remote sensing

MDPI

Article

Crop Yield Prediction Using Multitemporal UAV Data and Spatio-Temporal Deep Learning Models

Petteri Nevavuori [1,*], Nathaniel Narra [2], Petri Linna [2] and Tarmo Lipping [2]

[1] Mtech Digital Solutions Oy, 01301 Vantaa, Finland
[2] Faculty of Information Technology and Communication Sciences, Tampere University, 33014 Tampere, Finland; nathaniel.narra@tuni.fi (N.N.); petri.linna@tuni.fi (P.L.); tarmo.lipping@tuni.fi (T.L.)
* Correspondence: petteri.nevavuori@mtech.fi

Received: 4 November 2020; Accepted: 4 December 2020; Published: 7 December 2020

Abstract: Unmanned aerial vehicle (UAV) based remote sensing is gaining momentum worldwide in a variety of agricultural and environmental monitoring and modelling applications. At the same time, the increasing availability of yield monitoring devices in harvesters enables input-target mapping of in-season RGB and crop yield data in a resolution otherwise unattainable by openly availabe satellite sensor systems. Using time series UAV RGB and weather data collected from nine crop fields in Pori, Finland, we evaluated the feasibility of spatio-temporal deep learning architectures in crop yield time series modelling and prediction with RGB time series data. Using Convolutional Neural Networks (CNN) and Long-Short Term Memory (LSTM) networks as spatial and temporal base architectures, we developed and trained CNN-LSTM, convolutional LSTM and 3D-CNN architectures with full 15 week image frame sequences from the whole growing season of 2018. The best performing architecture, the 3D-CNN, was then evaluated with several shorter frame sequence configurations from the beginning of the season. With 3D-CNN, we were able to achieve 218.9 kg/ha mean absolute error (MAE) and 5.51% mean absolute percentage error (MAPE) performance with full length sequences. The best shorter length sequence performance with the same model was 292.8 kg/ha MAE and 7.17% MAPE with four weekly frames from the beginning of the season.

Keywords: crop yield prediction; UAV; spatio-temporal modelling; time series; deep learning; cnn-lstm; convolutional lstm; 3d-cnn

1. Introduction

The abundance of modern sensor and communication technology already present in production facilities and similar highly connected environments has also seeped into the realm of agriculture. Various globally, nationally and locally available data generating remote sensing systems are in place, providing relevant data for optimizing several agricultural outputs. On the global and national scale, satellite systems (Sentinel and Landsat missions, for example) provide temporally relevant spatial data about visible land surfaces. Nationally, there are various instruments in place to both track and predict climatological variables. Data for fields and relevant other entities is also gathered on a per-field basis by agricultural expert institutions. While satellite data is meaningful when monitoring large fields, smaller fields common to European countries, as an example, require higher resolution data. Human-operated unmanned aerial vehicles (UAV) play a key role in high resolution remote sensing in fields, that otherwise would wholly be covered by just tens or, at most, a couple hundreds of open-access satellite spatial data resolution pixels (10 $\times$ 10 m/px for Sentinel-2, for example). Also, utilizing modern sensors and global navigation satellite system (GNSS) tracking with agricultural machinery further adds detail to the pool of generated data. Modern data-based modeling techniques also benefit from increased resolution of spatial data, as they are able to better learn the relevant

features in performing a given task, e.g., intra-field yield prediction. Feeding this data to automated processing and decision making pipelines is a vital part of Smart Farming enabling Decision Support Systems [1].

In [2] we performed crop yield estimation with point-in-time spatial data, point-in-time estimation being contrary to time series regression. In this study we examined the effect of time, as an additional feature, on intra-field yield prediction. Especially, we focused on the capabilities of deep learning time series models utilizing UAV remote sensing time series data as their inputs. Firstly, we wanted to see if we could surpass the performance of the point-in-time model [2] by using spatio-temporal deep learning model architectures. Secondly, we wanted to see which spatio-temporal architecture would perform better in the same task. Lastly, we perform comparative evaluation of different sequence configurations to perform actionable crop yield predictions with data collected at the beginning of the growing season.

We utilize the properties of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks to perform spatio-temporal modelling. The CNN is briefly discussed in Section 2.3.1 and the LSTM in Section 2.3.2. The model architectures that we implemented are the following:

CNN-LSTM. CNN and LSTM networks can be utilized as separate but sequentially connected feature extractors, where the CNN first ingests spatial data and then provides extracted spatial feature data to the LSTM [3]. These models are hereafter referred to as CNN-LSTM are discussed in Section 2.3.3.

ConvLSTM. Convolutional learning properties can also be utilized differently. Models that utilize convolutional layers embedded into the LSTM architecture in a manner that eliminates the necessity to use spatial feature extraction prior to feeding the data to a sequential model are hereafter referred to as ConvLSTMs [4] and discussed in Section 2.3.4.

3D-CNN. A fully convolutional architecture can also be used to model sequential data. It is done by applying the convolution in the direction of time (or depth) in addition to width and height dimensions of spatial data [5]. Fully convolutional models utilizing the third dimensions for convolution are hereafter referred to as 3D-CNNs and discussed in Section 2.3.5.

1.1. Related Work

Regarding data similar or related to our study, recent crop-related studies utilize satellite-based data at scales larger than single fields. Ref. [6] performed county-scale soybean yield prediction with a CNN-LSTM architecture in parts of US. In addition to US national weather and yield data, they used time series satellite data from the MODIS satellite system. The data resolution was from 500×500 m/px to 1×1 km/px. Ref. [7] performed crop type segmentation of small holder farms in Germany, Ghana and South Sudan using data from Sentinel S1, S2 (10×10 m/px) and PlanetScope (3×3 m/px) satellite systems and time of year as an additional feature. Ref. [8] performed crop type mapping with a 30×30 m/px crop-specific annual land cover data combined from various satellite data sources for the area of Nebraska, US. Ref. [9] classified crop varieties from satellite time series data frames collected by the Chinese Gaofen missions with data resolutions from 4×4 m/px up to 15×15 m/px.

In the broader context of time series modelling with remote sensing data, several recent studies utilize spatio-temporal model architectures. The US county-scale soybean yield prediction by [6] was performed using a CNN-LSTM composite architecture, where a sequence of input frames was transformed into vectors of spatial features and then fed to an LSTM. Ref. [7] employed both a CNN-LSTM and a 3D-CNN model to perform crop type classification in Ghana and South Sudan, feeding multi-layer remote sensing time series data frames to the models. Ref. [8] built and trained a bidirectional ConvLSTM to predict crop maps from satellite data at the early stages of the growing season in Nebraska. While their main contribution was to affirm the feasibility of such model, they also employed their model in a CNN-LSTM setting, using pre-trained CNN called VGG11 [10] to extract

spatial features from sequences of past crop map images and then feeding these sets of spatial-like features further to the ConvLSTM. Ref. [9] used a 3D-CNN architecture in their crop type mapping study, feeding sequences of RGB image data from distinct areas to the network thus having the model learn both spatial and temporal features from the data. Ref. [11] built and trained a bidirectional ConvLSTM to automatically extract meaningful features from hyperspectral data consisting of several hundred bands for land cover pixel classification. They utilized the sequential modeling power of the ConvLSTM for feature extraction from individual images, feeding distinct bands to the model as if they were items in a sequence. Among other models, they also trained a 3D-CNN for the task to compare performance. Ref. [12] utilized a Gated Recurrent Unit (GRU) in building the convolutional recurrent model, i.e., ConvLSTM-like architecture. Their domain of application was in utilizing novel machine learning methodologies in performing land cover classification from satellite data. They employed their ConvLSTM-like model in parallel with a CNN to produce pixel-level land cover classification and report improved performance against widely utilized decision tree models for similar task. Ref. [13] employed the ConvLSTM in a encoder-decoder architecture to predict maize growth stage progression using several meteorological features using nationally collected meteorological data in China. The ConvLSTM was used as a feature extracting encoder while the decoder was an LSTM producing a desired output sequence. They modified the ConvLSTM to perform 1D convolutions on row-like data. A CNN-LSTM was also trained for comparative purposes. Ref. [14] compared the performance of 3D-CNNs against other deep learning architectures in the task of performing scene classification based on hyperspectral images. While the domain of their application is that of spectral-spatial and not spatio-temporal, they report 3D-CNN performing the best among other tested model compositions. Ref. [15] employ a wide array of CNN configurations to perform yield estimation using soil and nutrient information available pre-season arranged as spatial data. Most relevant to our study is their utilization of a 3D-CNN architecture which they use to ingest point-in-time data and learn salient features across varying input data rasters to estimate the crop yield.

1.2. Contribution

In contrast to studies performed at larger spatial scales, the main contribution of our study is to perform time series based intra-field yield prediction with multi-temporal data collected during the growing season with UAVs. In the context of using of remote sensing data in performing data-based modeling to aid in Smart Farming, we perform time series regression with remote sensing data, which is both collectable using commercially available UAVs and has spatial resolutions well below 1×1 m/px. We also use meteorological information, cumulative temperatures, to inform the models about change between weekly data. Our study builds on [2], introducing an extra variable to the modeling task, time, to see whether using time series data is more beneficial than using point-in-time data only. We develop, train and compare several spatio-temporal models to determine the most suitable model for intra-field yield modelling from a selection of models already utilized in the context of spatio-temporal modelling with remote sensing data. To also see if the spatio-temporal models can be used with a limited sequence of data from the beginning of the growing season, we evaluate the predictive capabilities and, thus, the usability, of the best performing model by feeding it time series data limited in this manner.

2. Materials and Methods

2.1. Data Acquisition

RGB images. Nine crop fields totaling to approximately 85 ha and having wheat, barley and oats as the crop varieties, were included in the study. The data was acquired during the year 2018 in the proximity of Pori, Finland (61°29′6.5″ N, 21°47′50.7″ E). Specific information about the fields is given in Table 1. The fields were imaged with a SEQUIOA (Parrot Drone SAS, Paris, France) multispectral camera mounted on a Airinov Solo 3DR (Parrot Drone SAS, Paris, France) UAV from the average

height fo 150 m using a minimum of three ground control points for each field and preflight color calibration. The imaging was done weekly, from week 21 to 35 and spanning 15 weeks in total. Due to weather conditions precluding UAV flight, gaps in data were present. For each field-specific set of images, a complete mosaic image of a field was constructed with Pix4D (Pix4D S.A., Prilly, Switzerland) software and cut to match the shape of the field boundaries. Radiometric correction was perofrmed using the illumination sensor of the Sequioa camera. The field image data was used as inputs to perform predictions with the considered models.

Table 1. The fields selected for the study in the proximity of Pori, Finland. The thermal time is calculated as the cumulative sum of temperature between the sowing and harvest dates. Mean yield has been calculated from processed yield sensor data for each field.

Field Number	Size (ha)	Mean Yield (kg/ha)	Crop (*Variety*)	Thermal Time	Sowing Date
1	11.11	4349.1	Wheat (*Mistral*)	1290.3	13 May
2	7.59	5157.6	Wheat (*Mistral*)	1316.8	14 May
3	11.77	5534.3	Barley (*Zebra*)	1179.9	12 May
4	11.08	3727.5	Barley (*Zebra*)	1181.3	11 May
5	7.88	4166.9	Barley (*RGT Planet*)	1127.6	16 May
6	13.05	4227.9	Barley (*RGT Planet*)	1117.1	19 May
7	7.61	6668.5	Oats (*Ringsaker*)	1223.4	17 May
8	7.77	5788.2	Barley (*Harbringer*)	1136.1	21 May
9	7.24	6166.0	Oats (*Ringsaker*)	1216.4	18 May

Weather data. The weather data was acquired from the open interface provided by the Finnish Meteorological Institute for Pori area. The thermal growing season started on 13th of April in 2018 and the cumulative temperature was calculated using that as the beginning date. As growth of crops is dictated by the accumulation of sunlight amongst other climatological, soil and nutrient variables, cumulative temperature was deemed robust enough indicator of interval between subsequent data collection days (instead of e.g., time in days). Being a common way to express crop growth phase, the cumulative temperature was utilized as a part of the input data to encode passing of time for the temporal models.

Yield data. As the target data, i.e., the data used as the ground truth for training the models, yield data was acquired during the harvest of each field. The harvesters were equipped with either a Trimble Navigation (Sunnyvale, CA, USA) CFX 750 or John Deere (Moline, IL, USA) Greenstar 1 yield mapping sensor systems. The systems produce a cloud of geolocated points with multivariate information about the harvest for each point in vector format. This data was first accumulated field-wise and then filtered to contain data points where the yield was between 1500 and 15,000 kg/ha and the speed of the harvester was between 2 and 7 km/h [2]. Finally, the yield map rasters were generated by interpolating the vector points over each field.

2.2. Data Preprocessing

The RGB images taken with the UAV and the cumulative temperatures for imaging dates were utilized as the input data with which the predictions about yields were performed. As spatial models

generally have a built in limitation of being able to utilize data with fixed dimensions only, data had to be clipped to smaller fixed dimension frames. As an intended side-effect, using smaller frames makes it possible to better model intra-field yield variability. Like in [2], the fields were split into smaller overlapping frames of size 40 × 40 m with a lateral and vertical step of 10 m. cumulative temperature was added as an additional layer in conjunction with the RGB-layers to have the data contain necessary information for temporal feature learning. The added layer contains constant values corresponding to the field and time of acquisition. The design choice of introducing this data as an additional layer was to have a single source of similarly constructed data for each model architecture.

During the extraction of frames we included every frame that had at least half of its data present at field edges into the final data set. The reasoning behind this was that the spatial models effectively learn filters that are applied over the spatial input data in a successive manner (see Section 2.3.1 for more). Thus, salient features are expected to be present in a frame albiet being just partial due to being located at a field's edge.

The data was also scaled to aid the models in their learning. All values were scaled to the range $[0, 1]$ using feature-wise maximum values as scalers. For the value ranges of unscaled input RGB data, the cumulative temperature calculated from the beginning of the thermal growing season and yield data, see Table 2. As the input data for this study was temporally sequential, the geolocationally matched frames were clipped across every image acquired at a different date for each field. Each sequence of frames was then coupled with geolocationally matching average yield.

Table 2. The value ranges of used input and target variables prior scaling.

Data	Min	Max	Mean	Std
RGB: R	105	254	186.0	19.5
RGB: G	72	243	154.3	18.8
RGB: B	58	223	126.7	18.9
Cumulative °C	388.6	2096	1192	545.0
Yield, kg/ha	1500	14,800	5287	1816

As the last step, the sequences of frames coupled with matching yield information were shuffled and split to training and hold out test data sets with 70%/30% ratio. The samples in the training set are used to optimize the model during the training. The test set is then utilized to evaluate model capabilities with previously unseen data, i.e., its generalization capabilities.

With the total number of generated sequences of frames being 2586, the training data set contained 1810 frame sequences (27,150 frames) and the test set 776 frame sequences (11,640 frames). The general process of generating the frames is depicted in Figure 1.

With the resolution of 0.325 px/m, a single spatial layer in the input data had the dimensions of 128 × 128 px. Using RGB-data with an additional layer constructed from the cumulative temperature conforming to the imaging date, a single frame of data consisted of four layers. With 15 frames, each frame corresponding to a particular week of the growing season, an input sequence of frames thus had the dimensions of $[15 \times 4 \times 128 \times 128]$.

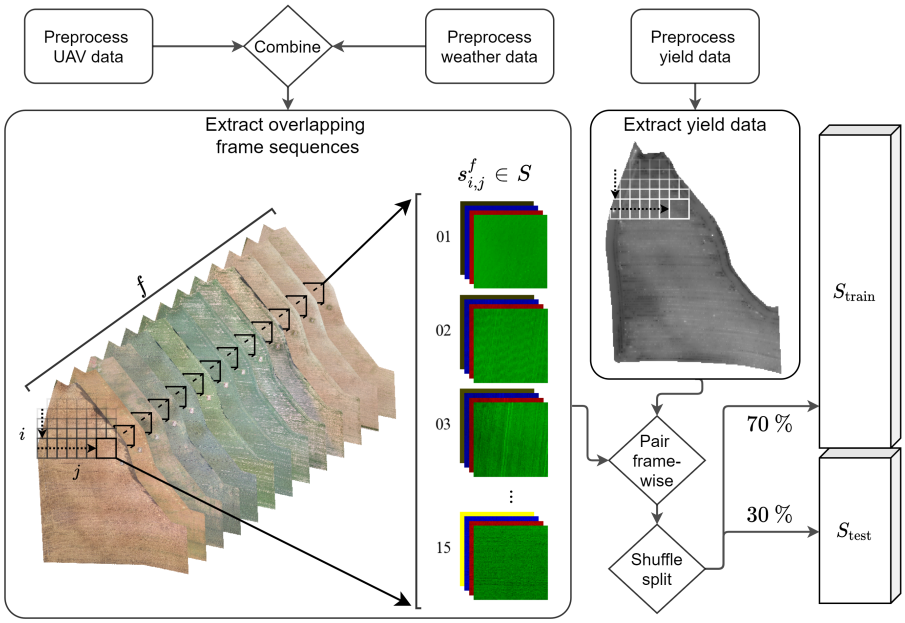

Figure 1. Input frame sequence and target average yield extraction process. Sequences of frames S of fixed width and height were extracted from cumulative temperature enhanced RGB image mosaic sequences as the input data, with f being a distinct field and $s^f_{i,j}$ an extracted sequence of frames from f. The four-layer YBRG, Y being the cumulative temperature, input frames were then geolocationally paired with corresponding yield data to form input-target pairs. Lastly, data was shuffled and split to training and test sets.

2.3. Model Architectures

2.3.1. Convolutional Neural Networks

Convolutional neural networks, often referred to as CNNs, have solidified their place in modeling tasks where the input data is either spatial or spatially representable [16,17]. The main component of the model is the convolution operation, where a set of trainable kernels (or filters) is applied to the input data resulting in a set of spatial features describing the data. For more in-depth explanation of the operations within a single convolution layer, like the application of convolution and pooling, see [2]. The model learns basic features in the first layers and composite features of these basic features at further layers [18]. To help the model better learn these features, batch normalization can be applied to the inputs [19]. The final output of a plain CNN is a set of feature maps. Depending on the use case, these can be either directly utilized or, for example, flattened and fed to a fully connected (FC) layer for regression or classification purposes.

2.3.2. Long Short-Term Memory Networks

The Long Short-Term Memory (LSTM) networks, originally introduced in [20], have been widely utilized in sequence modeling tasks [21]. There are two general concepts to the LSTM that help it in learning temporal features from the data. The first is the concept of memory, introduced as the cell state. The other is the concept of gates, effectively trainable FC layers, manipulating this cell state in response to the new inputs from the data and past outputs of the model. To handle sequences of data, the model loops over the sequences altering its cell (C) and hidden (H) states in the process using the combination of learned parameters in the gates and non-linear activations when combining the gate outputs. Following the Pytorch [22] implementation of LSTM, the following functions are computed:

$$
\begin{aligned}
g_t^i &= \sigma(W_x^I x_t + W_h^I h_{t-1}) \\
g_t^f &= \sigma(W_x^F x_t + W_h^F h_{t-1}) \\
g_t^c &= \tanh(W_x^C x_t + W_h^C h_{t-1}) \\
g_t^o &= \sigma(W_x^O x_t + W_h^O h_{t-1}) \\
C_t &= g_t^f \odot C_{t-1} + g_t^i \odot g_t^c \\
H_t &= O_t = g_t^o \odot \tanh(g_t^c)
\end{aligned}
\tag{1}
$$

where $g_t^{\{i,f,c,o\}}$ are the outputs of the input, forget, cell and output gates, respectively. The gates of the model contain its trainable parameters W. x_t denotes the external input and h_{t-1} the model's previous output. t denotes the current time step. C_t and H_t are the final computed cell and hidden states, respectively. The output O_t of the model is the last computed hidden state H_t. b are the bias factors and $\odot$ is the dot product. The general architecture of an LSTM is depicted in Figure 2.

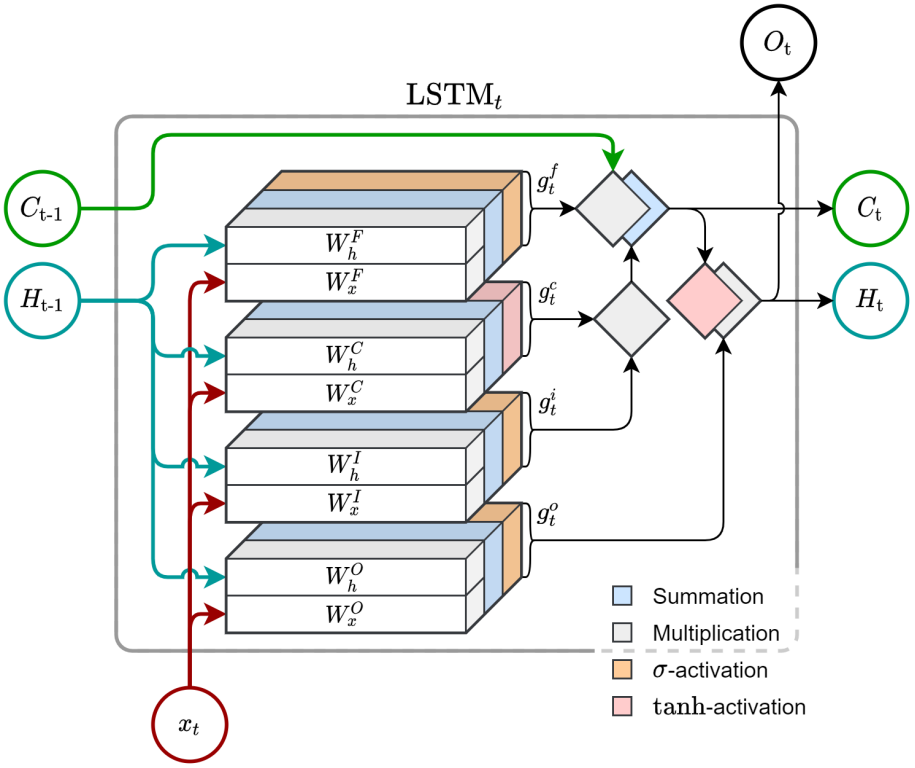

Figure 2. The inner architecture of an LSTM at a time step t. The model takes as its inputs the previous cell state C_{t-1} and hidden state H_{t-1} with the current item x_t of the input sequence. The H_{t-1} and x_t are then passed to forget (W^F), cell (W^C), input (W^I) and output (W^O) gates. These gates, effectively shallow FC layers, are responsible for determining what to keep from previous memory C_{t-1} accumulated from past experiences and what to incorporate to it as the current C_t. This is how the model is able to learn temporal features.

LSTMs can also be employed in bidirectional and stacked form. Bidirectional LSTMs train an additional model in comparison to the unidirectional LSTM presented in Figure 2. One LSTM reads the input from start of the sequence to end ($t_0 \rightarrow t_n$), while the other reads the input from end to start ($t_n \rightarrow t_0$). The outputs of these two parallel models are then combined as final temporal feature outputs [23]. When LSTMs are stacked, the first LSTM operates on the input sequence and subsequent LSTMs then operate on sequences of temporal feature outputs produced by

preceding models. Bidirectionality helps the model learn features from both sides of input sequences, while stacking helps in learning higher level temporal features [24].

2.3.3. CNN-LSTM

The CNN-LSTM is a composite model consisting of a spatial feature extractor or transformer, i.e., a pretrained CNN, and a temporal model, the LSTM [3]. The general idea is to both gain the ability to utilize spatial data and perform sequential modeling with LSTM networks.

The architecture of the pretrained CNN was implemented according to [2] with certain adaptations to have the model better serve as pre-trained spatial feature extractor of the composite CNN-LSTM. Firstly, the model was modified to accept four-band inputs. Secondly, the CNN layers were decoupled from the prediction-producing FC layers as a separate sub-module. Other than those two, the CNN consists of six convolutional layers with batch normalization in every layer and max pooling applied in the first and last layers. All convolution operations utilize 5×5 kernels with 128 kernels in the last operation and 64 in the ones preceding that. The convolutions are performed with zero padding to maintain constant dimensions. The maintaining of dimensions was initially implemented to allow adding an arbitrary number of in-between convolutional layers to the model without diminishing the intermediate hidden output dimensions to oblivion. The input max pooling uses a 8×8 and the output max pooling a 2×2 kernel. The output of the last convolutional layer is passed to a linear layer, squashing the hidden feature space to 256 features akin to [3]. These features are fed to the recurrent LSTM model. When pre-training the CNN only, the output of the squashing linear layer is fed to another linear layer producing the prediction outputs for error metric calculations. This outermost linear layer is omitted when the CNN is used as a part of the CNN-LSTM. The architecture of the spatial feature extracting CNN of the composite CNN-LSTM model is depicted in Figure 3.

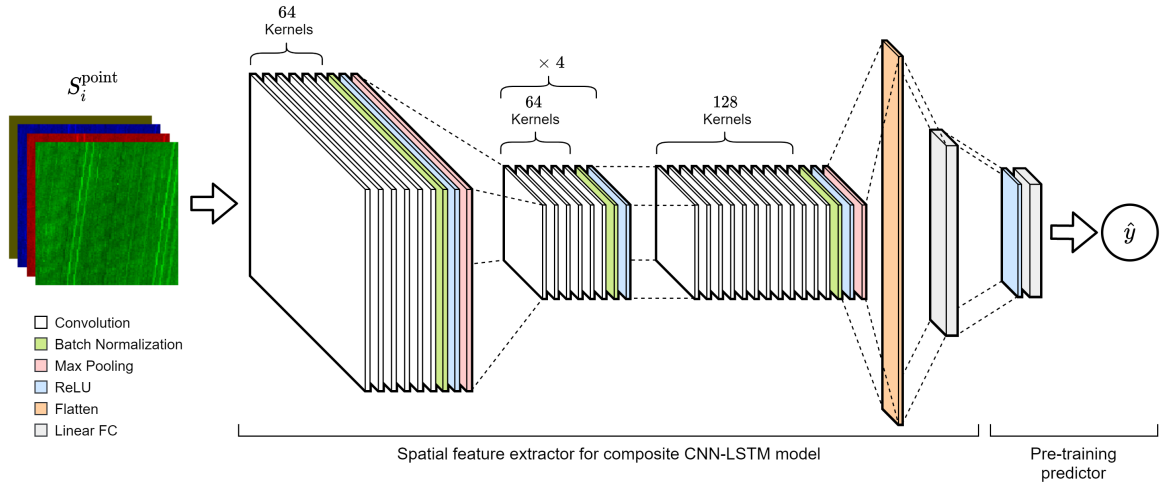

Figure 3. The spatial feature extracting CNN of the CNN-LSTM composite model, i.e., the pretrained CNN. The model is similar to the best performing model of [2]. Alterations in the FC layer composition had to be made to provide sufficient features for the LSTM utilizing the CNN as its input generator.

The temporal feature extracting part of the model is an LSTM, accepting sequences of spatial features as its inputs. During the hyperparameter optimization, we performed architectural experiments also with bidirectional and stacked (multi-layered) LSTMs. Generally, the option to use dropout [25], a regularization technique, is also part of the architectural implementation and that is also the case with Pytorch's LSTM implementation.

While the spatial feature extracting CNN could have been jointly trained with the LSTM, we chose to use a pre-trainend CNN to see whether the point-in-time spatial features could be utilized to perform sequential regression, similar to [10]. We selected this approach also to tie the composite model better

to the framework of the study by [2] by first training the CNN and then examining the ability of the LSTM to learn the temporal features in isolation from the CNN.

2.3.4. ConvLSTM

ConvLSTM [4] is a model combining the features of convolutional and sequential models into a single architecture, using convolutional layers (convolution with pooling etc.) as the LSTM's gate functions. This makes it possible to feed the sequential model the spatial data directly. Akin to how convolutional networks learn, the gates learn to utilize the convolutional kernels to provide the best set of spatial features when building and modifying the cell state C. Thus, contrary to the CNN-LSTM, no pre-extraction of spatial features for further spatial modelling is required. Our implementation of the recurrent architecture follows Equation (1).

From the point of architectural composition, the ConvLSTM is an LSTM at its essential core. In the ConvLSTM, using Figure 2 as a reference, the cell and hidden state altering gates $W^{\{F,C,I,O\}}$ have, however, been changed from conventional LSTM's shallow FC layers to shallow CNNs. To extract robust features from the input data, the gates W_x^* for inputs also employ a max pooling layer with a 3×3 kernel having padding and stride to halve input image dimensions after the first convolution. Due to the nature of CNNs learning spatial features in increasing complexity from layer-to-layer, we also allowed the model to utilize up to two convolutional layers for each W. Like with the CNN of the CNN-LSTM, we wanted to make sure that the intermediate feature map dimensions remaing unchanged, i.e., do not diminish as items in the sequences are processed. Thus, we used 32 convolutional kernels with 5×5 kernel shape and sufficient padding. The possibility to use batch normalization for inputs, stacking, bidirectionality and dropout were also implemented to find the best performing architectural composition.

2.3.5. 3D-CNN

As initially reported by [5], 3D-CNNs performed remarkably well in modeling tasks involving spatio-temporal data. Being CNNs, the 3D-CNNs utilize all same architectural features as more commonly used convolutional models. What's different is their use of convolution in the depth dimension, searching for robust features across sequences of input data in addition to spatial features extracted from the individual images. The sequential nature of input data is not limited to time, but can also be, for example, hyperspectral multi-layer point-in-time data with the aim of finding salient intra-band features [14]. The 3D-convolution is applied with a learnable three dimensional kernel, depicted in Figure 4. Kernel dimensions are in $[Z \times X \times Y]$ format, where Z denotes the time dimension.

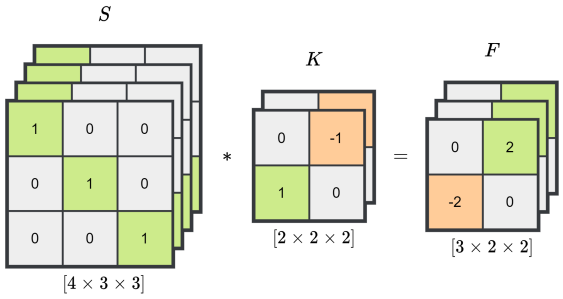

Figure 4. An illustration of 3D convolution. The 3D convolution operation effectively applies the kernel in one additional dimension cmompared to the normal convolution, the depth or z-axis. Like the input, the kernels are three dimensional. The dimensions of the feature map conform to how many times a kernel can wholly be applied to the input data along all three dimensions. With stride of one in each dimension, a $[2 \times 2 \times 2]$ kernel is applied on a $[4 \times 3 \times 3]$ input sequence of layers two times in x and y dimensions and thrice in z dimension, resulting in a $[3 \times 2 \times 2]$ feature map, its values being sums of products over distinct applications of K akin to 2D convolution.

The general architecture of the 3D-CNN we implemented conforms closely to how a CNN is generally constructed with the exception of using 3D instead of 2D convolutions. In the first layer the data is, however, grouped by layers as depicted in Figure 5. This is to have the model learn the spatio-temporal features of the data on a per-layer basis.

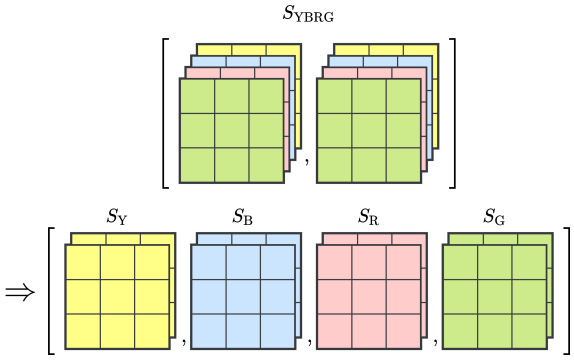

Figure 5. In the first layer of the 3D-CNN, the sequences of multi-layer input data are handled layer-wise. This helps the model first learn layer-wise spatio-temporal features, which are then composed as interlayer spatio-temporal features in the subsequent layers.

Our implementation follows the general CNN architecture of Figure 3. All layers prior to the last have the same number of kernels, the last having twice as many. We employ max pooling only in the first and last layers while the intermediate layers preserve intermediate feature map dimensions. The exact number of kernels is determined via hyperparameter tuning. As per [5], we perform convolutions with $[3 \times 3 \times 3]$ kernels with zero padding, having the pooling layers perform the diminishing of feature map dimensions. Mixing the depth-wise steps from [5] and spatial steps from [2], the first max pooling employs a $[1 \times 8 \times 8]$ kernel while the last a $[2 \times 2 \times 2]$ kernel. The kernels' strides equal respective kernel sizes, i.e., no overlap is applied. Like with the ConvLSTM, we the option to utilize batch normalization in every layer was also implemented for hyperparameter tuning purposes.

2.4. Training and Optimization

The process of training neural networks generally requires hyperparameter tuning. While model parameters, such as the layer-wise weights, are optimized during training in response to regression errors with the selected optimization algorithm, the hyperparameters are what dictate how the model is initialized and in what manner the optimization is applied. Examples of these hyperparameters include the learning rate and model depth. From available hyperparameter tuning methods we chose to use random search, in which a distribution is defined for each hyperparameter and then a value is randomly drawn for each distinct training [26].

We first performed the hyperparameter tuning for the pretrained CNN of the CNN-LSTM. Unlike sequential models, the pretrained CNN was fed single frames (i.e., point-in-time) drawn randomly from the set of all training data set frames. The goal was to have the model learn general spatial features for the whole growing season. Following [2], we used Adadelta [27] as the optimizer. Due to having input data consist of four distinct layers instead of only RGB layers, we performed tuning for the learning rate. Weight decay and the ρ coefficient were utilized from [2].

For the spatio-temporal models, we used Adam [28] as the optimizing algorithm for each model architecture akin to [7,8,11]. The spatio-temporal models were trained with frame sequences. Each model was tuned for LSTM and CNN architectural (where applicable) and optimizer hyperparamenters. The architectural and optimizer hyperparameters are given in Table 3. All hyperparameters were tuned simultaneously and not in sequential succession, meaning that the

hyperparameter values were drawn from their respective distributions for each hyperparameter at the start of a training.

Table 3. The model-specific hyperparameters and their distributions tuned during the random search. Square brackets indicate closed interval with lower and upper limit included, while curly brackets indicate a set from which a value was chosen from. The presence of $\vee$ indicates a boolean toggle with $p = 0.5$. The $\log_{10}$-uniform distribution used for the learning rate draws a value a from a float-uniform distribution in a given range to calculate 10^a.

Hyperparameter	Distribution	Pre-CNN	CNN-LSTM	ConvLSTM	3D-CNN
LSTM Architectural parameters					
LSTM layers	int-uniform	-	$[1, 3]$	$[1, 3]$	-
Dropout	float-uniform	-	$[0, 1] \vee 0$	$[0, 1] \vee 0$	-
Bidirectional	bool		$0 \vee 1$	$0 \vee 1$	-
CNN Architectural parameters					
CNN layers	int-uniform	-	-	$[1, 2]$	$[2, 5]$
Batch normalization	bool	-	-	$0 \vee 1$	$0 \vee 1$
Kernels	set	-	-	$\{32, 64, 128\}$	$\{32, 64, 128\}$
Optimizer parameters					
Learning rate	$\log_{10}$-uniform	$[-4, -1]$	$[-4, -2]$	$[-4, -2]$	$[-4, -2]$
L2-regularization	float-uniform	-	$[0, 1] \vee 0$	$[0, 1] \vee 0$	$[0, 1] \vee 0$

With each sequential model type we performed 300 distinct model training session, using random search for hyperparameter tuning and Skorch [29] as the training framework. For the pretraining of the CNN-LSTM's spatial feature extracting CNN, 50 models were trained to tune the learning rate due to the additional layer in inputs. The ρ-coefficient of the Adadelta algorithm and the weigth decay parameters were utilized from [2]. The total number of trained models was thus 950. The parameters of each model were initialized with *xavier*-uniform intialization [30]. During training, we utilized early stopping with patience for stagnant progress of 50 epochs. A single training iteration was allowed to continue for a maximum of 250 epochs. With continued training, where the best performing model parameter configuration is utilized as the starting point for a subsequent round of training, a model was allowed to be trained a maximum of 500 epochs. However, the use of early stopping was allowed to halt the training prior reaching that limit. Training was conducted with a separate training data set, having the training process utilize 5-fold cross-validation, where the training and validation batches are derived from the training data set. The final evaluation of a trained model was performed with the hold-out test data set. The models were trained in a distributed computation environment, utilizing Nvidia Tesla V100 Volta and Pascal architecture cloud GPUs.

3. Results

From the sets of trained models produced during the hyperparameter tuning process, the best performing models were singled out. During training we monitored the mean squared error (MSE) of the 5-fold cross validation. We also computed metrics for root mean square error (RMSE), mean absolute error (MAE) of unscaled targets, mean absolute percentage error (MAPE) and the coefficient of determination (R^2). The best performing model architecture was the 3D-CNN, expressing notably better performance with the best performing model than the rest of the trained architectures. The model performing worst was somewhat surprisingly the ConvLSTM, showing performance inferior even to the pretrained CNN trained with just point-in-time data. The performance metrics for the unscaled predicted and true target values for each model architecture are given as in Table 4.

Table 4. The performance metrics of the best-performing models resulting from model-specific hyperparameter tuning process with samples from the test set. The trained models were evaluated with a hold-out test set. Best performance was achieved with the 3D-CNN architecture. The number of trainable parameters indicate the model complexity. Best performance values are in bold text.

Model	Test RMSE (kg/ha)	Test MAE (kg/ha)	Test MAPE (%)	Test R^2 -	Trainable Parameters
Pretrained CNN	692.8	472.7	10.95	0.780	2.72×10^6
CNN-LSTM	456.1	329.5	7.97	0.905	2.94×10^6
ConvLSTM	1190.3	926.9	22.47	0.349	9.03×10^5
3D-CNN	**289.5**	**219.9**	**5.51**	**0.962**	7.48×10^6

The most consistently fitting sequential architecture was the CNN-LSTM in terms of test set performance with trained models. Other architectures produced occasional ill-fitted models with errors several magnitudes higher than their best performing counterparts. The RMSE percentiles depicting the general consistency in fitting for the spatio-temporal models are given in Table 5.

Table 5. The RMSE percentiles accross all trained spatio-temporal models. The RMSE percentiles indicate the consistency of a model architecture in generalizing to unseen samples with the training data. Out of the three, the CNN-LSTM was most consistent in how it was able to fit to the data and produce generalizable results. The training of other model architectures produced occasionally ill-fitted models. Best performance values are in bold text.

Model	Test RMSE (kg/ha)				
	Min	25%	50%	75%	Max
CNN-LSTM	456.1	**655.1**	**1475.6**	**1623.7**	$\mathbf{2.152 \times 10^3}$
ConvLSTM	1190.3	1477.8	1646.6	8750.2	1.334×10^6
3D-CNN	**289.5**	1355.4	1493.6	1649.0	1.926×10^6

Due to the training of the model architectures being a process of empirically evaluating randomly drawn hyperparameter sets, visualization of the hyperparameters against a performance metric further helps in understanding model fitting consistency. Out of the architectures, the CNN-LSTM and the 3D-CNN show similar behaviour in hyperparameter value distribution, the latter having a discernible dispersion in the values against the performance metric. ConvLSTM, as already stated, exhibits clearer sporadicity. The architecture-specific hyperparameter distributions plotted against the test RMSE are given in Figure 6.

The best performing configuration of hyperparameters dictating how a model is to be initialized and trained were sought by performing random search. In random search, each hyperparameter is assigned with a distribution, from which a value is drawn for each independent training of the model. The hyperparameters for the best performing models are given in Table 6.

In addition to performing comparative performance evaluation between the selected deep learning architectures with data sequences spanning the time from sowing to harvest, we also evaluated the performance of the best performing model configuration (architecture with hyperparameters) using data from an actionable time frame. In other words, we combined various configurations of input data sequences starting from image data acquisition dates closest to sowing (week 21) and ending at the midsummer (week 25). The following sequence configurations were built using the aforementioned time range:

- Weeks 21, 22, 23, 24, 25; five temporal frames.
- Weeks 21, 22, 23, 24; four temporal frames.
- Weeks 22, 23, 24, 25; four temporal frames.

- Weeks 21, 22, 23; three temporal frames.
- Weeks 23, 24, 25; three temporal frames.
- Weeks 21, 23, 25; three temporal frames.

We trained ten iterations of the best model configuration, the 3D-CNN, for each input sequence type to account for the effects of random model parameter initialization. The training was conducted as before, utilizing 5-fold cross-validation with the training data and testing the generalization capabilities with the hold-out test data, separately. The performance of these trained models with the test data are given in Table 7, where each row corresponds to a distinct configuration of input frame sequences. The best perfoming configuration in terms of RMSE and MAE is the four week long sequence taken from the beginning of the season (weeks 21 to 24). In terms of MAPE, the best performing configuration, however, consists of five weeks from the beginning of the season (weeks 21 to 25), although the difference to the four week sequence is small.

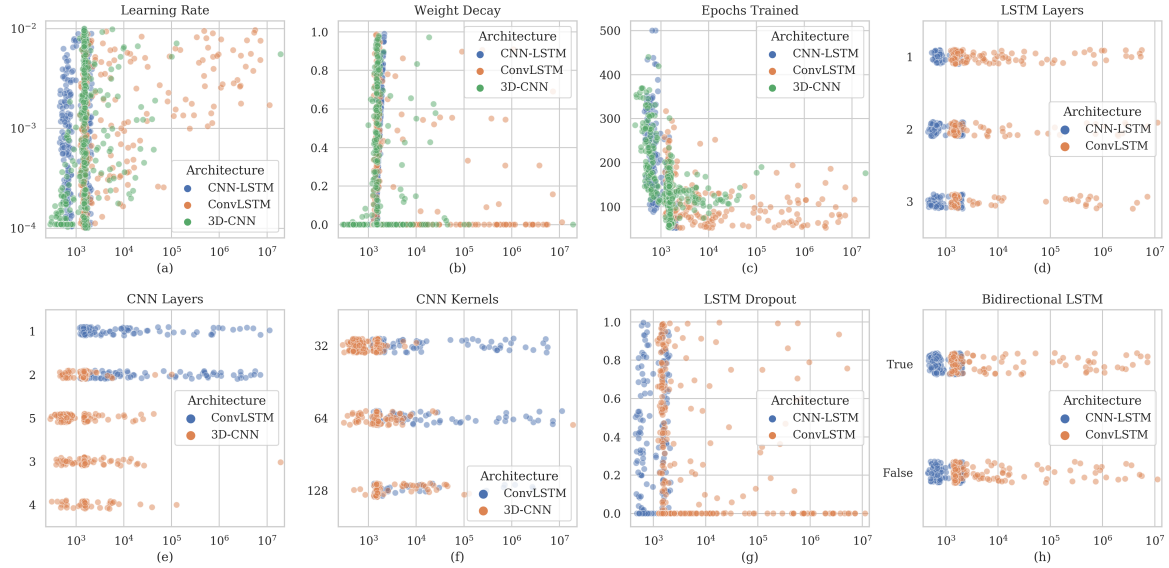

Figure 6. The architecture-specific distributions of hyperparameters against the test RMSE (x axis). (**a**,**b**) are the optimizer hypeparameters and (**c**) is the training length in epochs. (**d**,**g**,**h**) are the LSTM architectural hyperparameters, while (**e**,**f**) are the CNN architectural hyperparameters. (**d**,**e**,**f**,**h**) contain categorical values in y axis with values spread category-wise for easier observation of clustering. The rest of the sub-figures have a continuous y axis.

Table 6. The architecture specific hyperparameter values for the best performing models. The value types conform to the values given in Table 3. The feature extracting CNN of the CNN-LSTM was not tuned for hyperparameters as tuning results from previous study were utilized.

Hyperparameter	**Pre-CNN**	**CNN-LSTM**	**ConvLSTM**	**3D-CNN**
		LSTM Architectural parameters		
LSTM layers	-	2	2	-
Dropout	-	0.5027	0.9025	-
Bidirectional	-	0	1	-
		CNN Architectural parameters		
CNN layers	6 *	-	1	5
Batch normalization	Yes *	-	No	No
Kernels	128/64 *	-	32	32
		Optimizer parameters		
Learning rate	1.000×10^{-1}	7.224×10^{-4}	1.361×10^{-3}	1.094×10^{-4}
L2-regularization	0.9 *	0.0	0.0	0.0

* Values taken from [2].

Table 7. Retraining results of the best performing 3D-CNN configuration with various input sequence configurations from the test set. The input data was constructed from the first five imagings (weeks 21 to 25). The composition of weekly data was varied and variations evaluated by fitting the best performing 3D-CNN configuration to each variation. Best performance values are in bold text.

Weeks in Input Sequence	Test RMSE (kg/ha)	Test MAE (kg/ha)	Test MAPE (%)	Test R^2 -
21, 22, 23, 24, 25	413.8	320.6	**7.04**	0.921
21, 22, 23, 24	**393.9**	**292.8**	7.17	**0.929**
22, 23, 24, 25	439.3	343.0	7.90	0.911
21, 22, 23	543.5	421.4	10.02	0.864
23, 24, 25	425.0	326.6	8.25	0.917
21, 23, 25	478.1	369.3	8.72	0.895

Operating with single frames, the models can be used to construct predictions for whole fields. This is achieved by extracting frames from an image of the fields and feeding them as inputs to the model. Re-arranging the predictions to original field shape yields a map of frame-wise yield predictions. The performance of the best performing 3D-CNN configuration with both full length and shortened sequences is illustrated in Figure 7 with a 10 m step between predicted points. As the test set was constructed from frame sequences randomly taken from all extracted frame sequences, the illustrations contain frames from both the training and the test set.

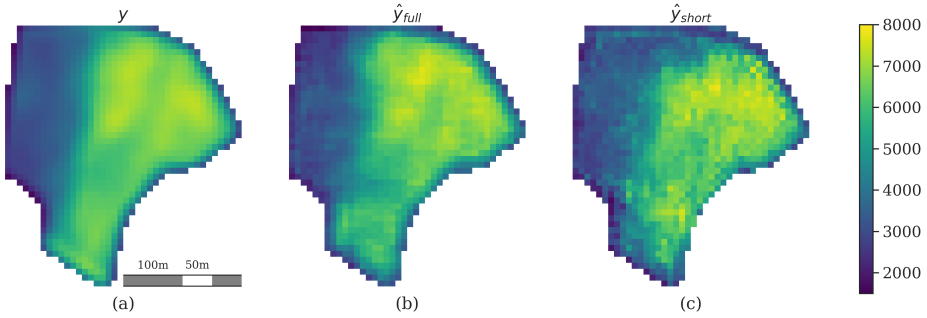

Figure 7. Frame-based 3D-CNN model performances against true yield data. (**a**) is the true yield map of the field. (**b**) is the modelled prediction, utilizing the full length frame sequences. (**c**) is then the actual in-season prediction utilizing four first frames of the weekly frame sequence. Units are absolute values in crop yield kg/ha. One pixel in the images corresponds to a 10 × 10 m area. Images are unsmoothed, represent the values as they were produced and contain samples from both training and test sets due to how the sets were constructed.

4. Discussion

In this study we evaluated the feasibility of using spatio-temporal deep learning architectures in modelling crop yield at the intra-field scale. Using sequences of UAV and weather data collected in the vicinity of Pori, Finland, during the growing season of 2018, we split the fields to geolocationally matched temporal sequences of frames of fixed width and height. We developed and trained three different model architectures: CNN-LSTM, ConvLSTM and 3D-CNN. We first determined the best performing architecture by performing hyperparameter tuning with complete temporal sequences of frames (15 time steps). With the best performing model architecture and hyperparameter configuration we then evaluated the predictive capabilities of the models by using a shorter temporal sequence of frames from the beginning of the growing season.

Of the architectures, the 3D-CNN performed the best in full sequence modelling. The best performing model consisted of five 3D-CNN layers using 32 kernels in the layers. Other architectural configurations are given in Section 2.3.5. The model attained 218.9 kg/ha test MAE, 5.51% test MAPE and 0.962 test R^2-score. Compared to the study presented in [2] using just a point-in-time single

frame predictor with 484.3 kg/ha MAE and 8.8% MAPE, the modelling performance was improved by 265.4 kg/ha MAE (54.8% improvement) and 3.29% MAPE (37.4% improvement). In terms of prediction performance with smaller sequences from the beginning of the season, the 3D-CNN performed the best using four first frames of the whole input sequences. With a shorter sequence the model attained 292.8 kg/ha test MAE, 7.17% test MAPE and 0.929 test R^2-score. The respective improvements to the best performing model presented in [2] were 191.5 MAE (39.5% improvement) and 1.63% MAPE (18.5% improvement).

Recent studies make use of UAVs in a variety of imaging applications. The use of UAVs has become more common, as shown in [31], a review of UAV thermal imagery applications in the domain of precision agriculture. One of the reasons is the increased need of performing classification and regression at scales more accurate than what is attainable by publicly available satellite data sources. However, the use of point-in-time UAV data is common. Ref. [32] utilized UAVs to gather hyperspectral data of potato tuber growth at the resolution of 2.5 cm/px. They utilized traditional ML methods, such as linear models and decision trees, to perform tuber yield estimation using individual data points gathered in-season at the intra-field scale, achieving 0.63 R^2-score for the tuber yield prediction accuracy with a Ridge regression. Ref. [33] used UAV to collect multispectral data from wheat and corn fields to estimate intra-field crop nitrogen content using linear regression and point samples—spatial features were not utilized. They fit multiple linear models to wheat and corn and attained 0.872 R^2-score on average. Ref. [34] performed wheat leaf area index and grain yield estimation with various vegetation indices derived from point-in-time multispectral UAV data using multiple machine learning methods, neural networks included. The highest performance they attained was 0.78 R^2-score with a Random Forest. However, they fed the input data as point samples.

Satellites perform frequent overflights over vast areas accross the globe. They are thus an ideal source of automatically generated multi-temporal remote sensing data [35]. This is one of the reasons, why spatio-temporal modelling is more notably present in the context of publicly available satellite data sources, contrary to UAV data requiring manual collection. Spatio-temporal models akin to the setting of our study have been utilized in various modelling tasks with remote sensing data in the domain of agriculture. Performing county-scale soybean yield prediction, ref. [6] used a CNN, an LSTM and a composite CNN-LSTM to model soybean yield with in-season satellite data. They achieved an average 0.78 R^2-score with the spatio-temporal CNN-LSTM model. Their input data resolutions were from 500 $\times$ 500 m/px to 1 $\times$ 1 km/px. Ref. [7] performed crop type classification in Europe and Africa with multi-temporal satellit data at resolutions from 3 $\times$ 3 m/px to 10 $\times$ 10 m/px. They attained F1 scores 91.4 for the CNN-ConvLSTM and 90.0 for the 3D-CNN, averaged over crop types in their Germany data set. Ref. [8] performed pre-season crop type mapping for the area of Nebraska, US, employing a CNN-ConvLSTM to extract spatio-temporal features from multi-temporal multi-satellite composite data set. Using prior years of crop type related data to predict a map of crop types, they attained an average accuracy of 77% across all crop types in their data. The data was processed to a resolution of 30 $\times$ 30 m/px. Ref. [9] utilized a 3D-CNN to classify crop types from multi-temporal satellite data gathered from an area within China, acquiring a classification accuracy of 98.9% with the model. Their input data resolutions were from 4 $\times$ 4 m/px to 15 $\times$ 15 m/px. Ref. [36] performed weekly UAV image collections in a controlled field experiment with soybeans, performing seed yield prediction with multiple linear models fit the multi-temporal data. Thus, spatio-temporal modelling with novel techniques was not performed. With seed yield prediction, they achieved 0.501 adjusted R^2 score. The resolution of their data was 1.25 $\times$ 1.25 cm/px.

In remote sensing, the multitemporal aspect of satellite sensor data the has been well studied. In their review of the applications of multisource and multitemporal data fusion in remote sensing, ref. [37] show how studies utilizing the temporal feature of satellite data are rather common. However, in terms of models and data usage settings, they only briefly mention how novel deep learning architectures have only recently been applied in this data domain. They cite that both data and methods, especially the latter, are still under development and a subject of further research. While some

studies have not found additional benefit in using multitemporal data [38], partly due to selected data utilization techiniques, others find benefit over using just point-in-time data [33,35–37].

Regarding the poor performance of the ConvLSTM in our study, the studies by [7,8] might provide some basis for understanding the phenomenon. In both studies, the ConvLSTM was preceded with an exclusively spatial feature extracting CNN model. The extracted feature maps were then fed to the ConvLSTM for temporal feature extraction. While in our study we experimented with multiple convolutional layers in the ConvLSTM model, it could very well be that using a pre-trained CNN akin to CNN-LSTM is required for the ConvLSTM as well. Model complexity is another way to look at this, as the 3D-CNN model was more complex compared to the ConvLSTM. This indicates that the effective capacity of the ConvLSTM might indeed be too low. Thus, increasing the effective capacity by either adding a spatial feature pre-extractor or increasing the gate-wise layer count could increase the performance of this model architecture in similar study setting.

As we utilized weather information at city-scale, the precision of change in the growth phase could be further improved with specifically located weather stations. Weather stations located in the approximate vicinity of the fields under scrutiny could provide better and more accurate measurements of the local temperatures and other climatological variables and thus might help the model produce even better predictions when sequences are involved. Using other data sources, such as soil information and topology maps, could also be further utilized to improve the predictive capabilities of the model. As growing season provides information about how the crops have concretely developed, the soil and topology maps provide more in terms of a prior that the UAV images are then used to further develop as new samples emerge.

A limitation to our study is the use of aggregated crop type data collected from various fields. Using a single model to predict for wheat, barley and oats prohibits both the inference with and the performance analysis of the model on a per-crop basis. Additionally, the remote sensing data based modelling approach doesn't take into account any existing crop growth models. Those could well be utilized to further provide better performance, akin to what has been done in [36], but this is outside the scope of our study. That being said, the modelling task of this study was not that of crop growth, but yield estimation with UAV remote sensing data.

5. Conclusions

Our study seeks to combine three increasingly common but yet seldom co-utilized concepts in the domain of crop yield estimation: the use of high resolution UAV image data, time series regression and novel spatio-temporal neural network architectures. It has already been shown that crop yield prediction with spatial neural networks, i.e., CNNs, is feasible and produces results accurate enough for performing actions in-season [2]. In this study, we show that adding time as an additional feature not only improves the modelling performance with UAV RGB data (see Table 4) but also improves the predictive capabilities (see Table 7). Furthermore, using weekly UAV data gathered during the first month provides enough data for the model to build an accurately predicted yield map from which to draw further conclusions.

To conclude, the use of multitemporal remote sensing data is not only common but also beneficial in crop yield modelling and prediction. Furthermore, the easy accessibility of commercially available UAVs with mounted RGB sensors enables image data acquisition in higher resolutions compared to satellites. This in turn opens up the possibilities to perform modelling and predictions at intra-field scale. As shown in our study, the use of UAV-based data and proper spatio-temporal deep learning techniques is an enabler of more sophisticated Decision Support Systems in the domain of agriculture.

Author Contributions: Conceptualization, P.N. and N.N.; methodology, P.N. and N.N.; software, P.N.; validation, P.N., N.N. and T.L.; formal analysis, P.N. and N.N.; investigation, P.N.; resources, P.N., N.N. and P.L.; data curation, P.N., N.N. and P.L.; writing—original draft preparation, P.N.; writing—review and editing, P.N., N.N. and T.L.; visualization, P.N.; supervision, N.N. and T.L.; project administration, P.N., N.N. and T.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by Mtech Digital Solution Oy, Vantaa, Finland.

Acknowledgments: We would like to thank Tampere University for providing the computational resources and MIKÄ DATA project for providing us with the data.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Narra, N.; Nevavuori, P.; Linna, P.; Lipping, T. A Data Driven Approach to Decision Support in Farming. In *Information Modelling and Knowledge Bases XXXI*; IOS Press: Amsterdam, The Netherlands, 2020; Volume 321, pp. 175–185. [CrossRef]
2. Nevavuori, P.; Narra, N.; Lipping, T. Crop yield prediction with deep convolutional neural networks. *Comput. Electron. Agric.* **2019**, *163*, 104859. [CrossRef]
3. Sainath, T.N.; Vinyals, O.; Senior, A.; Sak, H. Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks. In Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brisbane, Australia, 19–24 April 2015; pp. 4580–4584.
4. Shi, X.; Chen, Z.; Wang, H.; Yeung, D.Y.; Wong, W.K.; Woo, W.C. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 802–810.
5. Tran, D.; Bourdev, L.; Fergus, R.; Torresani, L.; Paluri, M. Learning spatiotemporal features with 3D convolutional networks. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 4489–4497.
6. Sun, J.; Di, L.; Sun, Z.; Shen, Y.; Lai, Z. County-Level Soybean Yield Prediction Using Deep CNN-LSTM Model. *Sensors* **2019**, *19*, 4363. [PubMed]
7. Rustowicz, R.; Cheong, R.; Wang, L.; Ermon, S.; Burke, M.; Lobell, D. Semantic Segmentation of Crop Type in Africa: A Novel Dataset and Analysis of Deep Learning Methods. In Proceedings of the CVPR Workshops, Long Beach, CA, USA, 16–20 June 2019; pp. 75–82.
8. Yaramasu, R.; Bandaru, V.; Pnvr, K. Pre-season crop type mapping using deep neural networks. *Comput. Electron. Agric.* **2020**, *176*, 105664. [CrossRef]
9. Ji, S.; Zhang, C.; Xu, A.; Shi, Y.; Duan, Y. 3D Convolutional Neural Networks for Crop Classification with Multi-Temporal Remote Sensing Images. *Remote Sens.* **2018**, *10*, 75. [CrossRef]
10. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), San Diego, CA, USA, 7–9 May 2015.
11. Liu, Q.; Zhou, F.; Hang, R.; Yuan, X. Bidirectional-Convolutional LSTM Based Spectral-Spatial Feature Learning for Hyperspectral Image Classification. *Remote Sens.* **2017**, *9*, 1330. [CrossRef]
12. Ienco, D.; Interdonato, R.; Gaetano, R.; Ho Tong Minh, D. Combining Sentinel-1 and Sentinel-2 Satellite Image Time Series for land cover mapping via a multi-source deep learning architecture. *ISPRS J. Photogramm. Remote Sens.* **2019**, *158*, 11–22. [CrossRef]
13. Yue, Y.; Li, J.H.; Fan, L.F.; Zhang, L.L.; Zhao, P.F.; Zhou, Q.; Wang, N.; Wang, Z.Y.; Huang, L.; Dong, X.H. Prediction of maize growth stages based on deep learning. *Comput. Electron. Agric.* **2020**, *172*, 105351. [CrossRef]
14. Li, Y.; Zhang, H.; Shen, Q. Spectral–Spatial Classification of Hyperspectral Imagery with 3D Convolutional Neural Network. *Remote Sens.* **2017**, *9*, 67. [CrossRef]
15. Barbosa, A.; Trevisan, R.; Hovakimyan, N.; Martin, N.F. Modeling yield response to crop management using convolutional neural networks. *Comput. Electron. Agric.* **2020**, *170*. [CrossRef]
16. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
17. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9. [CrossRef]
18. Zeiler, M.D.; Fergus, R. Visualizing and Understanding Convolutional Networks. In *Computer Vision—ECCV 2014*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2014; Volume 8689, pp. 818–833.
19. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv* **2015**, arXiv:1502.03167.

20. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
21. Schmidhuber, J. Deep Learning in Neural Networks: An Overview. *Neural Netw.* **2014**, *61*, 85–117. [CrossRef] [PubMed]
22. Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; Lerer, A. Automatic differentiation in PyTorch. In Proceedings of the NIPS-W, Long Beach, CA, USA, 4–9 December 2017.
23. Schuster, M.; Paliwal, K.K. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.* **1997**, *45*, 2673–2681. [CrossRef]
24. Graves, A. Generating Sequences with Recurrent Neural Networks. *arXiv* **2013**, arXiv:1308.0850. [CrossRef]
25. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958. [CrossRef]
26. Bergstra, J.; Bengio, Y. Random Search for Hyper-Parameter Optimization. *J. Mach. Learn. Res.* **2012**, *13*, 281–305. [CrossRef]
27. Zeiler, M.D. ADADELTA: An Adaptive Learning Rate Method. *arXiv* **2012**, arXiv:1212.5701. [CrossRef]
28. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *ICLR* **2014**, 1–15. [CrossRef]
29. Tietz, M.; Fan, T.J.; Nouri, D.; Bossan, B.; Skorch Developers. Skorch: A Scikit-Learn Compatible Neural Network Library That Wraps PyTorch. 2017. Available online: https://skorch.readthedocs.io/ (accessed on 16 October 2020).
30. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. *J. Mach. Learn. Res.* **2010**, *9*, 249–256.
31. Messina, G.; Modica, G. Applications of UAV thermal imagery in precision agriculture: State of the art and future research outlook. *Remote Sens.* **2020**, *12*, 1491. [CrossRef]
32. Sun, C.; Feng, L.; Zhang, Z.; Ma, Y.; Crosby, T.; Naber, M.; Wang, Y. Prediction of end-of-season tuber yield and tuber set in potatoes using in-season uav-based hyperspectral imagery and machine learning. *Sensors* **2020**, *20*, 5293. [CrossRef] [PubMed]
33. Lee, H.; Wang, J.; Leblon, B. Intra-Field Canopy Nitrogen Retrieval from Unmanned Aerial Vehicle Imagery for Wheat and Corn Fields. *Can. J. Remote Sens.* **2020**, *46*, 454-472. [CrossRef]
34. Fu, Z.; Jiang, J.; Gao, Y.; Krienke, B.; Wang, M.; Zhong, K.; Cao, Q.; Tian, Y.; Zhu, Y.; Cao, W.; et al. Wheat growth monitoring and yield estimation based on multi-rotor unmanned aerial vehicle. *Remote Sens.* **2020**, *12*, 508. [CrossRef]
35. Liu, S.; Marinelli, D.; Bruzzone, L.; Bovolo, F. A review of change detection in multitemporal hyperspectral images: Current techniques, applications, and challenges. *IEEE Geosci. Remote Sens. Mag.* **2019**, *7*, 140–158. [CrossRef]
36. Borra-Serrano, I.; Swaef, T.D.; Quataert, P.; Aper, J.; Saleem, A.; Saeys, W.; Somers, B.; Roldán-Ruiz, I.; Lootens, P. Closing the phenotyping gap: High resolution UAV time series for soybean growth analysis provides objective data from field trials. *Remote Sens.* **2020**, *12*, 1644. [CrossRef]
37. Ghamisi, P.; Rasti, B.; Yokoya, N.; Wang, Q.; Hofle, B.; Bruzzone, L.; Bovolo, F.; Chi, M.; Anders, K.; Gloaguen, R.; et al. Multisource and multitemporal data fusion in remote sensing: A comprehensive review of the state of the art. *IEEE Geosci. Remote Sens. Mag.* **2019**, *7*, 6–39. [CrossRef]
38. Hauglin, M.; Ørka, H.O. Discriminating between native norway spruce and invasive sitka spruce—A comparison of multitemporal Landsat 8 imagery, aerial images and airborne laser scanner data. *Remote Sens.* **2016**, *8*, 363. [CrossRef]