difference method is used to solve for the Laplacian of the bog to determine whether the surface has satisfied the equation which governs it (2.1) and to determine a stopping timestep for simulation.

Each step of the finite difference method was treated as an effective timestep. While the actual change strongly depends on the water table and precipitation conditions, due to the lack of data we are not accounting for these varying conditions and rather assuming the ratio is the constant 100. Under this constant and the starting state of the simulated peats, convergence happens after around 125 steps. Figure 2-2 depicts the evolution of the peat bog starting at timestep 0 and ending at timestep 125.

One observation to note regarding the data simulation is that the peat accrual across time is not linear as what drives the surface development is Poisson's equation. Figure 2-3 shows the rate of this accrual for a generated farm in the datasets as well as the decrease in the residual between the Laplacian of the bog surface and the expected morphology dictated by Equation 2.1. As shown by this figure, peat accrual flattens out after around 125 timesteps and the residual error begins to shrink much more slowly after that, thus 125 was decided to be the stopping point for simulation.

Two datasets were generated using this method, a smaller one consisting of 125 randomly generated farms and a larger one consisting of 625 randomly generated

Figure 2-3: Peatland Accrual and Convergence Across Time



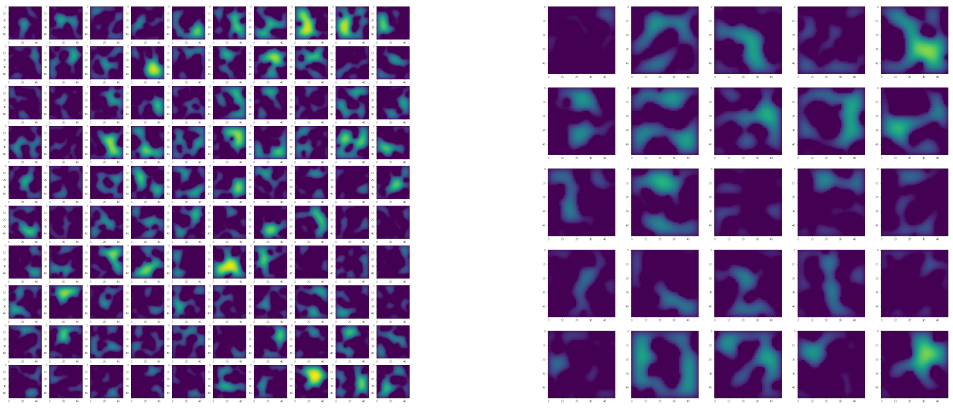
Left: The net carbon amount in a farm across timesteps 0 to 200. Right: The residual error between the surface Laplacian and the expected stable morphology across time.

farms. The farms are of size $K \times K$, $(K = 48)$ simulated across $T = 125$ timesteps. Let $\mathcal{T}_i(x, y, t)$ represent the "ground truth" soil carbon stock of a peat bog i at position (x, y) where $x, y \in [0, K-1]$ and some time $t \in [0, T]$. The smaller dataset of generated farms are shown in Figure 2-4 split into training, validation, and test sets for use in later experiments.

A note is that the generated data doesn't have explicit units, however units can be assigned. The summary statistics on the pixel-wise carbon amount and farm-wise carbon amount of the smaller and larger datasets are displayed in Tables 2.1, 2.2, 2.3, and 2.4 respectively. Overall the test and training sets for both dataset have similar distributions with a few outliers evidenced by the discrepancy in the maximum carbon per farm-timestep snapshot between the training and test sets. As visualized by Figure 2-4, which contextualizes the model and experiment results discussed in further chapters, much of the area in the farm snapshots contain no carbon whatsoever. In fact, the median per pixel carbon amount is 0.

Figure 2-4: Smaller Generated Farm Dataset

(a) Training and Validation Set Farms (b) Test Set Farms



The randomly generated bog masses for the complete smaller simulated dataset pictured at the final simulated timestep.The training set (left) consists of 100 farms and the test set (right) of 25. The lighter areas indicate higher soil carbon content and vice versa.

Table 2.1: Smaller Simulated Dataset Pixel-Wise Carbon Summary Statistics

	Smaller Dataset Per Pixel Carbon								
	Mean	Std.	Min	25%	50%	75%	Max	Pixel Observations	Farms
Train Val Set	0.1712	0.2734	0.0000	0.0000	0.0000	0.2585	2.0543	29030400	100
Test Set	0.1497	0.2402	0.0000	0.0000	0.0000	0.2304	1.6949	7257600	25

This table presents the summary statistics for the carbon present at each pixel of the simulated data for the training validation set and the test set of the smaller 125 farm dataset. Pixel observations represents the total number of observations from the 48x48 pixels per farm across all farms and simulated timesteps.

Table 2.2: Smaller Simulated Dataset Farm-Timestep Snapshot Carbon Summary Statistics

	Smaller Dataset Farm - Timestep Snapshot Carbon								
	Mean	Std.	Min	25%	50%	75%	Max	Farm Timestep Obs.	Farms
Train Val Set	394.5013	245.3079	26.9769	201.3374	341.1628	527.6023	1457.0801	12600	100
Test Set	344.8715	223.0766	17.8296	175.8912	284.3709	517.3605	1078.8898	3150	25

This table presents the summary statistics for the aggregated carbon of each 48x48 farm snapshot across the 126 simulated timesteps for the training validation set and the test set of the smaller 125 farm dataset. Farm Timestep Obs. denotes the number of farm-timestep observations.

Table 2.3: Larger Simulated Dataset Pixel-Wise Carbon Summary Statistics

	Larger Dataset Per Pixel Carbon								
	Mean	Std.	Min	25%	50%	75%	Max	Pixel Observations	Farms
Train Val Set	0.1685	0.2696	0.0000	0.0000	0.0000	0.2539	2.3467	145152000	500
Test Set	0.1702	0.2677	0.0000	0.0000	0.0000	0.2590	2.1044	36288000	125

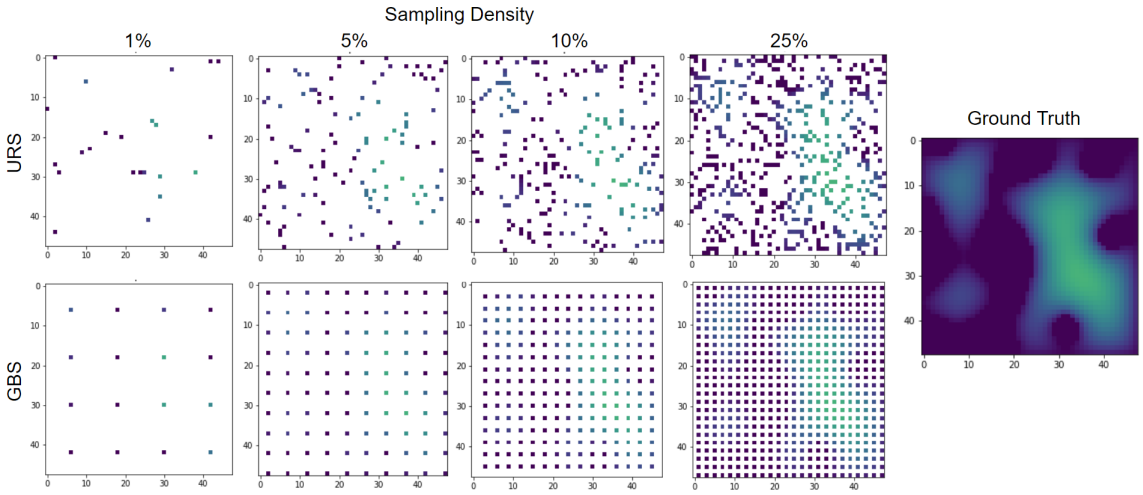
This table presents the summary statistics for the carbon present at each pixel of the simulated data for the training validation set and the test set of the larger 625 farm dataset. Pixel observations represents the total number of observations from the 48x48 pixels per farm across all farms and simulated timesteps.

Table 2.4: Larger Simulated Dataset Farm-Timestep Snapshot Carbon Summary Statistics

	Smaller Dataset Farm - Timestep Snapshot Carbon								
	Mean	Std.	Min	25%	50%	75%	Max	Farm Timestep Obs.	Farms
Train Val Set	394.5013	245.3079	26.9769	201.3374	341.1628	527.6023	1457.0801	12600	100
Test Set	344.8715	223.0766	17.8296	175.8912	284.3709	517.3605	1078.8898	3150	25

This table presents the summary statistics for the aggregated carbon of each 48x48 farm snapshot across the 126 simulated timesteps for the training validation set and the test set of the larger 625 farm dataset. Farm Timestep Obs. denotes the number of farm-timestep observations.

Figure 2-5: Comprehensive Sampling Strategy Visualization



This figure presents how farm 5 of the smaller training set at timestep 97 of the simulation is sampled for the dataset. The columns from left to right are samples taken at the 1%, 5%, 10%, and 25% densities. The top row corresponds to samples taken through uniform random sampling (URS) and the bottom row with grid based sampling (GBS). The rightmost image is the ground truth from which the samples are taken.

2.2.2 Sampling Simulation

Finally, in order to develop models that take in spatial carbon samples as input to output carbon maps and recommended sampling points, the above data needs to be sampled. The sampling strategies were chosen based on what is currently practiced in the field today. Grid based and uniform random sampling were performed on these generated farms at the following densities 1%, 5%, 10%, and 25% for each timestep between 0 and 125 as shown in the example in Figure 2-5. Due to the nature of grid based sampling for a constant sampling density, the same points are sampled at each farm-timestep snapshot whereas for uniform random sampling each farm-timestep snapshot will have a different set of sampling locations. Additionally, due to the spatial constraints of grid based sampling, 16, 100, 225, and 576 samples were taken under this strategy to be as close to 1%, 5%, 10%, and 25% of the $K \times K$, $(K = 48)$ grid without going over. This results in a larger number of samples under the uniform random sampling regime, however the largest discrepancy is under the 5% sampling density with 115 points for URS and 100 points for GBS.

Using this data, the prediction component of this project involves three parts:

estimating the stock at the current timestep, predicting the stock at future timesteps given historical sampling data, and outputting sampling points that are optimized with regards to cost and prediction accuracy to be covered in the next chapters.

Chapter 3

Stock Estimator

Using the simulated data, a machine learning model was trained to predict and generate a soil carbon map for a single-timestep sampling. This section is concerned with predicting the soil carbon levels at a given timestep without additional temporal knowledge.

3.1 Autoencoders

The problem of generating a dense carbon map from scattered samples can be framed as a simplified image inpainting problem, where images with missing pixels are filled in to reconstruct the image realistically. Autoencoders have been used for many decades for feature reduction and recently many variants have been used for generative modeling, such as image inpainting, as well [33]. The samples in the dataset and the surrounding ground truth areas are highly spatially correlated and fit well with the image reconstruction problem. Given some input, Autoencoders work by encoding this data to a latent space (lower dimensional representation) and then reconstructing the data from the latent representation.

A simple convolutional autoencoder model is used to generate a dense soil carbon map from samples. Although more sophisticated models have been proposed such as incorporating texture loss for more realistic refinement [31] and using partial convolutions to handle irregular holes [12] (such as those present in our dataset) the purpose

of this section is to establish a baseline proof of concept on a more simplistic dataset and while using some of these techniques could potentially improve performance, it is not the focus.

Additionally it is of note that not much work has been conducted on training models for infilling extremely sparse irregular holes. The "unmasked pixel" density in the dataset ranges from 1% to 25% to reflect the lack of dense samples in the soil carbon collection field today, whereas many of these models such as Liu et al.'s [12] train on masks with a hole to image ratio of at most .5 to .6 or at least about 17% of the image being "sampled". Many other networks don't consider irregular holes, such as the work by Yang et al. [31] which is trained on 128 x 128 images that have a 64 $\times$ 64 square hole at the center with 75% of the image being "sampled". This level of sampling density is currently unattainable for soil carbon collection due to the cost. Models that are trained to learn from a small amount of data are imperative in this space.

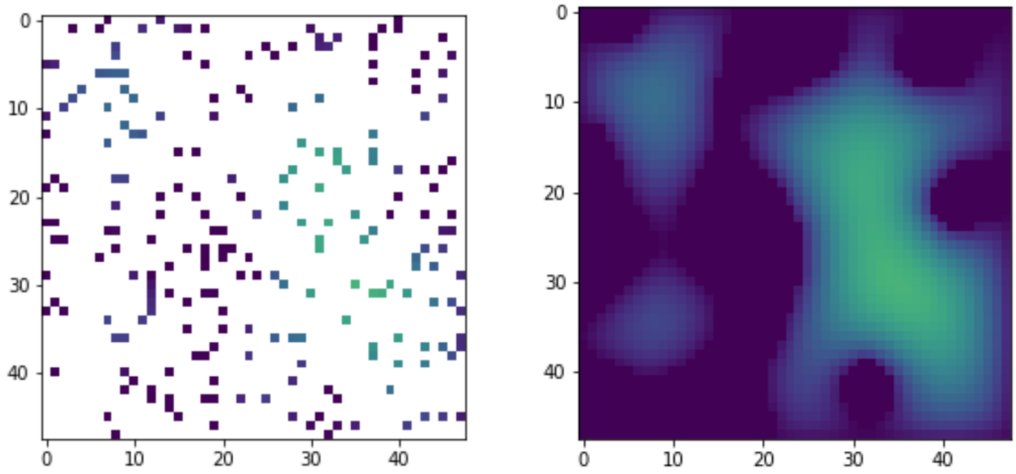
3.2 Model Architecture

This **stock estimator** $\mathcal{E}$ takes samples from some ground truth simulation $\mathcal{T}_i(x, y, t)$ of farm i at some timestep t, $X^{i,t} = \{X_1^{i,t}, X_2^{i,t}, ..., X_n^{i,t}\} \in \mathcal{T}_i(x, y, t)$ as input. $X^{i,t}$ is a sequence of positions (x, y) of length n which represents the locations of the samples. These samples are reshaped into $K \times K$ matrix where the soil carbon content of a sample at position (x, y) corresponds directly to the row x and column y of the matrix. The estimator outputs a superresolution estimate of the current carbon stock, $\hat{\mathcal{T}}_{\mathcal{E}_i}(x, y, t)$ for all points on farm i at time t (ie: $\forall x, y \in [0, K-1]$). The model $\mathcal{E}$ has two input channels: a $K \times K$ empty image splatted with sampled values from the ground-truth simulation, $\mathcal{T}_i(x, y, t)$ as well as a $K \times K$ mask with a 1 at a position (x, y) if there is a sample at that position, otherwise 0. The expected input and output for this model is shown in Figure 3-1.

An implicit assumption made here is that the carbon stock is uniformly distributed within the mass of peat, so that no other covariates impact the carbon estimation

Figure 3-1: Stock Estimator Expected Input and Output

(a) Input Image (b) Ideal Output Image



Left: The input image to the proposed stock estimator containing an empty image with uniform random sampled values from the ground truth simulation. Another channel masking the samples with ones and the non sampled points with zeros is also passed in. Right: The expected output image containing superesolution carbon estimates across the entire farm.

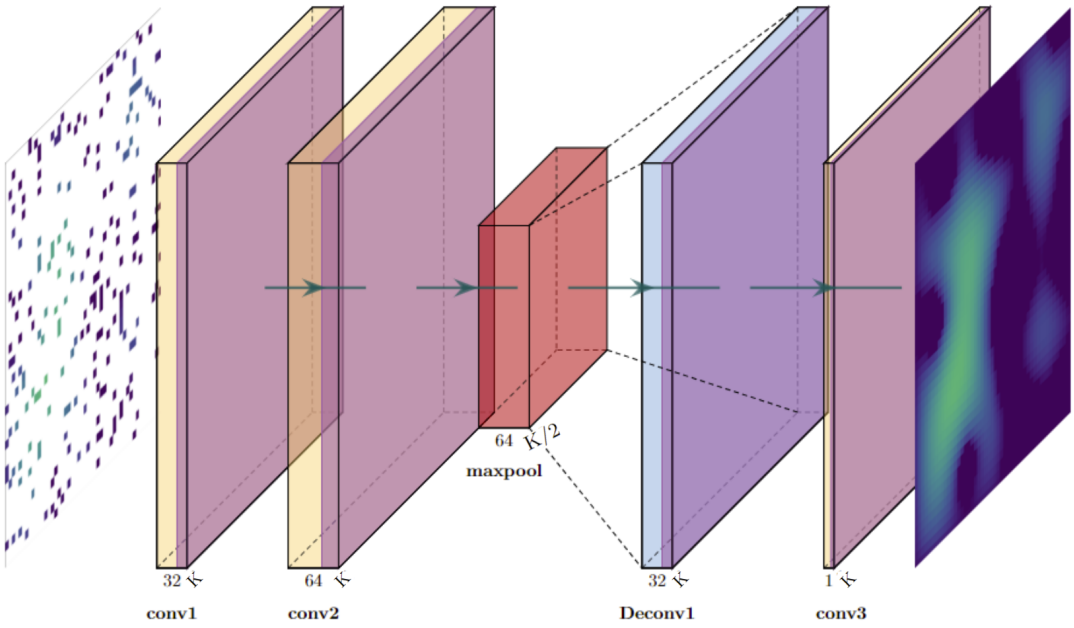
and the distribution of carbon in the bog is equivalent to the distribution of peat mass. By incorporating additional data, the model can be built out to relax this assumption.

The model is constructed from a simple convolutional encoder-decoder architecture. The encoder consists of two convolutional layers with ReLU activations and a final max pooling layer. The decoder similarily consists of a deconvolutional layer and an additional convolutional layer with ReLU activations as seen in Figure 3-2.

An L2 norm between the input and the model output was used as the model reconstruction loss (3.1).

$$\mathcal{L}_{\mathcal{E}} = \frac{1}{K^2} \sum_{x=0}^{K} \sum_{y=0}^{K} \left\| \mathcal{T}_i(x, y, t) - \hat{\mathcal{T}}_{\mathcal{E}_i}(x, y, t) \right\|^2 \tag{3.1}$$

Figure 3-2: Stock Estimator Model Architecture



The model architecture of the stock estimator where the yellow blocks correspond to convolutional layers, the purple shading with ReLU activations, the red block to max pooling, and the blue blocks corresponding to deconvolutional layers. The leftmost image are the samples fed into the model and the rightmost image is the ideal reconstruction. The "K" on the bottom of the blocks represents how the dimensions of the image change after passing through each layer of the network. The number next to the "K" denotes the number of feature maps at each layer.

3.3 Method

A random 80-20 train validation split was used on the individual farm-timestep snapshots to separate the data. The decision was made to have a test set of bog shapes the model hadn't seen to prevent conflating learning with the memorization of shapes in the training set. Consequently the final training, validation, test split is 64%, 16%, and 20%, respectively. When feeding in the sample snapshots into the model, zero padding was used to replace the unsampled areas.

Each model was trained for 30 epochs and the model with the lowest validation loss was used to evaluate performance on the test set (consisting of bog shapes never before seen by the model). The model was trained in all of the experiments with a batch size of 32 and using the Adam optimizer with a learning rate of .001.

To establish a baseline of performance, two dimensional Gaussian process regression or kriging was also evaluated on the dataset. Gaussian process regression fits a function to a set of data points by defining a probability distribution over the set of possible functions and taking the mean of this distribution. In the context of this problem, as this process relies on finding the joint multivariate normal distribution of the sampled and unsampled areas, the choice of covariance kernel is highly specific to the dataset and often requires tuning [8]. A common choice of kernel for spatially related smooth data is the radial basis function kernel which was used to establish the kriging baseline. The covariance between two points x_i and x_j with this kernel is given below in Equation 3.2. $d(x_i, x_j)$ denotes the Euclidean distance and l is the kernel length scale.

$$k(x_i, x_j) = exp\left(-\frac{d(x_i, x_j)^2}{2l^2}\right) \tag{3.2}$$

The bounds on the length of this kernel were determined by the dataset with a length scale bound between almost 0 and $K \times 2$ (which provides an upper bound between the distance between any two points). For smaller sampling densities (ie: 1%) an initial length scale of 4 for each dimension was set to help the model produce better results, all other sampling densities used an initial length scale of 2 for each

dimension. These length scales were chosen from doing a search over integer scales 1, 2, 4, and 8 and choosing scales with the best results. The length parameter is then optimized during fitting. The mean of the predicted distribution upon fitting is then taken to establish the baseline prediction.

An L2 norm between the input and the model output was used as the model reconstruction loss (3.3).

$$\mathcal{L}_{\mathcal{E}} = \frac{1}{K^2} \sum_{x=0}^{K} \sum_{y=0}^{K} \left\| \mathcal{T}_i(x, y, t) - \hat{\mathcal{T}_{\mathcal{E}_i}}(x, y, t) \right\|^2 \tag{3.3}$$

3.4 Experiments

In total a model was trained for each of the exhaustive combinations of sampling strategy (uniform random sampling, grid based sampling) and sampling density (1%, 5%, 10%, 25%) on both the smaller and larger datasets. In addition for each of these combinations, Gaussian process regression as outlined above was performed on every sample in the datasets to generate a baseline for performance.

3.5 Results

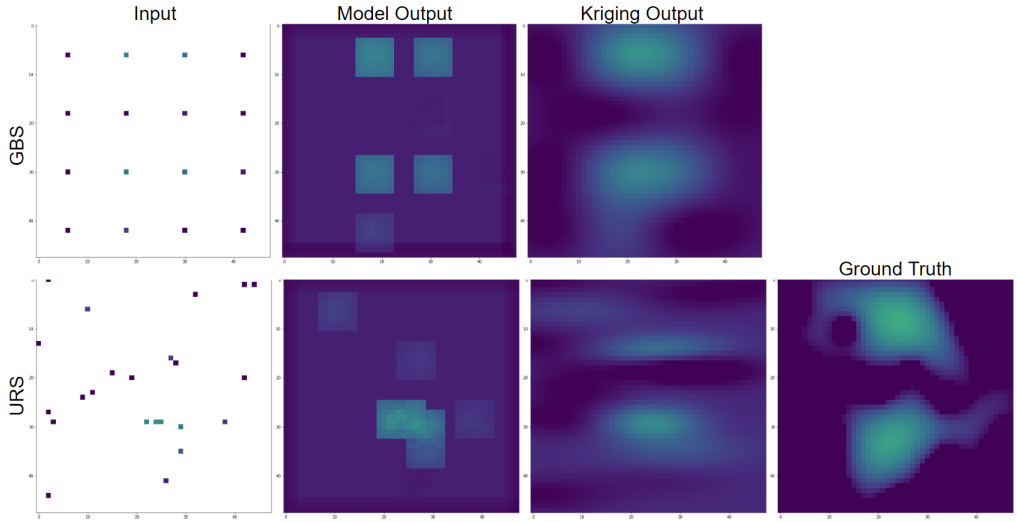
3.5.1 Smaller Dataset Overall Sampling Density Results

Visual examples of the stock estimator results for the 1%, 5%, 10%, and 25% sampling densities can be found in Figures 3-3, 3-4, 3-5, and 3-6 respectively.

Quantitative results for the models trained on uniform random sampling can be found in Table 3.1 and the grid based sampling results in Table 3.2.

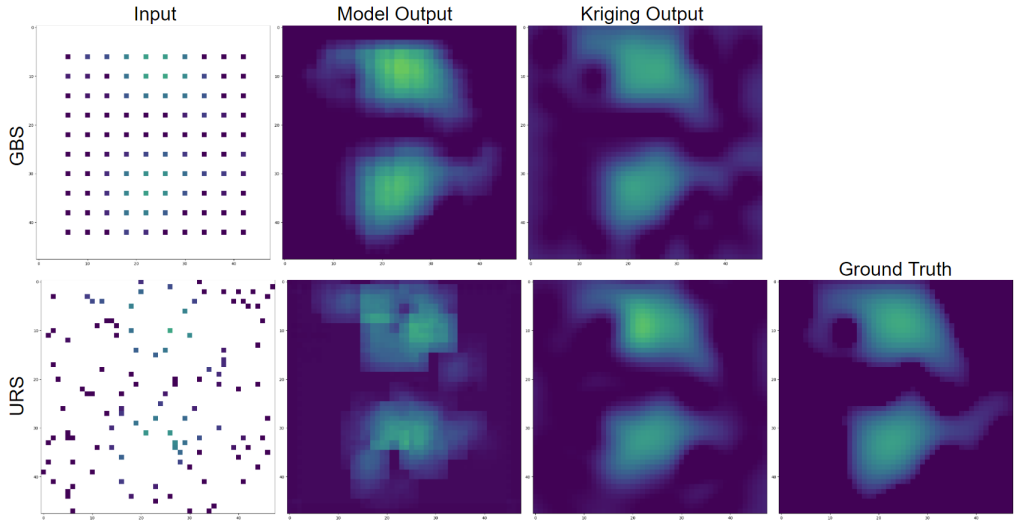
The loss in these tables represents the average per-pixel squared error across all pixels in the dataset. To contextualize this against the pixel values present within our dataset, the URS model test error is 24.2%, 5.1%, 1.6%, and 0.2% of the mean pixel value for the 1%, 5%, 10%, and 25% sampling densities. Similarly, the GBS model test error is 24.4%, 3.6%, 0.2%, and 0.0% of the mean for these respective densities. Ultimately, as shown in these tables, as sample density increases so does

Figure 3-3: Results of 1% Sample on Smaller Test Set Farm 5 Timestep 97



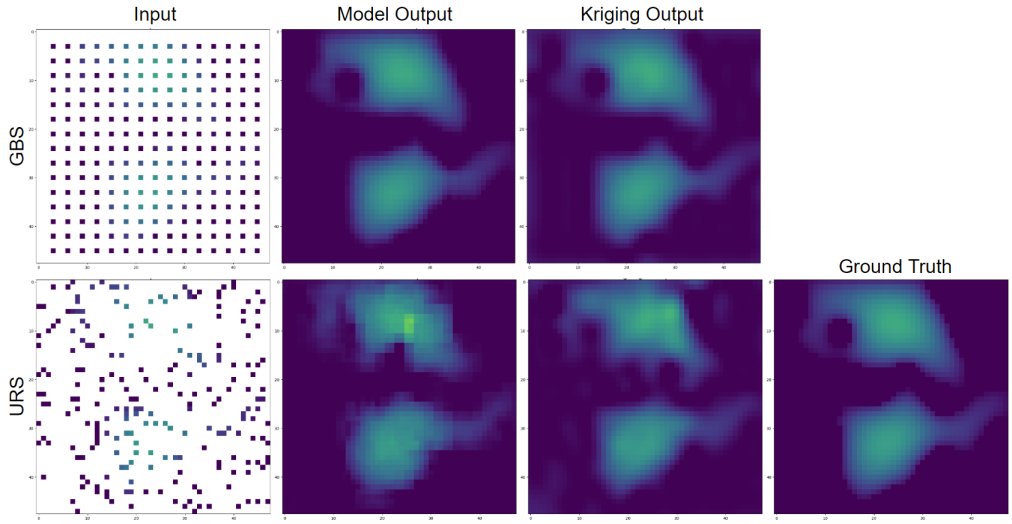
This figure presents results for a 1% sample of farm 5 of the test set at timestep 97. The columns from left to right are the input to the model, the model output, and the kriging model output. The top row corresponds to samples taken through grid based sampling (GBS) and the bottom row with uniform random sampling (URS). The rightmost image is the ground truth from which the samples are taken.

Figure 3-4: Results of 5% Sample on Smaller Test Set Farm 5 Timestep 97



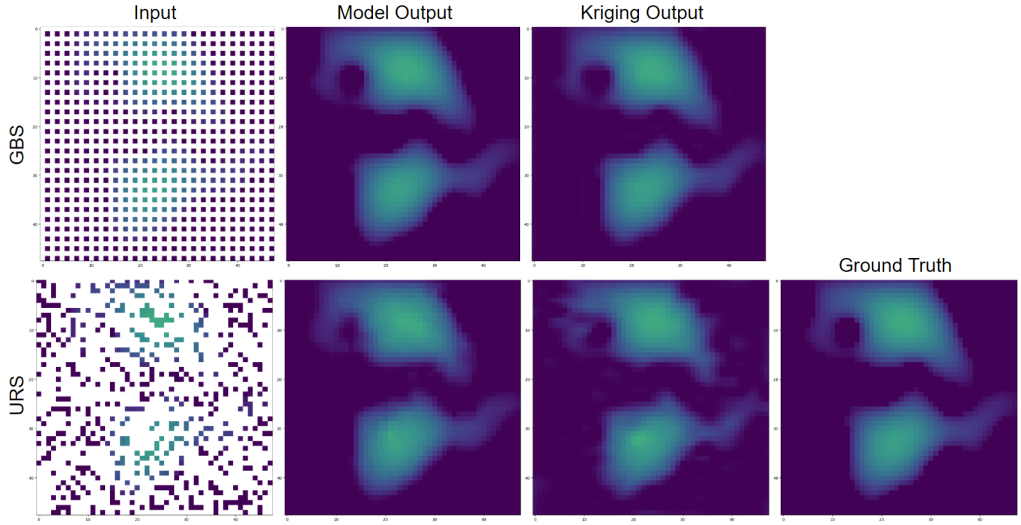
This figure presents results for a 5% sample of farm 5 of the test set at timestep 97. The columns from left to right are the input to the model, the model output, and the kriging model output. The top row corresponds to samples taken through grid based sampling (GBS) and the bottom row with uniform random sampling (URS). The rightmost image is the ground truth from which the samples are taken.

Figure 3-5: Results of 10% Sample on Smaller Test Set Farm 5 Timestep 97



This figure presents results for a 10% sample of farm 5 of the test set at timestep 97. The columns from left to right are the input to the model, the model output, and the kriging model output. The top row corresponds to samples taken through grid based sampling (GBS) and the bottom row with uniform random sampling (URS). The rightmost image is the ground truth from which the samples are taken.

Figure 3-6: Results of 25% Sample on Smaller Test Set Farm 5 Timestep 97



This figure presents results for a 25% sample of farm 5 of the test set at timestep 97. The columns from left to right are the input to the model, the model output, and the kriging model output. The top row corresponds to samples taken through grid based sampling (GBS) and the bottom row with uniform random sampling (URS). The rightmost image is the ground truth from which the samples are taken.

Table 3.1: Smaller Dataset Uniform Random Sampling Model and Kriging Results

Uniform Random Sampling				
	Model Train MSE	Model Validation MSE	Model Test MSE	GPR Test MSE
1%	0.04482	0.04648	0.03616	0.02774
5%	0.00997	0.01011	0.00851	0.00508
10%	0.00263	0.00273	0.00234*	0.00275
25%	0.00038	0.00040	0.00035*	0.00124

This table presents the stock estimator model results trained on the uniform random sampling strategy for the smaller dataset. The rows in the leftmost column indicate the sampling density each model was trained at. MSE stands for mean squared error and GPR stands for Gaussian process regression. The first three columns denote the model performance on the training, validation, and test set. The last column reports the Gaussian process regression mean squared error on the test set. Entries with a * indicate where the model outperformed Gaussian process regression.

Table 3.2: Smaller Dataset Grid Based Sampling Model and Kriging Results

Grid Based Sampling				
	Model Train MSE	Model Validation MSE	Model Test MSE	GPR Test MSE
1%	0.04565	0.04713	0.03650	0.02524
5%	0.00212	0.00219	0.01085	0.00537
10%	0.00034	0.00036	0.00035*	0.00145
25%	0.00002	0.00002	0.00002*	0.00024

This table presents the stock estimator model results trained on the grid based sampling strategy for the smaller dataset. The rows in the leftmost column indicate the sampling density each model was trained at. MSE stands for mean squared error and GPR stands for Gaussian process regression. The first three columns denote the model performance on the training, validation, and test set. The last column reports the Gaussian process regression mean squared error on the test set. Entries with a * indicate where the model outperformed Gaussian process regression.

model performance with the GBS 25% trained model reconstructing unseen data almost perfectly. Even under a smaller data regime, the models trained on the 10% and up sampling density strategies under both sampling regimes were able to outperform kriging. Due to similar performance across the board on the train, validation, and test sets we see that the models were able to converge well without overfitting on the bog shapes it had seen during training.

Visually, the 1% density is too sparse for the model to upsample in a realistic manner and the results are quite pixelated under both uniform and grid-based sampling. This aligns with what is expected. This sampling density is too sparse for the model to memorize the data as many potential bogs could contain the same samples. Although numerically it seems kriging does not perform much better than the model at this density, visually, the model is significantly better. The models begin to pick up on the bog shapes beginning at the 5% density mark and qualitatively the results demonstrate features which are more characteristic of the ground truth, with the realism increasing with sampling density.

3.5.2 Larger Dataset Overall Sampling Results

The same experiment on the smaller dataset was repeated on the larger dataset and the results for uniform random sampling and grid based sampling can be found in Tables 3.3 and 3.4 respectively.

Overall, while performance on the test set increased at the 5% and up level for grid based sampling at the 10% and up level for uniform random sampling, this increase was higher at the 5% GBS level with a 67% decrease in loss (.00736 difference or about 4.3% of the mean pixel value in the test set) however all other increases in performance had a MSE difference of at most .00010 (0.0% of the mean pixel value in the test set). Simply increasing the dataset is not enough to increase performance in this task. The following analysis however is performed on the better performing models trained on the larger dataset.

Table 3.3: Larger Dataset Uniform Random Sampling Model and Kriging Results

Uniform Random Sampling				
	Model Train MSE	Model Validation MSE	Model Test MSE	GPR Test MSE
1%	0.04419	0.04367	0.04357	0.03274
5%	0.00943	0.00937	0.00943	0.00518
10%	0.00228	0.00226	0.00231*	0.00255
25%	0.00025	0.00025	0.00025*	0.00144

This table presents the stock estimator model results trained on the uniform random sampling strategy for the larger dataset. The rows in the leftmost column indicate the sampling density each model was trained at. MSE stands for mean squared error and GPR stands for Gaussian process regression. The first three columns denote the model performance on the training, validation, and test set. The last column reports the Gaussian process regression mean squared error on the test set. Entries with a * indicate where the model outperformed Gaussian process regression.

Table 3.4: Larger Dataset Grid Based Sampling Model and Kriging Results

Grid Based Sampling				
	Model Train MSE	Model Validation MSE	Model Test MSE	GPR Test MSE
1%	0.04475	0.04430	0.04480	0.02378
5%	0.00356	0.00350	0.00349*	0.00521
10%	0.00031	0.00031	0.00033*	0.00123
25%	0.00001	0.00001	0.00001*	0.00022

This table presents the stock estimator model results trained on the grid-based sampling strategy for the larger dataset. The rows in the leftmost column indicate the sampling density each model was trained at. MSE stands for mean squared error and GPR stands for Gaussian process regression. The first three columns denote the model performance on the training, validation, and test set. The last column reports the Gaussian process regression mean squared error on the test set. Entries with a * indicate where the model outperformed Gaussian process regression.

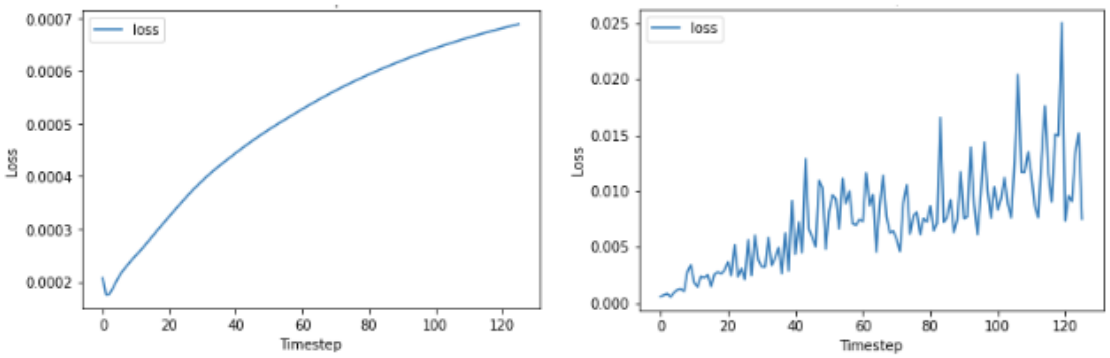
3.5.3 Grid Based Sampling vs Uniform Random Sampling

Overall from the above tables, there is a stark contrast at nearly all levels between uniform random sampling and grid based sampling. A model trained on grid based sampling performs around the same as a model trained on uniform random sampling at the 1% level (with a loss of .04480 vs .04357 on the larger test set). However, past this point sampling strategy does seem to matter. This is visually apparent in the above figures, as the URS results appear much more pixelated and jagged than their GBS counterparts. On the larger dataset, models trained on grid based sampling outperform models trained on uniform random sampling for all remaining sampling densities. The starkest contrast is at the 25% level where the URS MSE is 25 times larger than the GBS MSE on the larger test set. This lends further empirical evidence to how sampling location can dramatically improve results and that optimizing for this location is especially important when sampling capacity is limited. Plotting out the MSE over time as in Figure 3-7 illustrates this point. The dataset is split randomly rather than temporally, yet the GBS MSE curve is much smoother than the URS MSE curve. This graph is generated for a farm at the 10% sampling density however, these results were observed for multiple farms at all densities. One note is that under both sampling regimes the loss does increase over time. Because of the way the peatland develops, it is expected that in earlier timesteps the loss is lower as the range for peat amount at a certain pixel is a lot lower as the surface hasn't developed yet.

Another important note is that because the sampling locations under grid-based sampling never change, whereas the sampling locations under uniform random sampling do, this leads to predictions (in the case of URS) that depend significantly on location and are inconsistent over time. Often, the shape of the farm that is predicted is inconsistent from one timestep to the next, as illustrated in Figure 3-8. In the URS column the bog shape changes based on the location of the samples whereas the GBS results are much more consistent.

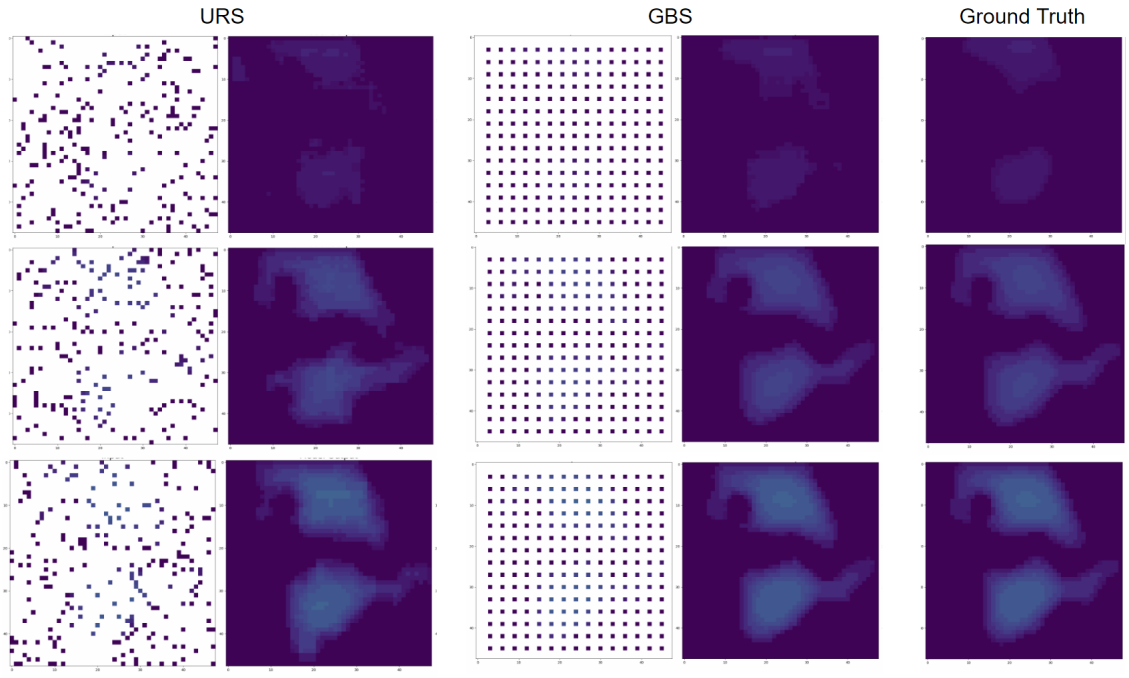
It is possible that taking in a sequence of samples rather than a single sample

Figure 3-7: Loss Over Time: Grid Based Sampling vs Uniform Random Sampling



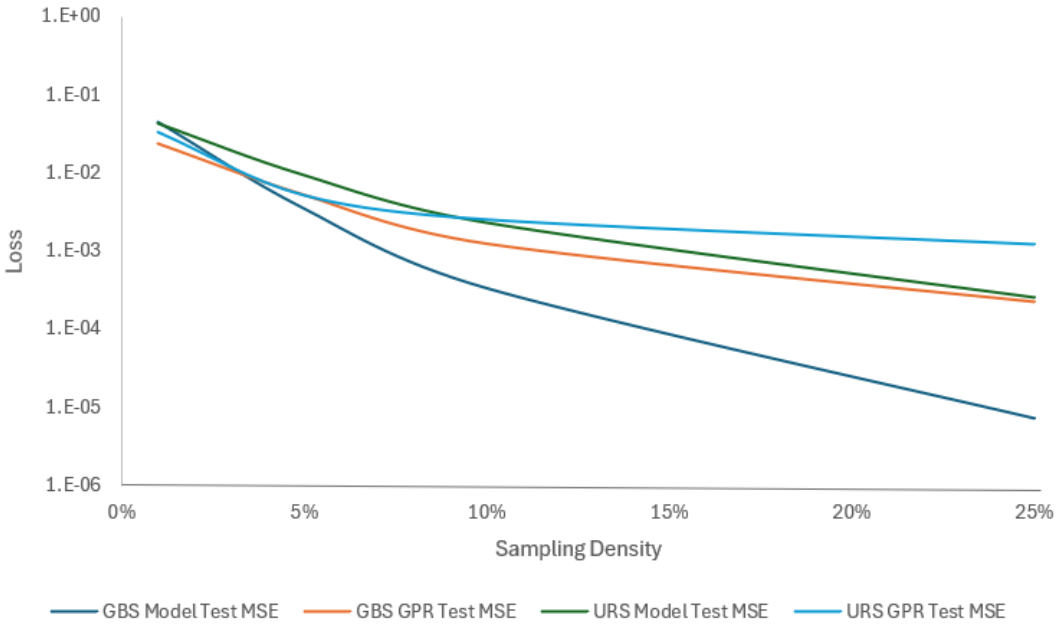
The model MSE loss over simulated timesteps for a farm. Left: Results for a model trained on the 10% GBS sample dataset. Right: Results for a model trained on the 10% URS sample dataset

Figure 3-8: Uniform Random Sampling vs Grid Based Sampling Spatial Prediction Across Time



This figure shows the sample input and model prediction for uniform random sampling in the first column, grid based sampling in the second column, and the ground truth image in the third column. The rows correspond to results at timesteps 0, 20, and 40.

Figure 3-9: Loss for Various Models Trained at the 1%, 5%, 10%, and 25% Sampling Densities



This figure presents the loss for various models at different sampling densities on the larger test set plotted on a log base 10 scale. GBS stands for grid based sampling, URS for uniform random sampling, and GPR for Gaussian process regression.

could improve stock estimation at every timestep, leading to consistent farm carbon boundaries across time, especially due to the differing locations covered through URS. This idea is explored further in Chapter 4.

3.5.4 Performance Against Baseline Kriging

Figure 3-9 compares the the loss on the test set for the GBS and URS models trained at different sampling densities and for kriging. As shown in the figure, kriging outperforms the trained models on the same data up until the 5% point for GBS and the 10% point for URS. At 10% the GBS trained model outperforms the kriging on GBS data and similarly for the URS trained model against kriging on URS data. It is important to note that although the models outperform kriging at this density, the GBS model vastly outperforms kriging compared to the URS model at 10% with a 73% difference in error for GBS and a 9% difference for URS. This smaller difference could potentially be rectified with more careful kernel selection.

A key consideration and further point of comparison is that Gaussian process regression is incredibly computationally expensive, especially as the number of samples and variables increases [19]. The process relies on matrix inversion, which has a cubic time complexity, and while approximation methods have been derived to make the process faster, these are not yet able to achieve scale. For example, each of the smaller models presented above took around 30 minutes to train on a CPU and 2 minutes to evaluate on the test set to produce predictions, with both timescales being independent of sample density. In contrast, running the Gaussian process regression at the 1%, 5%, 10%, and 25% densities took around 2 minutes, 20 minutes, 90 minutes, and 240 minutes respectively.

3.6 Conclusions and Future Work

The model provides a less computationally expensive alternative to traditional kriging under certain sampling regimes. Even without access to temporal information, the models trained for the stock estimator are able to outperform the baseline kriging for grid based sampling under the 5% sampling regime and for both grid based sampling and uniform random sampling under the 10% sampling density regime. At the highest sampling density of 25% the models had a loss of 0% and .1% of the mean pixel value of the test set, successfully reproducing a dense carbon map from the samples.

As noted above, no hyperparameter tuning was conducted on the stock estimator models. Further tuning and use of multiple random seeds could result in better overall model performance, especially at lower sampling densities, and a clearer understanding of relative performances. Currently, the model has access to the noiseless ground truth, which is not a realistic condition. Further work should involve the addition of noise to the simulation to be more representative of real world conditions. The model could potentially benefit from additional depth and complexity. Additionally, as the dataset is created as the solution to a differential equation, further research can be conducted into using an Implicit Neural Representation to reconstruct the carbon map more accurately for this dataset.

Chapter 4

Change Predictor

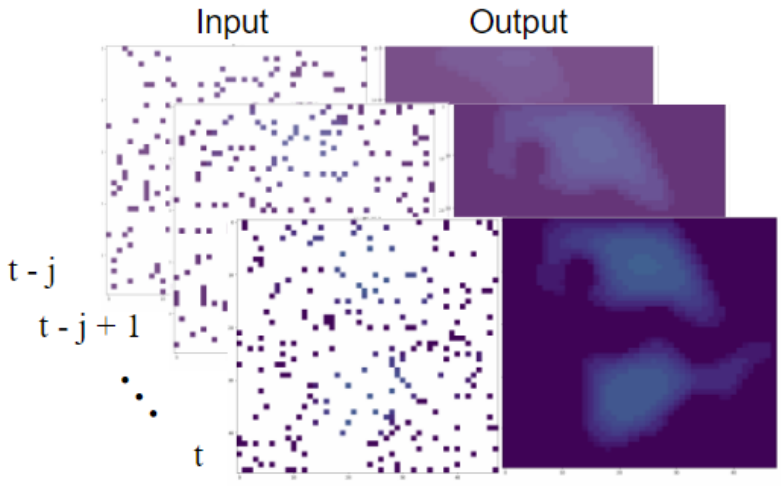
While being able to generate a carbon stock estimate at a timestep is important for carbon accounting and regulatory compliance, being able to extend past this to also predict the change in carbon in the future given a limited sample size can be helpful to inform land use policy. This section proposes an end-to-end stock estimator and change predictor which given a sampling history, generates the estimates for each timestep in this history alongside a dense carbon map prediction for some timesteps in the future. This task can be broken down into two smaller parts. First, given a sequence of samples to infill these samples and second given the estimated carbon density maps generated from the infillings to predict carbon maps in future timesteps. Ultimately, our goal is to have the samples $X^{i,t}$ inputted to this end-to-end estimator and predictor be the set of samples outputted by the sampling optimizer $\mathcal{S}$ described in the next chapter.

4.1 Sequential Stock Estimator

4.1.1 Model Architecture

The stock estimator trained in the previous chapter provides a good baseline for the performance we can expect with infilling, as given a sequence of sampling history one can pass in each image through the single timestep stock estimator. As discussed in

Figure 4-1: Sequential Estimator Inputs and Outputs

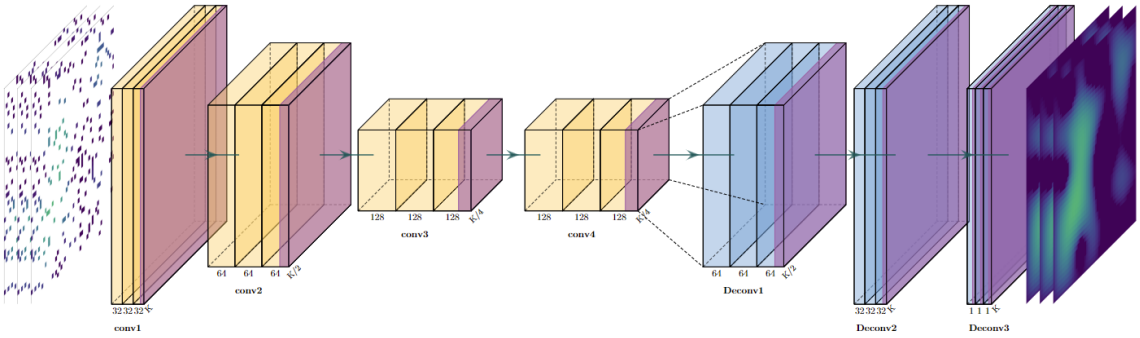


This figure presents an example input and output of the sequential estimator models. The model takes in a sequence of samples starting at timestep t-j up until timestep t, and outputs the soil carbon estimation for each of these timesteps.

Chapter 3, due to the new temporal dimension of data a more complicated model architecture could leverage the temporal relations between samples for better performance. A 3D CNN was trained to generate dense carbon maps, given a sequence of samples. These sequential stock estimators are defined as they were in the previous chapter, however they now take in a sequence of samples at farm i from some timestep $t-j$ until some timestep t (including sequence length one, as with the original estimator) $X^i_{j,t} = \{X^{i,t-j}, X^{i,t-j+1}, ..., X^{i,t}\}$ and output a sequence of stock estimates for each timestep from $t-j$ to t, $\mathcal{E}^i_{j,t} = \{\mathcal{E}^i_{t-j}, \mathcal{E}^i_{t-j+1}, ..., \mathcal{E}^i_t\}$. The expected input and output of this sequential stock estimator is presented in Figure 4-1. The first three convolutional layers additionally have batch norms applied to them.

The 3D CNN model is a direct expansion of the original stock estimator in Chapter 3, where the kernels are instead 3D to perform convolutions across timesteps as well. While the model is still a standard encoder-decoder structure, it differs from the one presented in Chapter 3 as there are an increased number of feature maps to handle the temporal dimension and strided convolutions are used rather than a max pooling layer to keep more finegrained information at higher computational cost.

Figure 4-2: Sequential Estimator Model Architecture



The model architecture of the sequential stock estimator where the yellow blocks correspond to convolutional layers, the purple shading with ReLU activations, and the blue blocks to deconvolutional layers. The leftmost images are the sequence of samples fed into the model and the rightmost images are the ideal reconstruction. The "K" on the bottom of the blocks represents how the dimensions of each image in the sequence changes after passing through each layer of the network. The number next to the "K" denotes the number of feature maps at each layer.

The encoder consists of three 3D convolutional layers with ReLU activations with increasing feature maps. There is a convolutional bottleneck layer between the encoder and the decoder. Finally, the decoder consists of three deconvolutional layers to bring the representation back into a one channel sequence form as seen in Figure 4-2.

4.1.2 Method

All timesteps from the first 90% of farms from the larger training-validation set were used for training and the remaining 10% of farms were used for spatial validation. All farms from the larger generated test set were used for evaluation. Given an infilling window $L = j + 1$, we divide the data for each farm into multiple pairs of input sequences and target sequences of size L. The input sequences are the sampled sequences and the target sequences are the corresponding ground truth dense carbon maps. In order to split this data into sequences, an overlapping rolling window method similar to that of the previous section (Algorithm 2) was used. In addition, random horizontal and vertical flips were applied to the training set to augment it.

Each model in the experiments was trained for 20 epochs and the model with the

Table 4.1: Sequential Estimator Test Set MSE Results

	1% Density		5% Density		10% Density		25% Density	
L	GBS	URS	GBS	URS	GBS	URS	GBS	URS
1	0.01759	0.02032	0.00071	0.00149	0.00007	0.00038	0.00001	0.00011
2	0.01766	0.00825	0.00072	0.00037	0.00007	0.00013	0.00002	0.00005
5	0.01794	0.00179	0.00073	0.00013	0.00006	0.00007	0.00001	0.00003
10	0.01778	0.00075	0.00074	0.00009	0.00008	0.00005	0.00002	0.00003
15	0.01752	0.00057	0.00074	0.00008	0.00007	0.00005	0.00001	0.00003
20	0.01764	0.00051	0.00072	0.00008	0.00007	0.00005	0.00002	0.00003

This table presents the mean squared error of the sequential estimator on the test set for various sampling densities, strategies and input lengths L. URS stands for uniform random sampling and GBS stands for grid based sampling.

lowest validation loss was used to evaluate performance on the test set. All models were trained with a batch size of 32 and using the Adam optimizer with a learning rate of .0001. As in Chapter 3, the L2 norm between the model output and ground truth was used for the reconstruction loss. The only difference is that this loss is taken over the entire sequence, rather than at a single timestep (4.1).

$$\mathcal{L}_{\mathcal{E}} = \frac{1}{L \times K^2} \sum_{t=t-j}^{t} \sum_{x=0}^{K} \sum_{y=0}^{K} \left\| \mathcal{T}(x, y, t) - \hat{\mathcal{T}}_{\mathcal{E}_i}(x, y, t) \right\|^2 \tag{4.1}$$

4.1.3 Experiments

The model was trained on the exhaustive combination of sampling strategies, densities, and input lengths L in [1, 2, 5, 10, 15, 20]. These results were compared to the baseline established in Chapter 3 to determine the effectiveness of sequential sampling for infilling.

4.1.4 Results

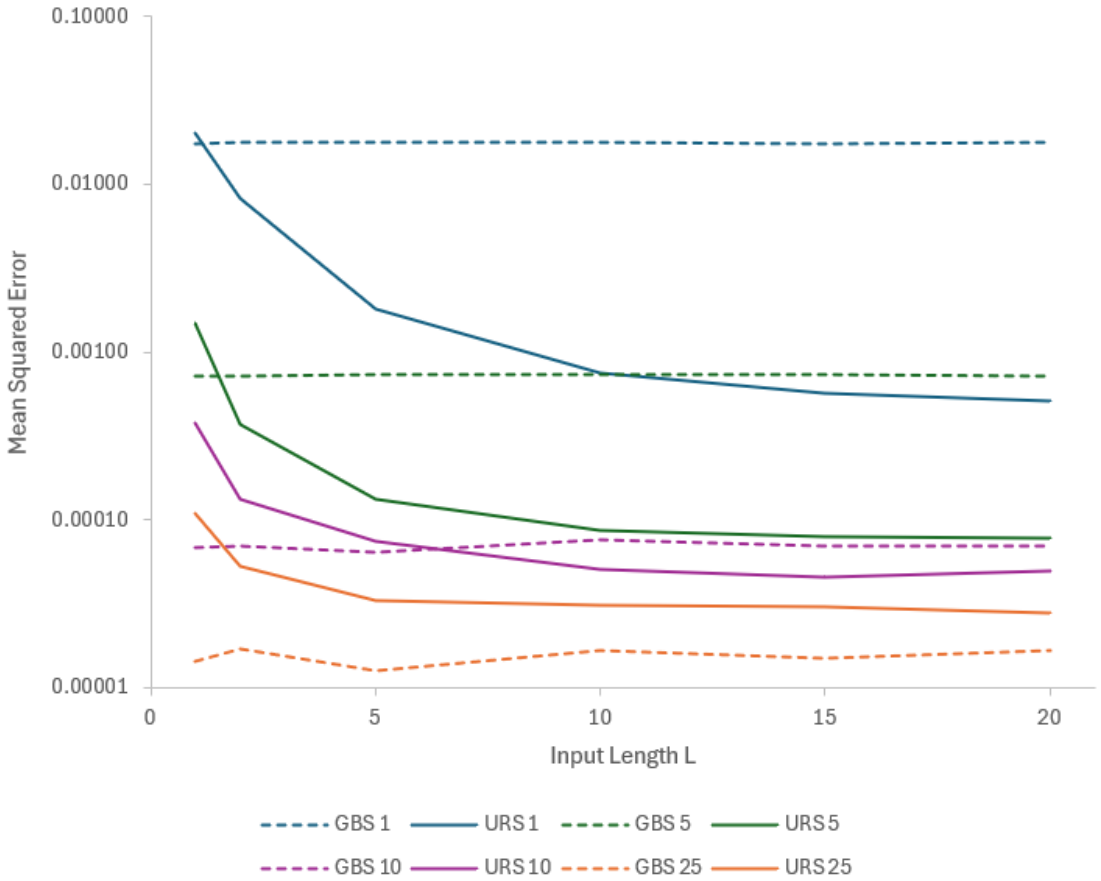
The table containing the MSE on the test set for the exhaustive parameters combinations is presented in Table 4.1. Although the training:validation data split is slightly different, first 90% of farms for training and the remaining 10% for validation vs. the

80:20 random split across all farm time snapshots between the models, we see that overall this architecture greatly outperforms the one presented in Chapter 3 at the same $L = 1$ on the same test set for all sampling densities. It outperforms kriging for all sampling densities and strategies (Tables 3.3 and 3.4). In particular, for $L = 1$ the model with the worst parameters (URS at the 1% sampling density) has an MSE of .02032 which is 12% of the mean pixel error of the test set whereas the model with the best parameters (GBS at the 25% density) has an overall test MSE of .00001 which is 0% of this mean pixel error.

An observation to note is that uniform random sampling consistently outperforms grid based sampling as soon as there are at least two historical samplings, except for in the case of 25% sampling density where grid based sampling outperformed uniform random sampling at all input lengths. While the general shape of the peatland and thus carbon distribution does not significantly vary in the dataset (which is not necessarily the case with realistic soil carbon distrbutions), this provides a method of leveraging uniform random sampling versus the more traditionally used grid based sampling with historical data to achieve better carbon estimation. Figure 4-3 compares the performance of uniform based sampling and grid based sampling with different sampling densities as the sequence input length increases. Overall as seen from Figure 4-3, as the input length L increases model performance increases for uniform random sampling regimes, but not for grid based sampling regimes, encouraging varied location sampling given historical data. However, the benefits of increasing the historical data input for URS tapers out for $L \geq 10$, with the greatest performance boost occuring when the input length increases from one to two.

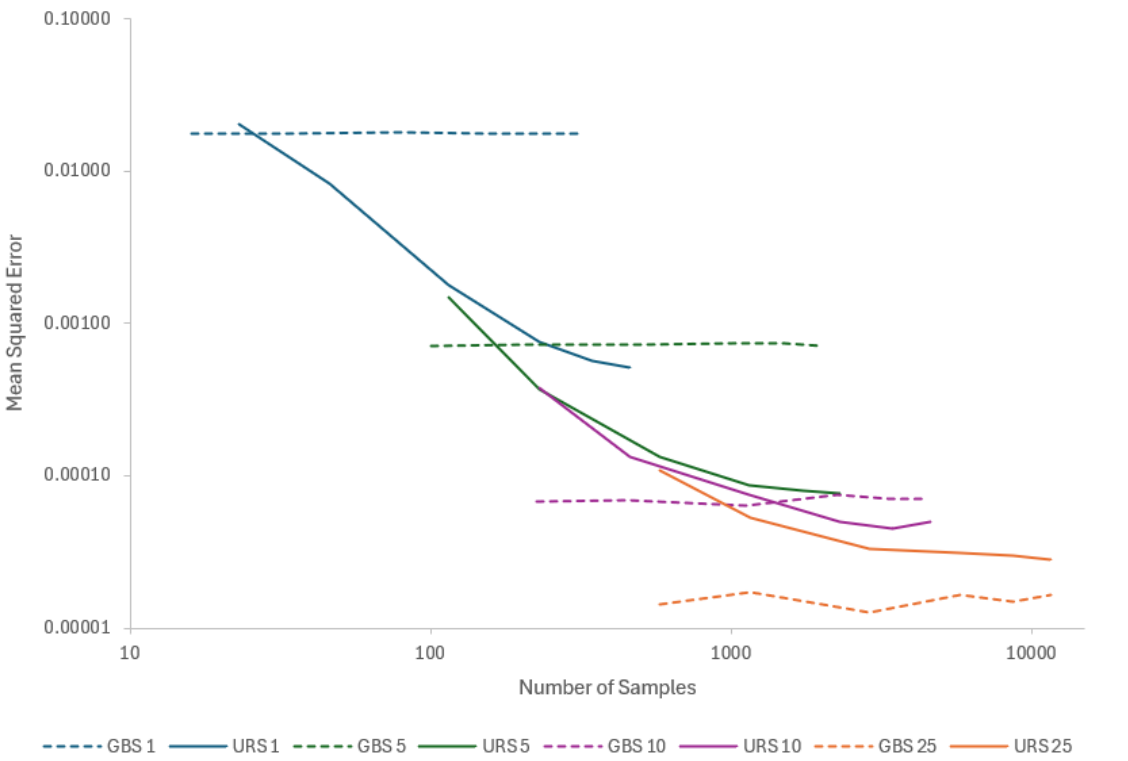
Another note is that models trained at a lower density were able to outperform models trained at a higher density given enough historical data. For example, uniform random sampling at 1% was able to achieve a lower test MSE for $L \geq 15$ than grid based sampling at the 5% level when $L \leq 10$ and uniform random sampling at the 5% level when $L = 1$. However, it is important to consider the cost of gathering this historical data. Figure 4-4 presents the mean squared error as a function of number of samples (a function of input length, sampling strategy, and sampling density) rather

Figure 4-3: Sequential Estimator MSE over Input Lengths for Various Sampling Strategies



This figure presents the log base 10 scale MSE on the test set of the sequential estimator model for the combination of different sampling strategies and densities as the input length L increases. URS stands for uniform random sampling, GBS stands for grid based sampling, and the number next to the strategy represents the sampling density (1%, 5%, 10%, or 25%). URS models are all plotted with solid lines and GBS models are plotted with dashed lines. Models trained on the same sampling density share the same color.

Figure 4-4: Sequential Estimator MSE over Sample Number for Various Sampling Strategies



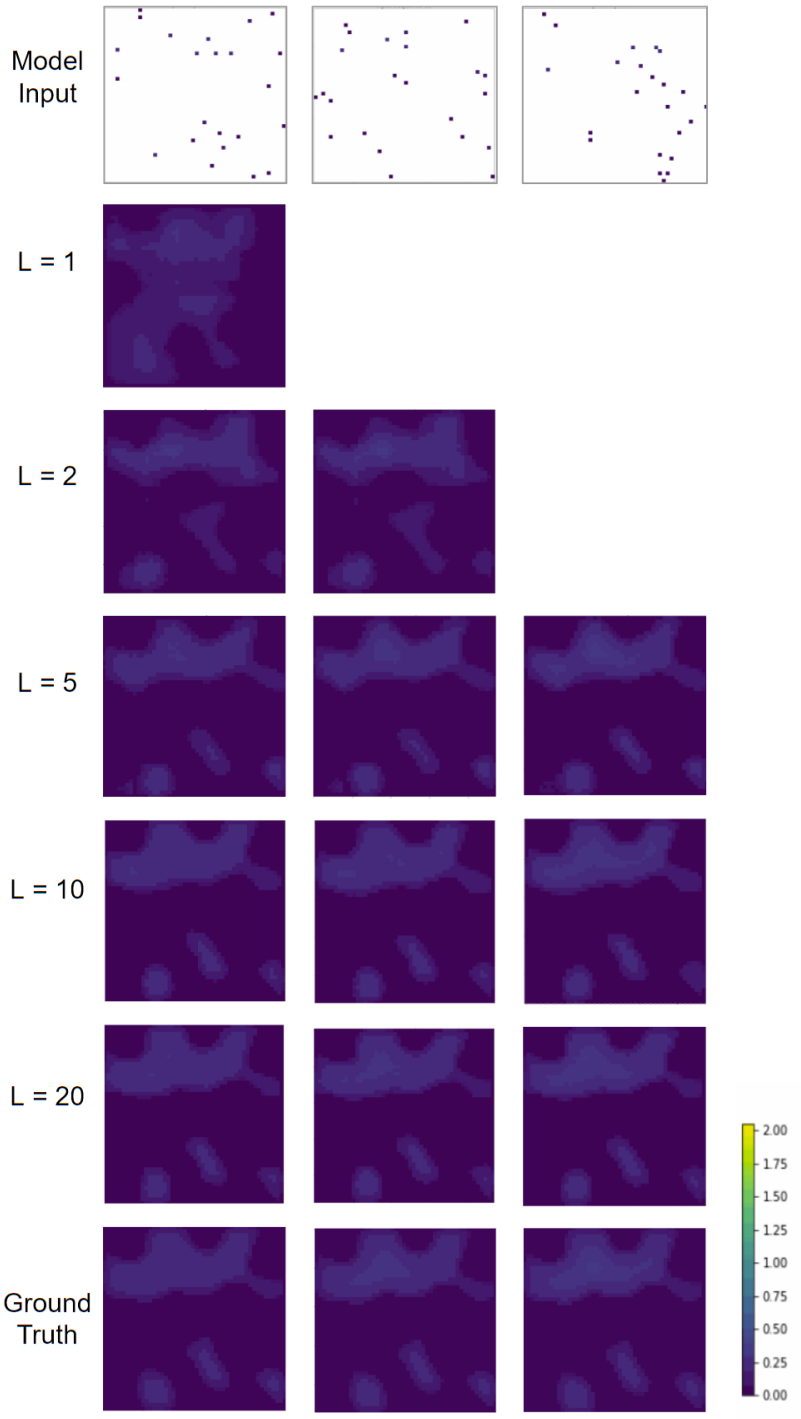
This figure presents the log base 10 scale MSE on the test set of the sequential estimator model for the combination of different sampling strategies and densities as the total number of samples increases (also on a log base 10 scale). URS stands for uniform random sampling, GBS stands for grid based sampling, and the number next to the strategy represents the sampling density (1%, 5%, 10%, or 25%). URS models are all plotted with solid lines and GBS models are plotted with dashed lines. Models trained on the same sampling density share the same color.

than input length L. From this figure, we see that more total samples at lower densities are required to make up this difference and that generally higher density sampling at a timestep is more effective; however at 230 samples (URS 5% where $L = 2$ and URS 10% $L = 1$) sampling at 5% outperforms sampling at 10% by .00001. For example, we find that at URS 1% 230 samples over 10 timesteps are needed to achieve a lower test MSE than the 115 samples gathered at one timestep for URS 5%.

Strictly speaking, as the cost of sampling is so high to solve the sequential infilling problem this cost may not be justified. However future stock prediction is important and the change predictor heavily depends on the sequential infilling performance as well as the length of historical data. Thus, this error could propagate further incentivizing the gathering of less dense historical samples, rather than a single dense sampling as what has been traditionally done. This is explored in the last section of this chapter, which combines the sequential estimator and change predictor into a single end-to-end model.

A visual example of how the results of the sequential estimator change depending on the input length L is presented in Figure 4-5. This figure is for the models trained on the 1% sampling density data, which is the density that showed the most improvement as L scaled. As shown in the figure, the visual jump in output from one sample input to two sample inputs is very dramatic. With only one sample snapshot, the model is unable to estimate the shape of the peat bog but with two sample history snapshots, the model is able to get much closer. Even at the 1% level with only one sampling snapshot, the result is a large improvement from the architecture in Chapter 3 as visualized by Figure 3-3. Overall, as visualized and quantified, we are able to improve on stock estimation by leveraging temporal information and reconstruct the soil carbon map of these bogs very accurately even at extremely low sampling densities and smaller input lengths.

Figure 4-5: 1% URS Sequential Estimator Test Set Sequence Results



This figure presents the results of the sequential estimator on a farm in the test set for a variety of input lengths L. The length of the sequences have been visually cropped to 3 for conciseness and better comparison, however the length of the model input and model output is L.

4.2 Change Predictor

4.2.1 LSTM Models for Seq-to-Seq Prediction

Sequence-to-sequence learning using an encoding and decoding long short term memory models (LSTMs) was first proposed by Sutskever et al. [25] with the main result being on a language translation task. The results were successful, and efforts to use modify this architecture to work with images for natural phenomena prediction such as the motion of sea ice have also born fruit [17]. Similarly the problem of predicting peat growth can also be modeled using a convolutional sequence-to-sequence LSTM framework where the input sequence is a series of past carbon maps and the output sequence represents future carbon map predictions. An encoder-decoder CNN-LSTM based off of the above work was used for this change predictor seq-to-seq prediction task.

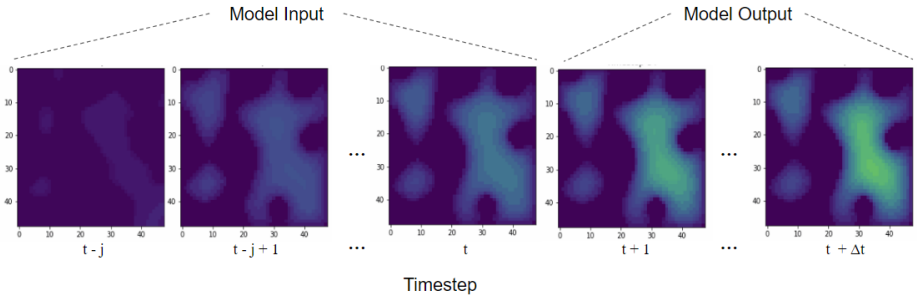
4.2.2 Model Architecture

A **change predictor** $\mathcal{P}$ which takes as input a sequence of superresolution estimates $\mathcal{E}^i_{j,t} = \{\mathcal{E}^i_{t-j}, \mathcal{E}^i_{t-j+1}, ..., \mathcal{E}^i_t\}$ created by the stock estimator from timestep $t-j$ up until timestep t. The output of the stock estimator $\mathcal{E}^i_t$ for farm i at unspecified time t given samples $X^{i,t}$ is: $\mathcal{E}[X^{i,t}] = \hat{\mathcal{T}_{\mathcal{E}_i}}(x, y, t)\ \forall x, y \in [0, K-1]$. The change predictor then predicts the superesolution images of the stock for timesteps between t and $t + \Delta t$ denoted as $\hat{\mathcal{T}}_{\mathcal{P}_i}(x, y, t)$ for all $t \in [t, t + \Delta t]$. Thus the model takes in a sequence of length $L = j + 1$ of stock estimates sized $1 \times K \times K$ and outputs a sequence of length $\Delta t = P$ estimate predictions.

As the purpose of the change predictor is to predict on timesteps for which samples have not yet been gathered, the performance of this module is highly dependent on the estimates output by the sequential stock estimator. The best possible input into this module is the ground truth, and consequently the model was trained on this data. A visual guide to the potential input and output of the model is in Figure 4-6.

The model architecture is an encoder-decoder CNN-LSTM containing the same

Figure 4-6: Change Predictor Inputs and Outputs



This figure presents an example input and output of the change predictor model. The model takes in a sequence of stock estimates or ground truth snapshots starting at timestep $t-j$ until timestep t and predicts dense carbon map from timestep $t+1$ until $t+\Delta t$

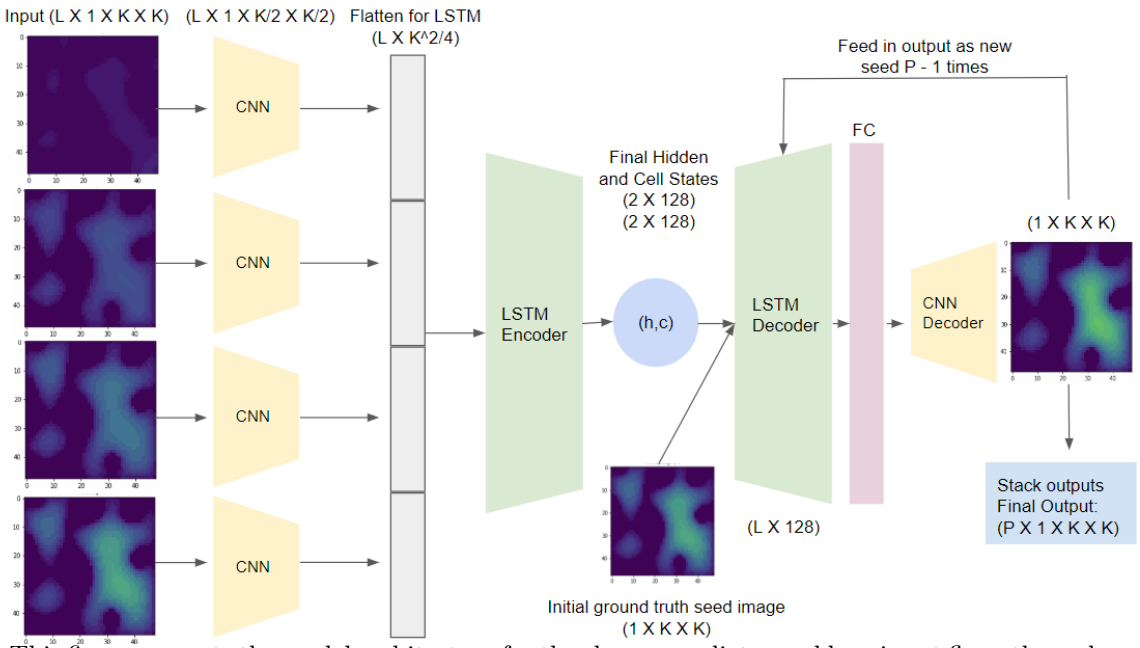
convolutional layers to the latent space as the Chapter 3 stock estimator CNN (2 convolutional layers with ReLU activations and a max pool); however, the output of the max pooling layer is passed into an encoding LSTM, to a decoding LSTM, to a fully connected layer before being passed through the rest of the original stock estimator network (consisting of a deconvolutional layer and another convolutional layer with ReLU activations) to generate an output. This output is fed back into the decoder LSTM an additional $\Delta t - 1$ times, where Δt is the length of the forecast horizon. The outputs from the decoder are stacked together to produce the final prediction output: a $P \times 1 \times K \times K$ sequence as visualized in Figure 4-7.

4.2.3 Method

The first 80% of timesteps of 90% of the farms from the larger training-validation set were used for training. The remaining 20% of timesteps from the first 90% of farms along with the remaining 10% of farms were used for validation. The timesteps were split along with individual farms to allow for both temporal and spatial validation. The test data was taken as-is, with overlap with the validation set temporally.

In order to split this data into sequences for training and evaluation two methods were used. Given a window of historical data of size $L = j+1$, and a forecast horizon of size $P = \Delta t$, we divide the data for each farm into multiple pairs of input sequences of length L and target sequences of length P which maintain the temporal relation

Figure 4-7: Change Predictor Model Architecture



This figure presents the model architecture for the change predictor and how input flows through the model. The leftmost images represent the input to the model that are fed to a CNN and LSTM to encode the features. The final hidden and cell states of this encoding are represented by (h, c). This alongside a seed image (the last ground truth image of the sequence) are passed to the decoder. FC represents the fully connected layer. The dimensions at the top/bottoms of the layers represents the size of the layer output.

of the data. Two methods are tried: one without overlapping windows (Algorithm 1) and one with overlapping windows (Algorithm 2) to increase data especially for a high L and P.

Algorithm 1 Rolling window with no overlap

Require: $L \geq 1$
  $inputs, targets \leftarrow [], []$
  $ii \leftarrow 0$
  while $ii \leq timesteps - L - P$ **do**
    $inputs \leftarrow inputs + farm\_carbon[ii : ii + L]$
    $targets \leftarrow inputs + farm\_carbon[ii + L : ii + L + P]$
    $ii \leftarrow ii + L + P$
  end while

Here the farm carbon snapshots stored in farm_carbon are split into input sequences of length L, and target sequences of length P. Inputs and targets hold the final divided sequences and timesteps represents the total amount of simulated farm snapshots for the farm.

Algorithm 2 Rolling window with overlap

Require: $L \geq 1$
  $inputs, targets \leftarrow [], []$
  $ii \leftarrow 0$
  while $ii \leq timesteps - L - P$ **do**
    $inputs \leftarrow inputs + farm\_carbon[ii : ii + L]$
    $targets \leftarrow inputs + farm\_carbon[ii + L : ii + L + P]$
    $ii \leftarrow ii + 1$
  end while

Here the farm carbon snapshots stored in farm_carbon are split into input sequences of length L, and target sequences of length P. Inputs and targets hold the final divided sequences and timesteps represents the total amount of simulated farm snapshots for the farm.

These algorithms are run on the farm-timestep snapshots once they have been separated out as described above to form the data the models are trained and evaluated on. The model was trained on solely the larger of the datasets, taking in a sequence input and producing a sequence output with one channel. Both the encoding and decoding LSTM in the model used in the experiments contain 2 layers with a hidden size of 128. Due to the simplicity of the data, no tuning of this parameter was performed. Each model in the experiments was trained for 30 epochs and the

model with the lowest validation loss was used to evaluate performance on the test set. All models were trained with a batch size of 32 and using the Adam optimizer.

While training each of the models, teacher forcing, a technique to feed the ground truth of the model and effectively backpropagate at each timestep to prevent divergence, was employed in order to accelerate model convergence. In addition, data augmentations were applied to the training dataset, consisting of random horizontal and vertical flips as well as rotations in order to increase spatial variety and prevent the model from overfitting on spurious spatial features in the training set.

As in Chapter 3, the L2 norm between the model input and output was used for the reconstruction loss. The only difference is that this loss is taken over the entire sequence, rather than at a single timestep (4.2).

$$\mathcal{L}_{\mathcal{P}} = \frac{1}{\Delta t \times K^2} \sum_{t=t}^{t+\Delta t} \sum_{x=0}^{K} \sum_{y=0}^{K} \left\| \mathcal{T}(x, y, t) - \hat{\mathcal{T}_{\mathcal{P}_i}}(x, y, t) \right\|^2 \tag{4.2}$$

4.2.4 Experiments

Multiple models were trained using the infrastructure described above with varying historical input length L and varying forecast horizon length P. The values considered for L included 1, 2, 5, 10, 15, and 20 and for P included 1, 5, 10, and 15. Additionally, two learning rates were tested: .001, and .0001. A grid search was used to train a model on the exhaustive combination of these hyper parameters on both the non-overlapping and overlapping window datasets.

4.2.5 Results

The results of this hyperparameter search on both datasets trained with a learning rate of .001 can be found in Table 4.2. The forecast horizon only goes up to 10 timesteps, as compared to 15 for the lower learning rate. As we can see from these results, the learning rate seemed to be too large for the data with inconsistent and surprising results. One would expect that as the prediction horizon increases that the error also increases, however there are multiple instances where this is not the

Table 4.2: Change Predictor Test Set MSE Results LR = .001

Input Length L	Prediction Length P = 1		Prediction Length P = 5		Prediction Length P = 10	
	No Overlap	Overlap	No Overlap	Overlap	No Overlap	Overlap
1	0.00128	0.00102	0.00628	0.00024	0.00570	0.00789
2	0.00141	0.00120	0.00108	0.00024	0.01298	0.00591
5	0.00192	0.00039	0.00115	0.00245	0.00707	0.00027
10	0.00208	0.00022	0.00334	0.00022	0.01543	0.00025
15	0.00195	0.00074	0.00367	0.00042	0.01532	0.00024
20	0.00245	0.00128	0.00953	0.00033	0.02364	0.00180

This table presents the results of the hyperparameter sweep on the lengths of the inputs and outputs of the change predictor model run with a learning rate of .001. Each column presents the per pixel mean square error on the overlapping and non overlapping dataset for each of the prediction horizons P being either 1, 5, or 10. Each row in said column corresponds to the length of the ground truth sequence input L ranging between 1 and 20.

case (ie: when $L = 2, P = 1$ vs $P = 5$ for both the overlapping and non overlapping datasets). Training on the overlapping dataset greatly improved performance for larger sequences when $L \geq 10$ and $P \geq 5$, for all combinations. In the no overlap data regime when $L = P = 1$, there are a total of 50 sequences per farm, for a total of 22500 sequences. In comparison, when $L = 20$, $P = 10$ there are a total of 3 sequences per farm in the training set, for a total of 1350 sequences. Under the overlapping data regime when $L = P = 1$, there are a total of 98 sequences per farm, for a total of 44100 sequences and when $L = 20$, $P = 10$ there are a total of 70 sequences per farm, for a total of 31500 sequences. In this case, as overlapping windows provide many more training examples equalizing the imbalance especially at large Ls and Ps, it is no surprise that increasing the available data highly improves model performance. A noted potential consequence of using overlapping windows is that it can quickly overfit training data, however it has not done so in this case.

When decreasing the learning rate, much more consistent and accurate results are found. The results with a smaller learning rate of .0001 are presented in Table 4.3. We see a similar pattern that for large Ls and Ps that training on the overlapping

Table 4.3: Change Predictor Test Set MSE Results LR = .0001

Input Length L	Prediction Length P = 1		5		10		15	
	No Overlap	Overlap	No Overlap	Overlap	No Overlap	Overlap	No Overlap	Overlap
1	0.00047	0.00030	0.00140	0.00032	0.00458	0.00041	0.00978	0.00067
2	0.00056	0.00026	0.00144	0.00028	0.00400	0.00036	0.00849	0.00045
5	0.00084	0.00026	0.00170	0.00027	0.00438	0.00036	0.00856	0.00042
10	0.00138	0.00025	0.00237	0.00027	0.00537	0.00034	0.00979	0.00042
15	0.00157	0.00026	0.00307	0.00029	0.00663	0.00034	0.01089	0.00046
20	0.00220	0.00028	0.00421	0.00029	0.00783	0.00034	0.01153	0.00043

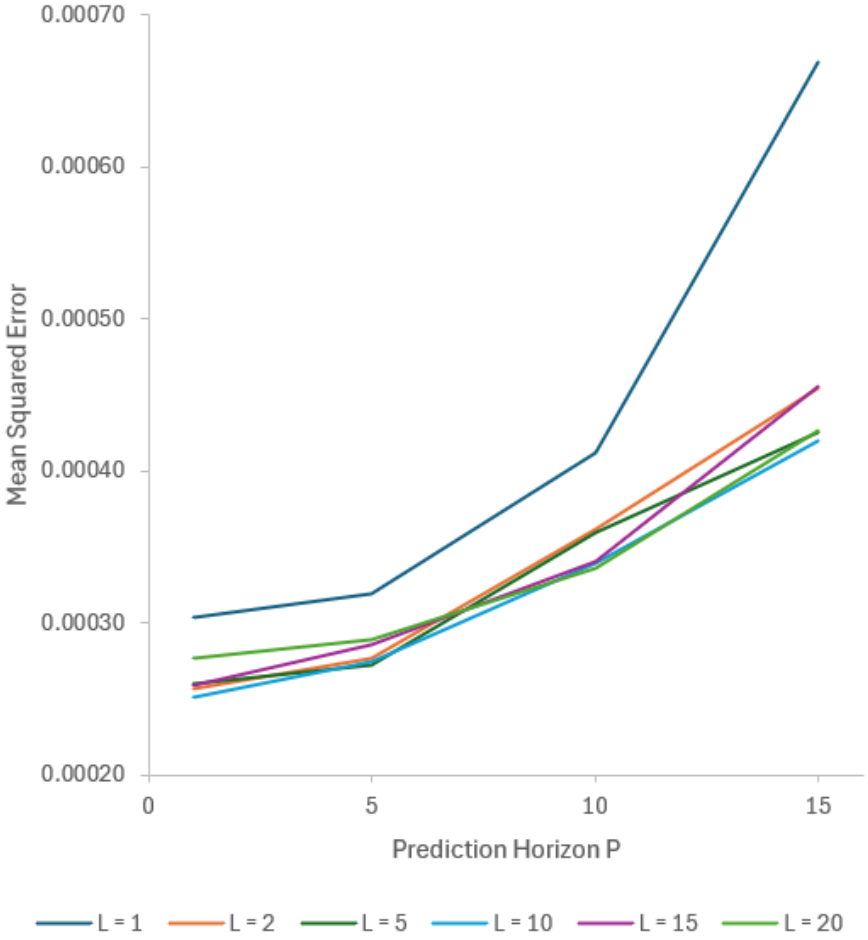
This table presents the results of the hyperparameter sweep on the lengths of the inputs and outputs of the change predictor model run with a learning rate of .0001. Each column presents the per pixel mean square error on the overlapping and non overlapping dataset for each of the prediction horizons P being either 1, 5, 10, or 15. Each row in said column corresponds to the length of the ground truth sequence input L ranging between 1 and 20.

dataset greatly improves test performance and the models trained on the overlapping dataset with a smaller learning rate of .0001 better than the rest of the models at all input and prediction lengths with the singular exception of the models trained with the larger learning rate when $P = 5$ and $L \leq 10$. In this case the largest difference in MSE performance is .00008 which is 0.05% of the mean pixel value of the test set, a very negligible difference. Thus the following analysis is performed on the best performing models: models trained with overlapping data and a smaller learning rate.

The highest MSE is for unsuprisingly when we try to predict 15 timesteps ahead with only 1 timestep worth of historical data. However, the MSE is only .00067 or about 00.39% of the mean pixel value of the test set. The lowest MSE occurs on the model that is trained to predict one timestep in the future with 10 timesteps worth of historical data. However, for all input lengths the results of predicting 1 or 5 timesteps in the future are all within $\pm.00007$ of each other. The change predictor is able to learn the long term time trends very well for this dataset. It is of note that the peatland only ever increases, thus the trend it is learning is not very nuanced, but the good performance nonetheless is a positive indicator for the potential of learning even more complicated trends as with actual soil carbon data.

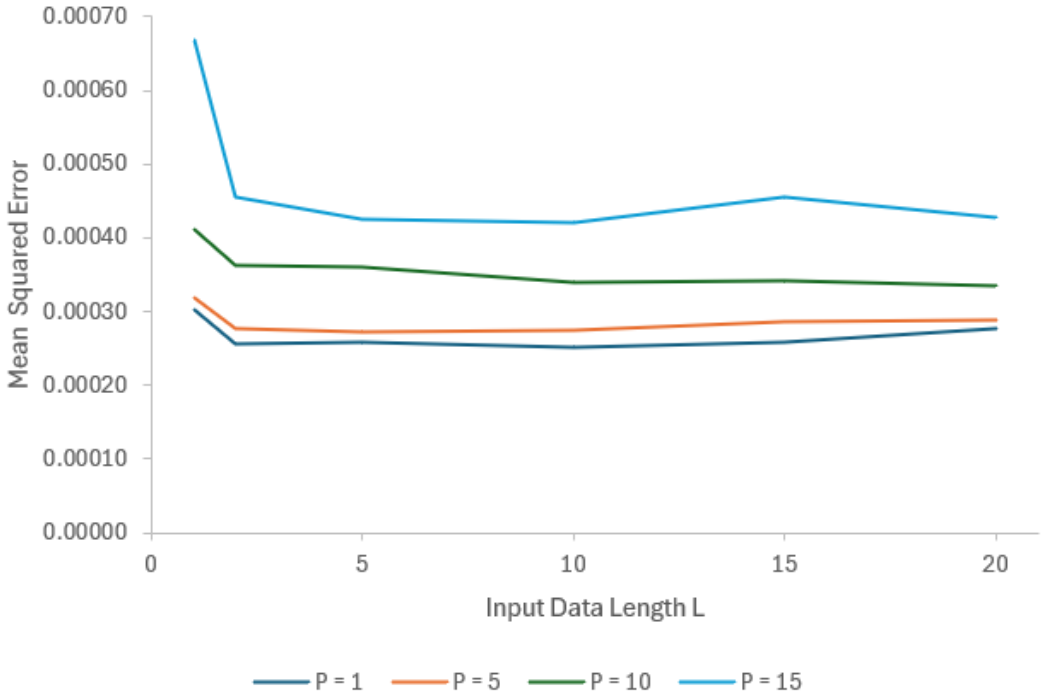
Figure 4-8 outlines how the MSE changes for each of the prediction horizons as

Figure 4-8: MSE over Prediction Horizon Length for Various Input Lengths



This figure presents the per pixel mean squared error as the prediction horizon P increases from 1 to 5 to 10 to 15 for varying input lengths L.

Figure 4-9: MSE over Input Data Length for Various Prediction Horizons



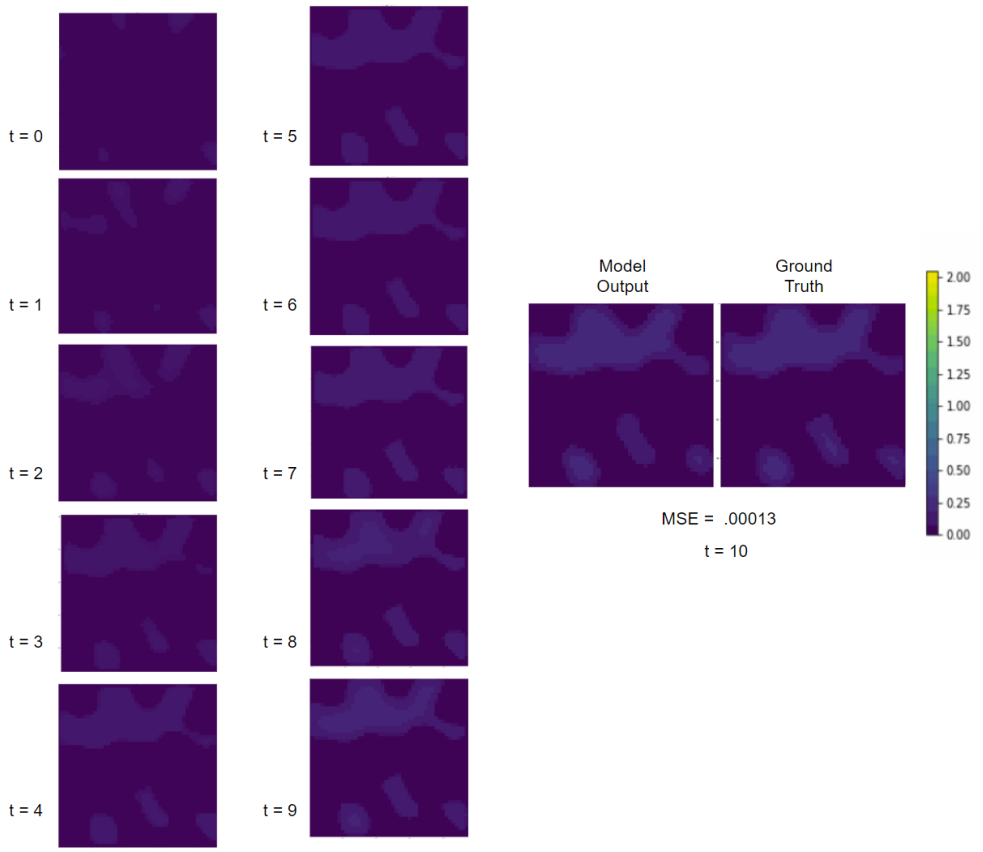
This figure presents the per pixel mean squared error as the input length L increases from 1 to 20 for varying prediction horizons P.

the input length increases. Unsurprisingly, as P increases so does the MSE for the models. The increase almost seems to happen linearly. The largest visible effect is when there is only one historical point of data. However for larger Ls the MSEs are clustered together more closely and the difference in MSEs could be due to variance and longer training times could potentially rectify this.

Figure 4-9 outlines how the MSE changes for each of the prediction horizons as the input length increases. As the above figure show, at $L = 2$ there is a huge dropoff in the mean squared error and after that the model doesn't make large improvements even as L keeps increasing. The model seems to be able to determine what stage of growth the peatland is at with only 2 historical data points and performs best when predicting 1 and 5 timesteps out in the future.

It is possible that as L and P change additional hyperparameter tuning (learning rate, batch size) would change the results outlined above; however due to limited computational resources an exhaustive tuning was not completed.

Figure 4-10: Best Performing Model Parameters ($L = 10, P = 1$) Test Set Sequence Results Earlier Timestep Example

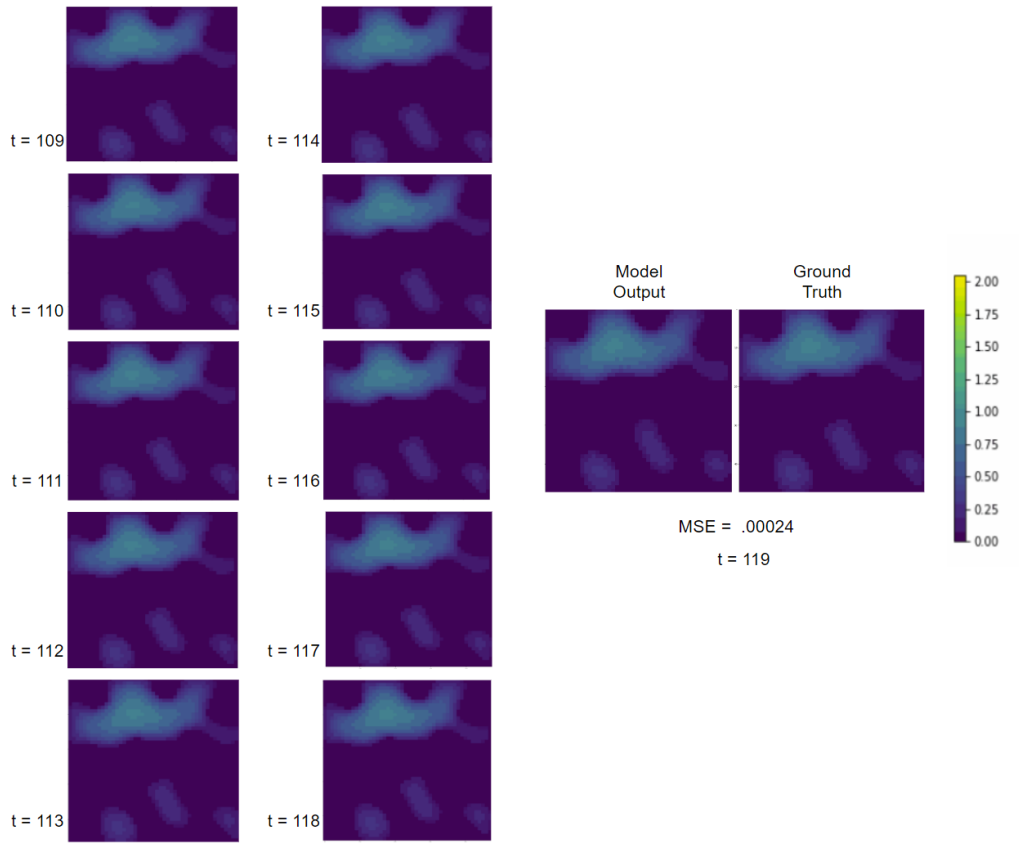


This figure presents the results of the best performing model parameters. The images in the two leftmost columns are the input sequence labeled by the timestep. The rightmost pair of images are the model output and the ground truth labeled accordingly. The MSE for this example is .00013.

A visualization of the best performing model parameters ($L = 10, P = 1$) on the later timesteps of the test set is presented in Figure 4-11. The model was untrained on all timesteps above 100. In these later timesteps, as seen in the figure, peat accumulates much slower and the model is able to understand how much peat should accumulate and adjust accordingly even without training on this temporal range. An example of the model performance on earlier timesteps where the peat accumulation is much more apparent from timestep to timestep is presented in Figure 4-10. The model is able to predict the increase in peat accordingly with good accuracy (MSE = .00013).

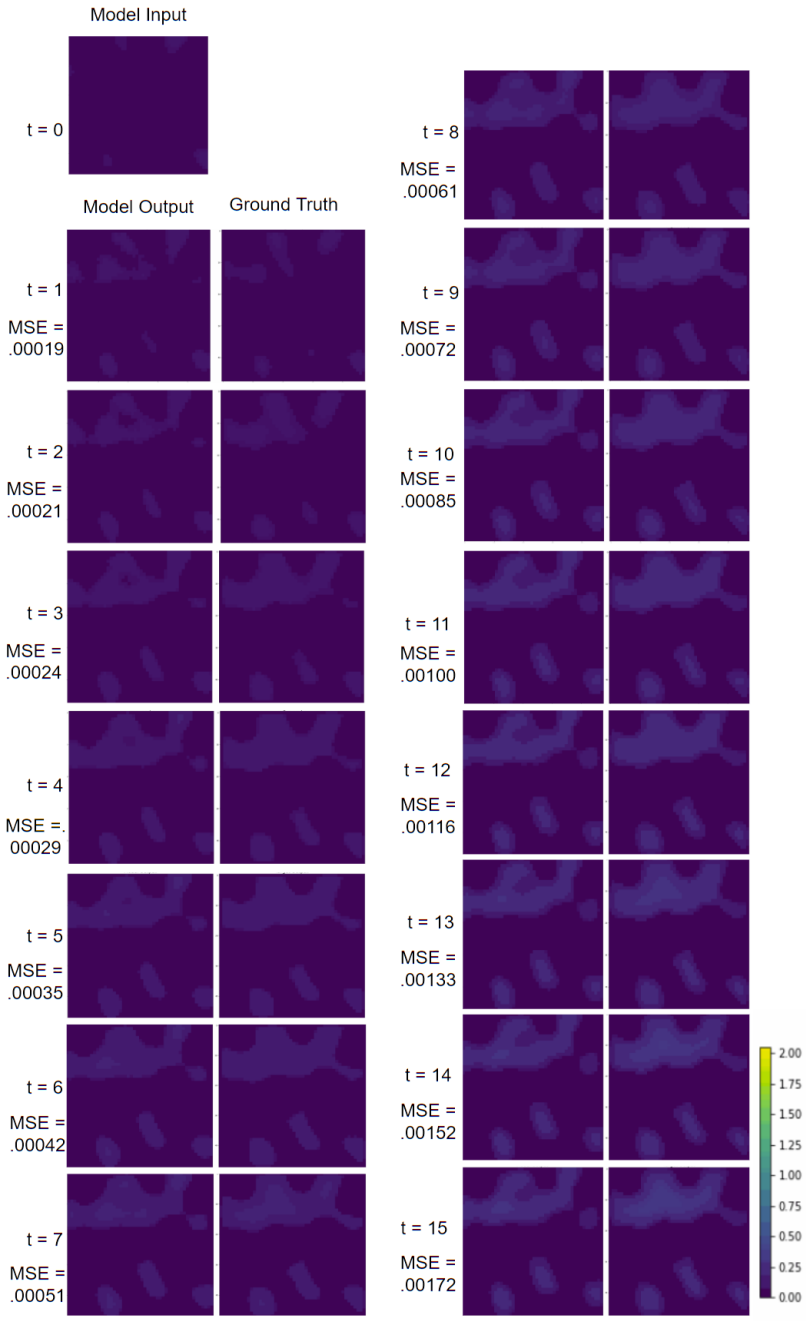
To contrast the performance of the easiest problem with the hardest: a visualiza-

Figure 4-11: Best Performing Model Parameters ($L = 10, P = 1$) Test Set Sequence Results Later Timestep Example



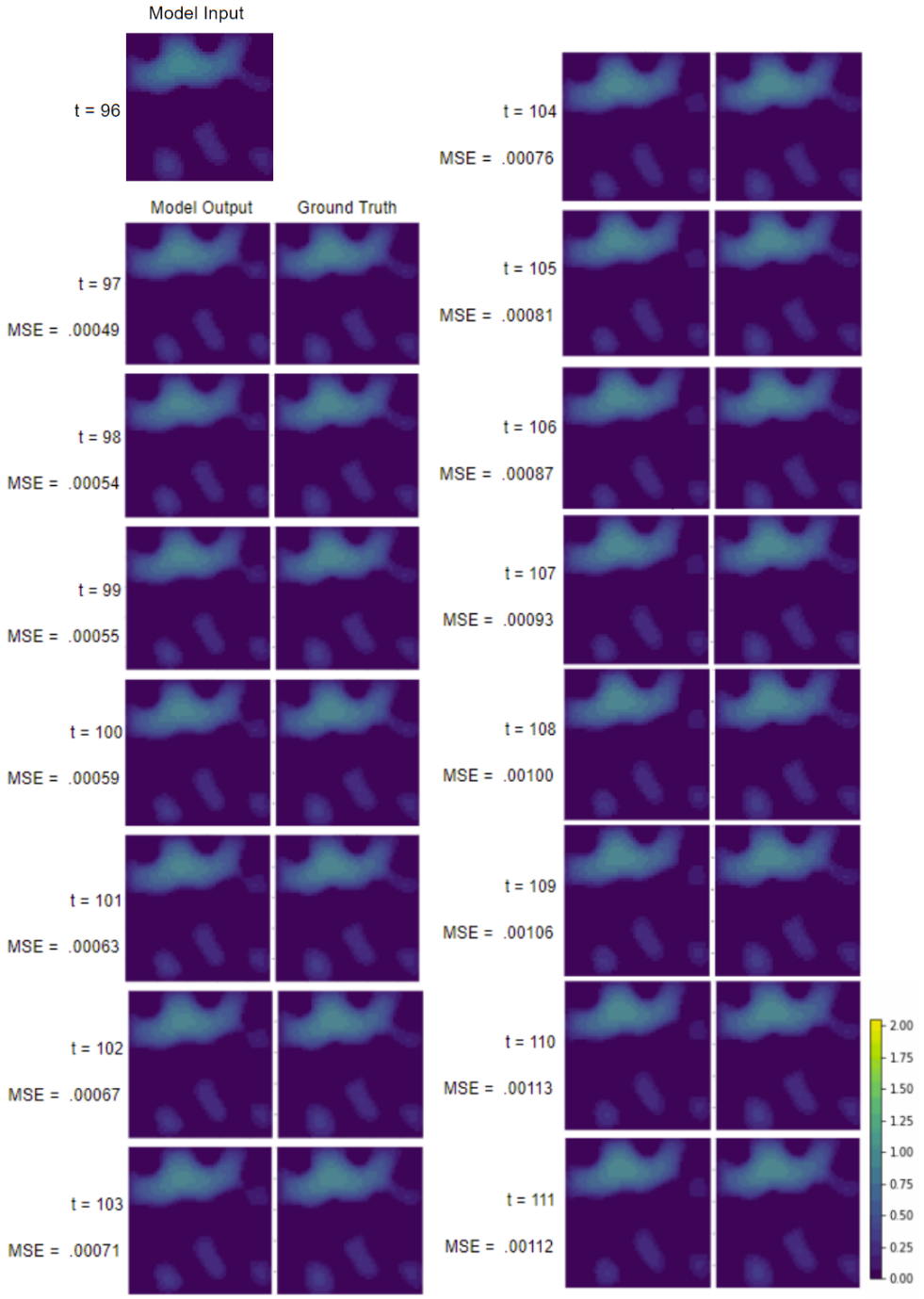
This figure presents the results of the best performing model parameters. The images in the two leftmost columns are the input sequence labeled by the timestep. The rightmost pair of images are the model output and the ground truth labeled accordingly. The MSE for this example is .00024.

Figure 4-12: Worst Performing Model Parameters ($L = 1, P = 15$) Test Set Sequence Results Earlier Timestep Example



This figure presents the results of the worst performing model parameters on the test set. The left column of the two image pairs represents the model output while the right represents the ground truth. The overall per pixel MSE for this example is .00074.

Figure 4-13: Worst Performing Model Parameters ($L = 1, P = 15$) Test Set Sequence Results Later Timestep Example



This figure presents the results of the worst performing model parameters on the test set on an untrained timestep range. The left column of the two image pairs represents the model output while the right represents the ground truth. The overall per pixel MSE for this example is .00079.

tion of the worst-performing parameters ($L = 1, P = 15$) on an earlier time range and on the untrained time range is presented in Figure 4-13 and in Figure 4-12 resspectively. It is of note that the error increases as the output timestep gets further away from the original, although it is difficult to pinpoint the difference visually as seen in Figure 4-13. Another interesting note is that at the earlier timesteps when the bogs are developing and a lot of mass is being added, the model is not able to generalize with only one training example and infer how the mass should be incorporated, leading to cavities in regions where the dome (i.e. the maximum of the carbon content) should be. However, the model performs much better once the bog shape has been established without strange shapes occurring (such as in later timesteps). While it has been pointed out that numerically, even the worst performing model's MSE is not significantly higher compared with the range of pixel values in the test set, these visual figures represent qualitatively evidence of this idea.

4.3 End-to-end Model

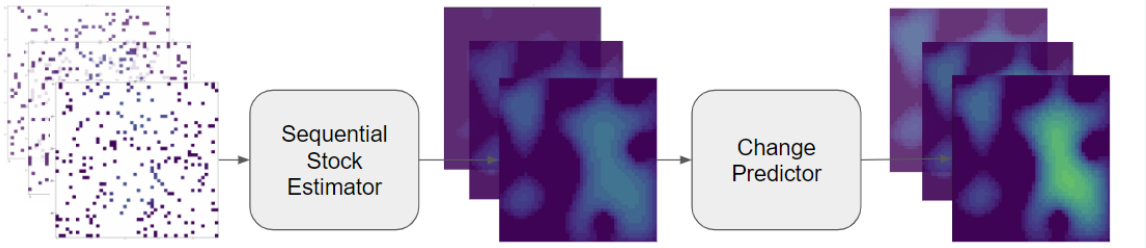
4.3.1 Model Architecture

The end-to-end model combines both the sequential stock estimator and the change predictor as visualized in Figures 4-2 and 4-7. However now rather than the change predictor taking in the ground truth it takes in the stock estimates from sequential stock estimator as shown in Figure 4-14.

4.3.2 Method

As models were already pretrained on various input lengths L and prediction horizons P in the previous two sections, no additional training of these models was conducted. In order to split the larger test set into pairs of sequences consisting of inputs (samples from $t-j$ until t) and targets (the farm carbon ground truth from $t-j$ until $t+\Delta t$) for evaluation, a method similar to the rolling window with no overlap algorithm was used (Algorithm 1). However, rather than the inputs being updated by the ground truth

Figure 4-14: Infilling Prediction End-to-end Model Architecture



This figure presents the overall model architecture combining the sequential estimator and the change predictor. The overall model takes in the sample snapshots as seen on the far left, creates infillings from the sequential estimator, feeds the infillings to the change predictor, and produces stock predictions on the far right.

farm carbon, they are updated by the samples taken at the corresponding ground truth timestep and targets is of length $L + P$ consisting of the ground truth for the infilling and the prediction steps.

4.3.3 Experiments

To establish a baseline for performance for the end-to-end model the pretrained networks for the sequential infilling and the change predictor were combined and evaluated end to end on the various sampling strategies and densities. The possible values for the length of sample inputs L and prediction horizon P were determined by the best performing parameters in experiments in the previous sections. For the sequential estimator the higher L the greater the information gain, especially for URS. For the change predictor, the values for P which achieved the lowest test MSE were 1 and 5 for values of $L \geq 2$. Thus a grid search was conducted over the following parameters $L \in [2, 5, 10, 15, 20]$ and $P \in [1, 5]$ for all sampling strategies and densities.

4.3.4 Results

The results of the pretrained end-to-end run reporting the overall mean squared error as well as comparison between predicting from the infills vs ground truth are presented in Table 4.4. The infilling results have already been reported in the previous section,

Table 4.4: End-to-end Model Test Set MSE Results

		P = 1			P = 5		
	URS Sampling Density	Overall MSE	Infilling Prediction MSE	Ground Truth Prediction MSE	Overall MSE	Infilling Prediction MSE	Ground Truth Prediction MSE
L = 2	1%	0.00821	0.00836	0.00025	0.00840	0.00869	0.00027
	5%	0.00042	0.00054		0.00053	0.00059	
	10%	0.00019	0.00032		0.00029	0.00036	
	25%	0.00013	0.00028		0.00022	0.00029	
L = 5	1%	0.00168	0.00186	0.00025	0.00179	0.00194	0.00027
	5%	0.00016	0.00033		0.00024	0.00035	
	10%	0.00011	0.00029		0.00019	0.00030	
	25%	0.00007	0.00026		0.00016	0.00029	
L = 10	1%	0.00066	0.00080	0.00024	0.00072	0.00087	0.00028
	5%	0.00010	0.00029		0.00017	0.00032	
	10%	0.00007	0.00027		0.00014	0.00031	
	25%	0.00005	0.00026		0.00012	0.00030	
L = 15	1%	0.00053	0.00078	0.00025	0.00061	0.00088	0.00029
	5%	0.00009	0.00030		0.00014	0.00034	
	10%	0.00006	0.00028		0.00011	0.00031	
	25%	0.00005	0.00027		0.00010	0.00030	
L = 20	1%	0.00047	0.00081	0.00027	0.00053	0.00087	0.00026
	5%	0.00009	0.00032		0.00012	0.00031	
	10%	0.00006	0.00030		0.00010	0.00029	
	25%	0.00004	0.00028		0.00008	0.00027	
	GBS Sampling Density						
L = 5	1%	0.00168	0.00186	0.00025	0.00179	0.00194	0.00027
	5%	0.00076	0.00092		0.00083	0.00094	
	10%	0.00010	0.00028		0.00018	0.00029	
	25%	0.00005	0.00026		0.00015	0.00028	

This table presents the mean squared error results of the end-to-end infilling and prediction model on the test set. P represents the prediction length and L represents the input length to infill. The overall MSE column is the sum of the infilling MSE and prediction MSE weighted by sequence lengths L and P respectively. Infilling prediction MSE is the mean squared error of just the predictor part of the end-to-end model which is fed the infilled images whereas the ground truth prediction MSE is the MSE of predictor part fed the ideal infilled images (the ground truth images).

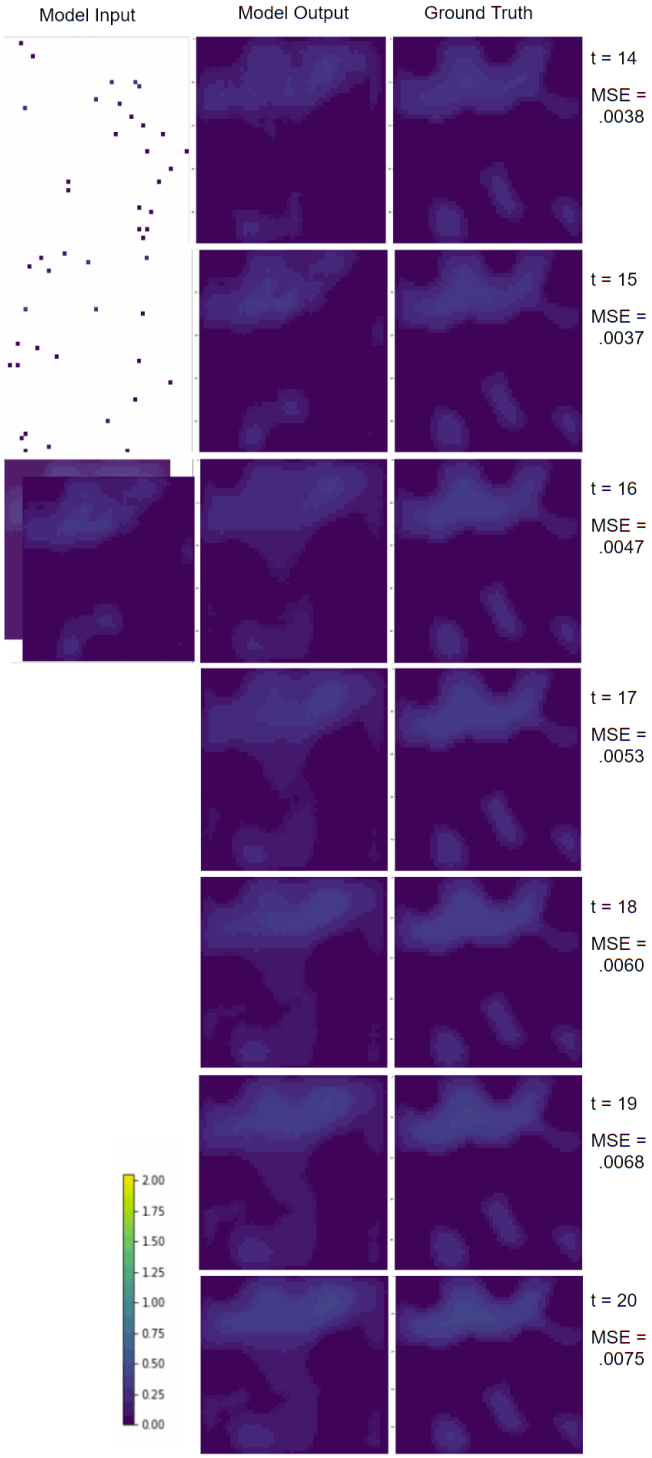
although the exact numbers differ (but are very similar) due to the infilling test set in this section being a subset of the test set from the earlier section. As the infilling performance for grid based sampling doesn't vary in choice of L and the prediction performance is very similar for $L \geq 2$, only the results for $L = 5$ (or the best performing parameter for infilling at the 25% and 10% sampling densities) for GBS is reported.

As seen in Table 4.4 the prediction from infilling error is much higher than the overall error as well as the error of predicting from the ground truth for all densities. The performance bottleneck comes as a result of the infilling inaccuracies snowballing downstream as the prediction length increases. Overall, while the end-to-end model shows some divergence from the ground truth at worst ($L = 2$, $P = 5$ at the 1% URS sampling) the mean squared error was .00869 or 5% of the test set mean pixel value. Over the $K \times K$ grid of the farm, this error compounds. As expected, the largest divergence from the ground truth prediction results from the parameters which also perform the worst at infilling as discussed in the Sequential Estimator section. From the Change Predictor results, the best performance with the given architecture and ideal ground truth infillings is a MSE of .00027. Thus the model MSE diverges by .00842 - an increase of around 31 times the ideal. A visualization of this snowball divergence is presented in Figure 4-15 where the model gets increasingly off based off of the initial infill estimates.

The lowest end-to-end MSE (.00004) resulted when $L = 20$, $P = 1$ for URS at the 25% sampling density. At that density with enough historical data the difference between the infilling prediction MSE and ground truth prediction MSE is negligible. To be precise, the MSE increases by 2%. One note is that the performance of the sequential estimator increases as L increases. As the overall MSE is a weighted function of the estimator and predictor, the higher L is compared to P the lower the overall MSE will be.

However, part of the problem in this space is achieving the best possible results while minimizing the number of samples taken. The methodology for doing this is explored more in depth in the next chapter. Given fixed sampling strategies and

Figure 4-15: End-to-end Model Worst Performing Parameters ($L = 2$, $P = 5$, 1% URS) Example



This figure presents the results of the end-to-end model worst performing parameters on the test set. The model takes in samples and produces infills which get fed in at the next timestep to produce predictions. The overall MSE for this example is .0054.

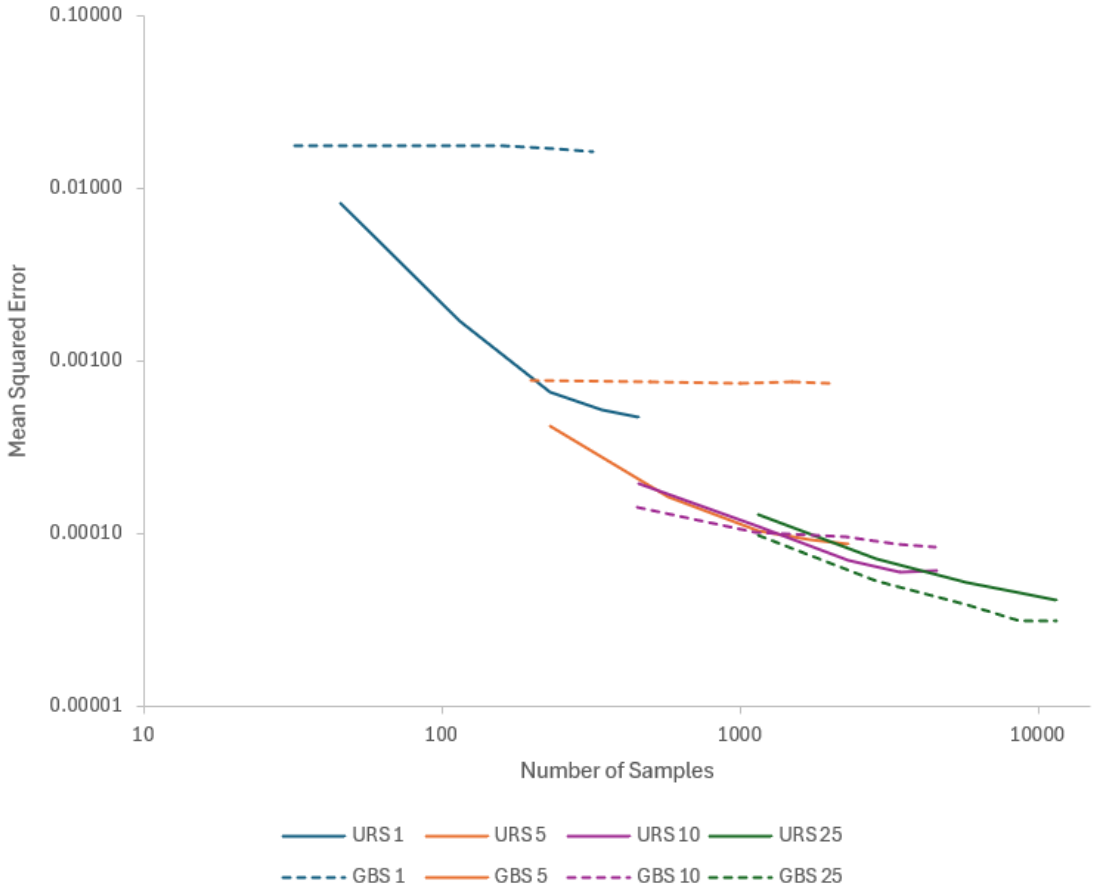
densities, Figure 4-16 presents the accuracy for the combinations of hyperparameters as a function of number of samples taken for $P = 1$ (as the MSE is across the board lower for $P = 1$.)

Unlike the Sequential Estimator, it is much clearer that sampling at a lower density for a longer period of time can lead to better performance end-to-end than sampling at a higher density for less time. For example, in Figure 4-16 around the 1000 sample mark we see that URS 10% and URS 5% are able to achieve better performance than URS 25%. There are definitely costs associated with waiting longer to take samples especially with more complicated realistic data, however the Sequential Estimator's performance on this dataset evens out after around 5 timesteps worth of data for all densities, which is just around the 1000 sample mark for both URS 5% and URS 10%. This provides incentive to conduct historical studies while keeping the total number of samples constant to increase performance while minimizing cost. Of course, there are additional costs to consider with taking samples across multiple timesteps. Similar to the Sequential Estimator, at a certain point there are diminishing returns to how increasing the number of points sampled increases knowledge. Ideally, we would know when to stop sampling as much due to the abundance of historical data that we have. While no URS models were able to outperform GBS at the 25% sampling density, other densities and sampling strategies were able to achieve a similar performance with fewer samples lending further intuition to how varying density and location can optimize sampling.

While the pretrained models were used above, it is possible improve the model performance by tuning the entire end-to-end model altogether. As the change predictor was trained on the ground truth, it is unlikely that the stock estimate results would provide additional information that could improve this part of the model. Thus to improve the model, additional tuning can be done on the earlier infilling layers.

It is possible that information from earlier layers in the network is lost as the sequence of samples passes across the network. For example, it might be possible to further improve model performance by passing in sequences of 3 channel images to the portion of the change predictor making the future predictions, including the

Figure 4-16: End-to-end Model MSE over Sample Number for Various Sampling Strategies, $P = 1$



This figure presents the log base 10 scale MSE on the test set of the end-to-end model for the combination of different sampling strategies and densities as the total number of samples increases (also on a log base 10 scale). URS stands for uniform random sampling, GBS stands for grid based sampling, and the number next to the strategy represents the sampling density (1%, 5%, 10%, or 25%). URS models are all plotted with solid lines and GBS models are plotted with dashed lines. Models trained on the same sampling density share the same color.

original samples, sample location mask, in addition to the infillings generated by the earlier layers of the model. This would allow the propagation of the original sample through the change predictor (similar to U-Net) instead of simply relying on the stock estimator performance as the current end-to-end model does.

4.4 Conclusions and Future Work

Due to limited computational resources, Gaussian process regression was not run on the sequences for sequential infilling, carbon forecasting, or the complete change predictor. It would be helpful to establish this higher baseline for a better understanding of learned model performance in the future.

Overall in this section we build a performant end-to-end model for carbon estimation and prediction. By contrasting sampling strategies and analyzing performance, the possibility arises of varying sampling strategy from timestep to timestep to increase estimating and predictive power conditioned on historical data. Chapter 5 builds upon this infrastructure by implementing a model that recommends these varying sampling points to minimize the end-to-end model error and number of samples to take.

Chapter 5

Sampling Optimizer

This chapter explores how to vary the number of samples one takes as well as their locations based on historical data to better increase predictive power and minimize costs compared to traditional methods where sampling methods are more static. This chapter proposes an end-to-end machine learning model to accomplish this.

5.1 Transformers

Transformers were first introduced in 2017 by Vaswani et al. [26]. By utilizing the concept of self-attention, the different parts of the model input are able to be weighed differently to produce an output. This paper designed transformers for sequential data (being particularly performant on language translation tasks) and soon after the concept of a vision transformer was introduced in 2020 by Dosovitskiy et al. [5] to perform self-attention on images by transforming an image into a sequence of image patches for direct input to a pure transformer rather than relying on convolutional layers. In general, there has not been much related work with regards to using machine learning for sampling optimization. However, the concept of self-attention translates smoothly to understanding which areas of the farm need additional attention and samples for information gain. The sampling optimizer leverages this vision transformer architecture to attend to areas of the sampling grid that would be beneficial to sample from given the areas that have already been sampled from.

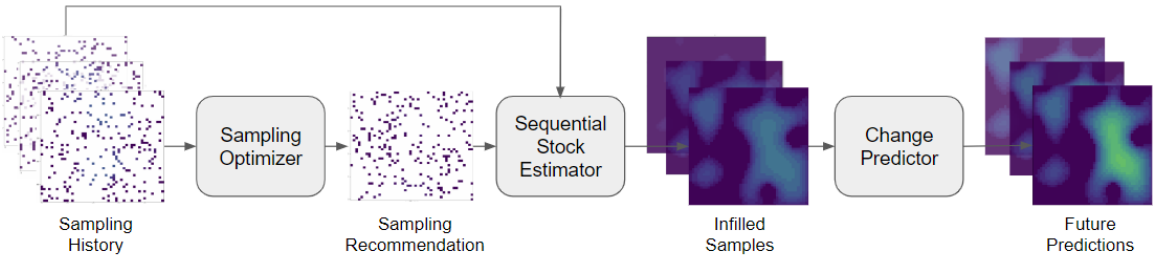
5.2 Model Architecture

This **sampling optimizer** $\mathcal{S}$ takes in some sampling history of the farm from j timesteps ago up to time $t-1$, $\mathcal{X}^i_{j,t} = \{X^{i,t-j}, X^{i,t-j+1}, ..., X^{i,t-1}\}$ and outputs an estimate $\hat{X}^{i,t} = \{\hat{X}^{i,t}_1, \hat{X}^{i,t}_2, ..., \hat{X}^{i,t}_n\}$ of the cost-optimal sample to apply in the cycle at current time t. $\hat{X}^{i,t}$ is a sequence of positions (x, y) of unspecified length n which represents the locations of the next samples to collect. In other words, a sampling history will be passed into the model to generate the optimal sampling points. The optimal sampling points will be determined through minimizing the prediction error (i.e., use the samples gathered at the recommended sampling points in the next timestep as input to the end-to-end change predictor and compare the results against the known ground truth) as well as cost of sampling (i.e., regularizing on the number of points outputted).

At a high level, the sampling optimizer consists of a module that takes in a sampling history and outputs a snapshot of the samples to take at the current timestep. Then given those samples and the sampling history, the end-to-end change predictor model is able to output the dense carbon map from time $t-j$ to $t+\Delta t$ consisting of the infilled samples and future time predictions. This is visualized in Figure 5-1. In the case of the simulated dataset, there is access to the ground truth sample values for training the sampling optimizer but in real time the user would need to take the sample locations outputted by the optimizer, sample at these locations, then plug the sampling history with the new sample values into the separate pretrained end-to-end change predictor. That is to say the model behavior is different when training and evaluating on known ground truths vs when optimizing for locations in the absence of ground truth. When training and evaluating as there is access to the ground truth, the model is able to "sample" at the recommended locations and generate the rest of the output for the end-to-end change predictor, as well as back propagate the average mean squared error across the entire network.

The sequential estimator and change predictor architectures have already been described in the previous chapter. The more detailed architecture for the sampling

Figure 5-1: Sampling Optimizer End-to-end Model Architecture



This figure presents the higher level model architecture of the end-to-end sampling optimizer. The overall model takes in the sampling history on the far left to output a recommended sampling. This is concatenated with the model input to become input for the rest of the model.

optimizer module is as follows. As, it is important to identify parts of the $K \times K$ grid that require more "attention", a vision transformer as described above was used to generate a mask of size $K \times K$ of where samples at timestep t should be taken. The model takes in a sequence of samples which are transformed into a larger sequence of patches by using a convolutional layer. The model architecture consists of a convolutional layer to convert each patch to an embedding. This embedding along with the positional encoding of each patch is passed into a transformer encoder (the encoder portion of the full transformer proposed by Vaswani et al. [26].) The positional encoding of each patch is a learned parameter. The output of this encoder is then aggregated across the sampling history timesteps by taking the mean. The aggregated embedding is passed through a final linear layer for reshaping with a Sigmoid activation function to produce the final mask of 0s and 1s.

5.3 Method

From the previous Chapter results, sampling at the 10% or 25% density provides a lot of information such that additional points don't result in that huge of an increase in performance. Additionally for $L = j$ timesteps, when $L \geq 5$, performance increases much slower. Thus, the model is trained on sampling histories of length $L = 1$ at the 5% density for an initial proof of concept.

In order to split the data into sequences for training and evaluation, the first 90% of farms were used from the larger training validation set for training and the remaining 10% of farms were used for validation. The data was then separated into sample sequences of size $L + 1$ and ground truth target sequences of $L + 2$ for the infilling and predictions using an overlapping window method similar to as described in Algorithm 2. In addition, data augmentations were applied to the training dataset, consisting of random horizontal and vertical flips to increase spatial variety.

Due to the image being small there was room for more computational resources. Thus each image was split into patches of size 4 in the patch embedding resulting in a sequence of 144 patches per image. The transformer encoder portion of the model consists of 6 encoding layers with an embedding dimension of 128, 8 attention heads, and a final MLP layer with dimension 256.

As an end-to-end model for infilling and prediction has already been trained from the previous Chapter, the weights of this end-to-end change predictor were loaded in the sampling optimizer to speed up training time. Each model was trained for 20 epochs and the model with the lowest validation loss was used to evaluate performance on the test set. All models were trained with a batch size of 32, the Adam optimizer, and a learning rate of .0001.

The loss for this sampler optimizer is the combination of the losses of the sequential estimator, the change predictor, and also a regularization term on the number of samples the optimizer outputs. To encourage the optimizer to output fewer samples but not completely go to 0, a ReLU function was applied over the difference between the number recommended points and the average number of points per timestep in the sampling history fed into the optimizer. The cost associated to each timestep t for individual farms is thus:

$$\mathcal{L}_{\mathcal{S}} = \alpha\mathcal{L}_{\mathcal{E}} + \beta\mathcal{L}_{\mathcal{P}} + \lambda\left(len(\hat{X}) - \frac{len(\mathcal{X})}{L}\right) \tag{5.1}$$

The values of α, β, γ are tunable and depend on details such as laboratory costs, and carbon credit prices. The model was trained with preliminary values of $\alpha = \beta = 1$

and $\lambda = .001$.

Altogether, the user is able to input samples (either collected through their methods our recommended by our system) to generate the expected soil carbon map for current and future timesteps. Suggested sampling points will also be outputted which when sampled at will result in an efficient way to accurately account for the carbon stock on the farm.

5.4 Experiments

The choice of starting sampling strategy can also be important. To understand if/how the spatial location of the initial historical samples impacts the model above was trained on starting samples from both uniform random sampling and grid based sampling.

The recommended sampling points of the model are plugged back into the pre-trained end-to-end infiller and predictor to generate the MSE. The end-to-end MSE is then compared between the new recommended sampling sampling strategy and the previous URS/GBS from the previous chapter where $L = 2$ (combining the historical input and sampling optimizer output) and $P = 1$.

5.5 Results

How the sampling optimizer performs when plugging in the recommended sampling points concatenated with the history into the end-to-end model with preloaded weights for $P = 1$ can be found in Table 5.1.

Overall we see that for URS the sampling optimizer was effective at reducing the mean samples per sequence; however, the test MSE was much higher for the same sequence length. An interesting note is that although the model's recommended points were unable to outperform the 5% URS model without looking at the number of samples, the end-to-end model trained on 1% URS data has a test MSE of .00168 at 115 samples ($L = 5$) and a test MSE of .00066 at 230 samples ($L = 10$) per Table

Table 5.1: Sampling Optimizer Model Test Set MSE Results and Comparisons

Sampling Strategy 5% Density	Test MSE	Mean Samples Per Sequence
URS Opt	0.00097	160.66280
URS No Opt L = 2 H = 0	0.00042	230.00000
GBS Opt L = 1 H = 1	0.00040	432.18340
GBS No Opt L = 2 H = 0	0.00078	200.00000
GBS No Opt L = 5	0.00076	500.00000

This table presents the results of the end-to-end model run on the various sampling strategies. The sampling optimizer here is trained on the 5% density level as are the rest of the strategies. URS stands for uniform random sampling and GBS stands for grid based sampling. Opt means that the sampling optimizer was used to generate samples for the model and vice versa for No Opt. L denotes the sampling snapshots inputted to the model generated through either URS or GBS and H denotes the sampling snapshots generated by the sampling optimizer given L. The Test MSE column denotes the end-to-end model performance with the inputted or generated sampling sequence on the test set when P = 1. The final column denotes the mean number of samples taken per sequence.

4.4. Thus, the sampling optimizer output is able to place us somewhere between these results however looking at Figure 4-16 the trend line for the 1% is still much higher than the 5% URS line. Ultimately, the optimizer did not produce better results for the URS.

While we were unable to see improvements for URS at this level, there were improvements for GBS. As discussed in the previous chapter, regardless of increasing the number of snapshots for GBS the test error did not decrease as samples were drawn from the same location every time. While the optimizer did increase the number of samples taken, the MSE achieved was lower than the performance of GBS with 200 samples per sequence and GBS with 500 samples per sequence.Additionally, at the 10% density (450 samples) a model trained on GBS is able to achieve a test loss of .000142 as pictured in Figure 4-16, which is a much better result given the comparable number of samples. Thus, even though there is an improvement when comparing against the same density with longer histories resulting in more samples, by comparing against a higher density with around the same number of samples we

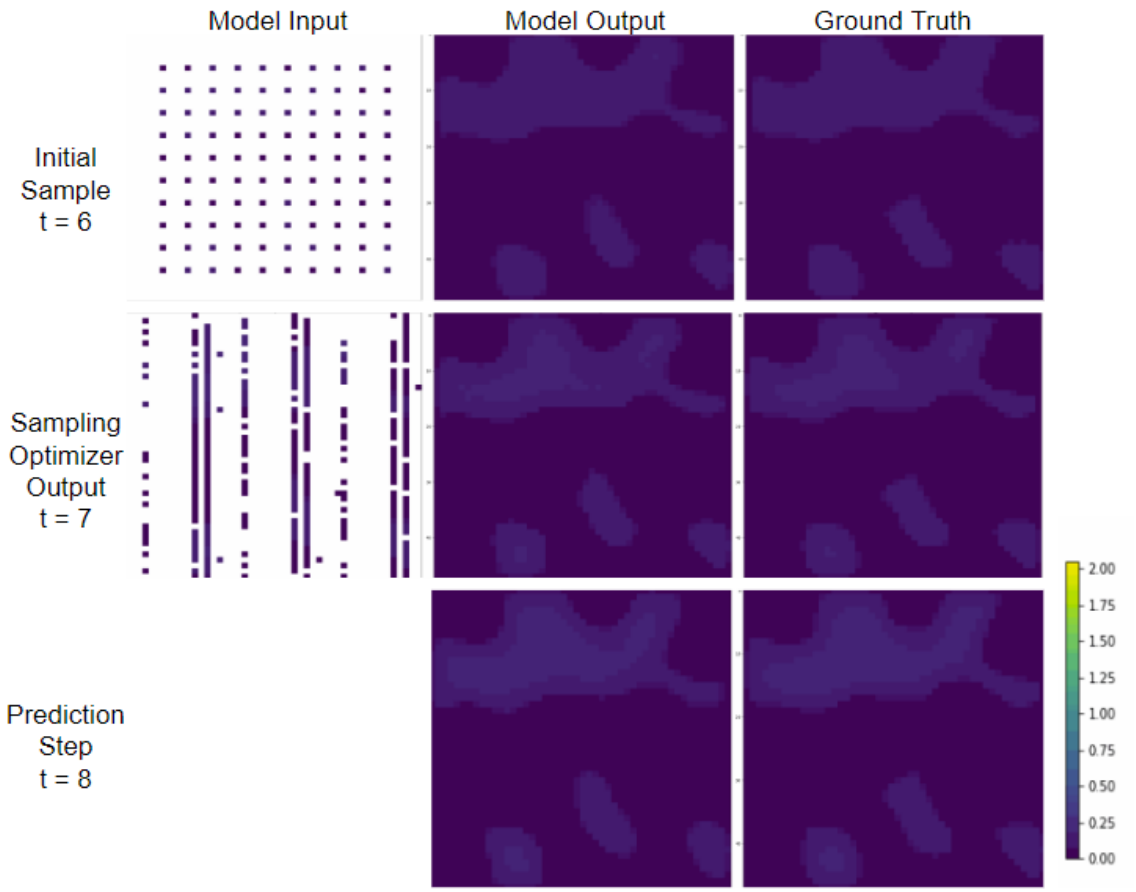
see this optimization isn't as efficient as it could be. An additional avenue of work would be to compare the sampling optimizer to a combination of established strategies ie: GBS and URS together.

For context on what the sampling optimizer is outputting, Figures 5-2 and 5-3 present the output of the sampling optimizer alongside the end-to-end model for an example sequence. As seen in Figure 5-2, the sampling optimizer chooses points that were not initially sampled in the original sample to sample next. This shows some promise. The shape of the selected samples is quite interesting as it closely resembles grid based sampling. However, while the performance improves as a result of using the sampling optimizer output, the fact that the sampling optimizer is sampling straight vertical lines across the farm is not quite as efficient. For URS, the results are a little unexpected. Theoretically one would expect the clear holes in the initial sample to be what the model focuses on. There is some variance in the URS sample recommendations compared to GBS but there is still a preference for the model to choose sample along the same vertical column.

5.6 Conclusions and Future Work

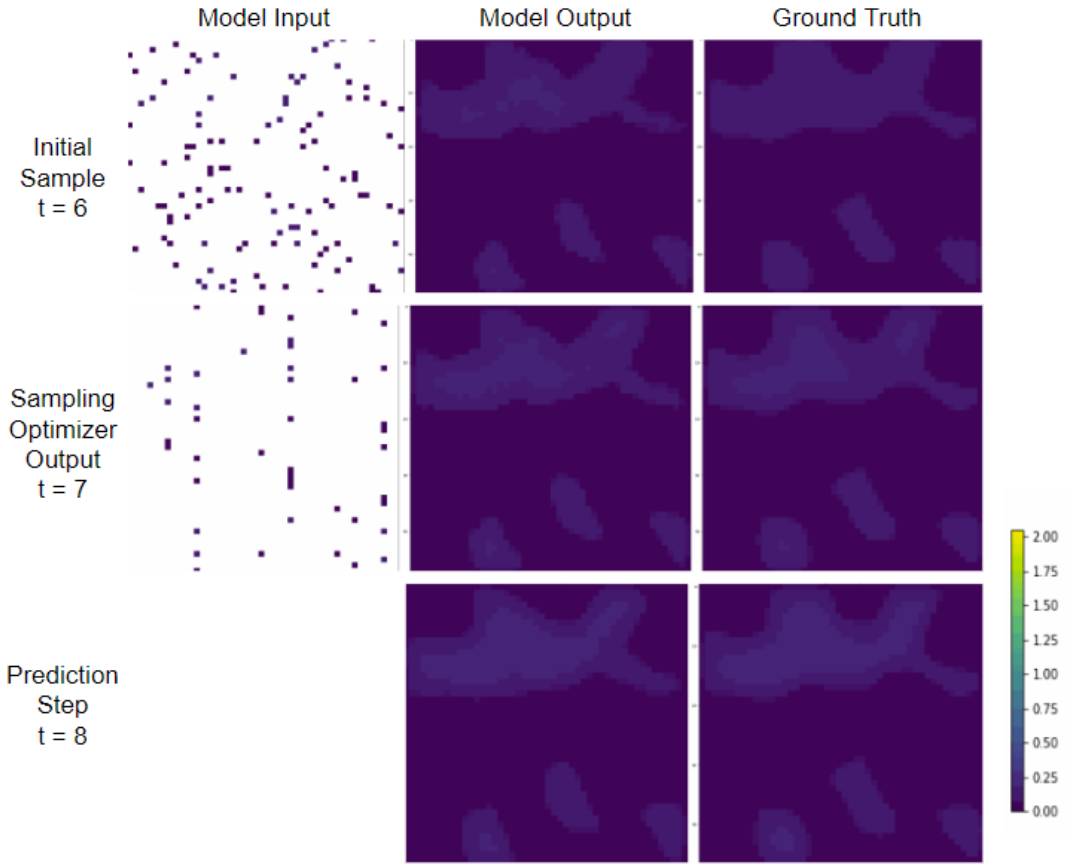
Overall, while the ViT did demonstrate how lowering the number of samples taken by varying their location can result in more efficient sampling there are a couple of areas of improvement for the model. First, the model tends to output around the same distribution of points for inputs that are the same. Additionally, the model tends to output sampling recommendations in the same column as each other even when it results without much of an improvement in performance. Modifications could be made to the attention mechanism to ensure that the attention scores vary sufficiently from point to point in the graph. Additional fine tuning of the model hyperparameters and potentially longer training could help improve model performance. Additionally, an experiment that would be interesting to run is how the sampling optimizer changes its recommendations as it is fed a history of its own recommendations. A patch size of 4 was chosen to still maintain some fine grained control but the hope was to classify

Figure 5-2: 5% GBS Sampling Optimizer and End-to-end Model Example



This figure presents the results of the sampling optimizer trained on 5% GBS data in the first column second row. This sample along with the initial sample is inputted into the fine-tuned end-to-end model to produce the model output in the second column.

Figure 5-3: 5% URS Sampling Optimizer and End-to-end Model Example



This figure presents the results of the sampling optimizer trained on 5% URS data in the first column second row. This sample along with the initial sample is inputted into the fine-tuned end-to-end model to produce the model output in the second column.

each patch as to be sampled or not in the next timestep, this could become more fine grained by classifying each pixel by decreasing the patch size to 1. This would be extremely computationally expensive though. Finally, considering other architectures could improve this result as well.

There is still a lot of exciting work remaining for this portion of the project and larger improvements in the model will lend further credibility to this approach and less of a reliance on traditional sampling methods in the future.

Chapter 6

Conclusions

The problem of maximizing information gain from chosen samples is not new and remains very important across a variety of fields. With regards to soil carbon sequestration where the costs of each and every sample is so high the ability to maximize information gain from every sample is imperative especially as the problem of carbon sequestration to prevent global warming becomes more and more pressing.

One of the largest barriers to this is the lack of data in the field. In order to counter this and develop a proof of concept system for estimation, prediction, and optimization we simulated our own farm scale carbon dataset for various farms across time.

This thesis has demonstrated across Chapter 3 and 4 that even under a sparse sampling regime it is possible to accurately reconstruct an esimation of a soil carbon map given adequate data. Using 2D and 3D CNNs, we were able to outperform the traditional Gaussian process regression that is performed in this field on our dataset with only one timestep of historical data. Additionally, we demonstrate the importance of gathering historical data to further improve carbon estimation. We demonstrate that without historical data, grid based sampling outperforms uniform random sampling at the same densities however with historical data uniform random sampling was able to surpass grid based sampling performance. By gathering even two timesteps of data we were able to demonstrate a huge decrease in the average per pixel mean squared error and at the higher end the mean squared error was 0%

of the average pixel value.

However, estimation is not the only pressing problem in this space. Oftentimes carbon contents from previous timesteps dictate the behavior of future timesteps. In this thesis we also demonstrate the capability of encoder decoder LSTMs to accurately predict the evolution of soil carbon in the future for our dataset given the ground truth. This prediction module was combined with the previous estimation module to create an end-to-end model that given a sequence of samples at various densities was able to infill these samples to create dense carbon maps and from these infillings generate predictions without too much divergence from the ground truth with sufficient sampling history.

The final problem this thesis attempts to solve is the problem of how to make more informed sampling decisions. Traditionally samples are taken after stratifying farmlands and applying grid based sampling across these stratum. This thesis proposes a novel way of choosing where to sample by developing an attention based model to recommend sampling points while minimizing the number of points needed to sample. While improvements were not found for uniform random sampling, some improvements were found for grid based sampling with regards to both the number of samples to be gathered as well as the average mean squared error. Ultimately, this model could benefit from additional hyperparameter tuning as well as potentially an architecture overhaul for better performance.

Overall, while the dataset used was simulated, the results are promising. With actual ground level samples as well as additional covariates, such as elevation, precipitation, or soil type, across time, the results will be more applicable. However, this thesis provides a foundation for integrating this additional data that does not yet exist to better understand the evolution of soil carbon across time under sparse sampling regimes as well as a methodology for deciding where to sample given previous information that traditional methods do not.

Bibliography

[1] Adrian Chappel, Jeff Baldock, and Raphael Viscarra Rossel. Sampling soil organic carbon to detect change over time. *CSIRO, Black Mountain, ACT*, 2013.

[2] Alexander R. Cobb, Alison M. Hoyt, Laure Gandois, Jangarun Eri, René Dommain, Kamariah Abu Salim, Fuu Ming Kai, Nur Salihah Haji Su'ut, and Charles F. Harvey. How temporal patterns in rainfall determine the geomorphology and carbon fluxes of tropical peatlands. *Proceedings of the National Academy of Sciences*, 114(26):E5187–E5196, 2017.

[3] William Gemmell Cochran. *Sampling techniques*. john wiley & sons, 1977.

[4] J.J. de Gruijter, A.B. McBratney, B. Minasny, I. Wheeler, B.P. Malone, and U. Stockmann. Farm-scale soil carbon auditing. *Geoderma*, 265:120–130, 2016.

[5] 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. *CoRR*, abs/2010.11929, 2020.

[6] Katerina Georgiou, Robert B Jackson, Olga Vindušková, Rose Z Abramoff, Anders Ahlström, Wenting Feng, Jennifer W Harden, Adam F A Pellegrini, H Wayne Polley, Jennifer L Soong, William J Riley, and Margaret S Torn. Global stocks and capacity of mineral-associated soil organic carbon. *Nat. Commun.*, 13(1):3797, July 2022.

[7] Anton Grafström and Lina Schelin. How to select representative samples. *Scandinavian Journal of Statistics*, 41(2):277–290, 2014.

[8] Jochen Görtler, Rebecca Kehlbeck, and Oliver Deussen. A visual exploration of gaussian processes. *Distill*, 2019. https://distill.pub/2019/visual-exploration-gaussian-processes.

[9] Nafiseh Kakhani, Moien Rangzan, Ali Jamali, Sara Attarchi, Seyed Kazem Alavipanah, and Thomas Scholten. Soilnet: An attention-based spatio-temporal deep learning framework for soil organic carbon prediction with digital soil mapping in europe, 2023.

[10] R. Lal. Soils and sustainable agriculture. a review. *Agronomy for Sustainable Development*, 28(1):57–64, March 2008.

[11] RM Lark. Soil–landform relationships at within-field scales: an investigation using continuous classification. *Geoderma*, 92(3-4):141–165, 1999.

[12] Guilin Liu, Fitsum A. Reda, Kevin J. Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions, 2018.

[13] A. B. McBratney and J. J. de Gruijter. A continuum approach to soil classification by modified fuzzy k-means with extragrades. *Journal of Soil Science*, 43(1):159–175, 1992.

[14] Neil McKenzie, P Ryan, P Fogarty, and J Wood. Sampling, measurement and analytical protocols for carbon estimation in soil, litter and coarse woody debris. *National Carbon Accounting System Technical Report No. 14*, 2000.

[15] Xiangtian Meng, Yilin Bao, Yiang Wang, Xinle Zhang, and Huanjun Liu. An advanced soil organic carbon content prediction model via fused temporal-spatial-spectral (tss) information based on machine learning and deep learning algorithms. *Remote Sensing of Environment*, 280:113166, 2022.

[16] A. Orgiazzi, C. Ballabio, P. Panagos, A. Jones, and O. Fernández-Ugalde. Lucas soil, the largest expandable soil dataset for europe: A review. *European Journal of Soil Science*, 69(1):140–153, Nov 2017.

[17] Zisis Petrou and Yingli Tian. Prediction of sea ice motion with convolutional long short-term memory networks. *IEEE Transactions on Geoscience and Remote Sensing*, PP:1–12, 04 2019.

[18] Eric Potash, Kaiyu Guan, Andrew J. Margenot, DoKyoung Lee, Arvid Boe, Michael Douglass, Emily Heaton, Chunhwa Jang, Virginia Jin, Nan Li, Rob Mitchell, Nictor Namoi, Marty Schmer, Sheng Wang, and Colleen Zumpf. Multi-site evaluation of stratified and balanced sampling of soil organic carbon stocks in agricultural fields. *Geoderma*, 438:116587, 2023.

[19] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[20] Kelly Ribeiro, Felipe S Pacheco, José W Ferreira, Eráclito R de Sousa-Neto, Adam Hastie, Guenther C Krieger Filho, Plínio C Alvalá, Maria C Forti, and Jean P Ometto. Tropical peatlands and their contribution to the global carbon cycle and climate change. *Glob. Chang. Biol.*, 27(3):489–505, February 2021.

[21] Copernicus Climate Change Service. Era5 hourly data on single levels from 1940 to present, 2018.

[22] Gregorio C. Simbahan, Achim Dobermann, Pierre Goovaerts, Jianli Ping, and Michelle L. Haddix. Fine-resolution mapping of soil organic carbon based on multivariate secondary data. *Geoderma*, 132(3):471–489, 2006.

[23] Pete Smith, Jean-Francois Soussana, Denis Angers, Louis Schipper, Claire Chenu, Daniel P. Rasse, Niels H. Batjes, Fenny van Egmond, Stephen McNeill, Matthias Kuhnert, Cristina Arias-Navarro, Jorgen E. Olesen, Ngonidzashe Chirinda, Dario Fornara, Eva Wollenberg, Jorge Álvaro Fuentes, Alberto Sanz-Cobena, and Katja Klumpp. How to measure, report and verify soil carbon change to realize the potential of soil carbon sequestration for atmospheric greenhouse gas removal. *Global Change Biology*, 26(1):219–241, 2020.

[24] Soil Survey Staff. Rapid carbon assessment (raca) project. *United States Department of Agriculture, Natural Resources Conservation Service*, Jun 2013.

[25] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks, 2014.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[27] Jonathan Verschuuren. Achieving agricultural greenhouse gas emission reductions in the eu post-2030: What options do we have? *Review of European, Comparative amp; International Environmental Law*, 31(2):246–257, June 2022.

[28] Alexandre M.J.-C. Wadoux, Benjamin P. Marchant, and Richard M. Lark. Efficient sampling for geostatistical surveys. *European Journal of Soil Science*, 70(5):975–989, 2019.

[29] Marissa Wiseman, Michael Patrick Cope, Cristine L.S. Morgan, and Jason Paul Ackerson. Soil sampling methods and systems.

[30] Boqiang Xie, Jianli Ding, Xiangyu Ge, Xiaohang Li, Lijing Han, and Zheng Wang. Estimation of soil organic carbon content in the ebinur lake wetland, xinjiang, china, based on multisource remote sensing data and ensemble learning algorithms. *Sensors*, 22(7), 2022.

[31] Chao Yang, Xin Lu, Zhe Lin, Eli Shechtman, Oliver Wang, and Hao Li. High-resolution image inpainting using multi-scale neural patch synthesis, 2017.

[32] Lin Yang, Yanyan Cai, Lei Zhang, Mao Guo, Anqi Li, and Chenghu Zhou. A deep learning method to predict soil organic carbon content at a regional scale using satellite-based phenology variables. *International Journal of Applied Earth Observation and Geoinformation*, 102:102428, 2021.

[33] Junhai Zhai, Sufang Zhang, Junfen Chen, and Qiang He. Autoencoder and its various variants. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 415–419, 2018.