ETH zürich

PRS Photogrammetry Remote Sensing

Interdisciplinary Project Work

Crop Classification
with Neural Ordinary Differential Equations

Nando Metzger

18^{st} of December 2020

Chair of Photogrammetry and Remote Sensing
ETH Zurich

in collaboration with: Swiss Federal Office for Agriculture

Professorship
Prof. Dr. Konrad Schindler

Supervision
Özgür Mehmet Türkoglu
Dr. Jan Dirk Wegner

Abstract

Optical satellite sensors cannot see the Earth's surface through clouds. Despite the periodic revisit cycle, image sequences acquired by Earth observation satellites are therefore *irregularly* sampled in time. State-of-the-art methods for crop classification (and other time series analysis tasks) rely on techniques that implicitly assume regular temporal spacing between observations, such as recurrent neural networks (RNNs). We propose to use neural ordinary differential equations (NODEs) in combination with RNNs to classify crop types in irregularly spaced image sequences. In a later stage, we expand our model to work with convolutional layers to also incorporate the spatial character of the satellite images. The resulting ODE-RNN and CONV-ODE-RNN models consist of two steps: an update step, where a recurrent unit assimilates new input data into the model's hidden state; and a prediction step, in which NODE propagates the hidden state until the next observation arrives. The prediction step is based on a continuous representation of the latent dynamics, which has several advantages. At the conceptual level, it is a more natural way to describe the mechanisms that govern the phenological cycle. From a practical point of view, it makes it possible to sample the system state at arbitrary points in time, allowing one to integrate observations whenever they are available, and extrapolate beyond the last observation. Our experiments show that ODE-RNN and CONV-ODE-RNN indeed improve classification accuracy over common baselines such as LSTM, GRU, and temporal convolution. The gains are most prominent in the challenging scenario where only a few observations are available (i.e., frequent cloud cover). Moreover, we show that the ability to extrapolate translates to better classification performance early in the season, which is important for forecasting.

Acknowledgement

I would like to thank the following people:

- Prof. Dr. Konrad Schindler and Dr. Jan Dirk Wegner for the opportunity of letting me work on this project possible.
- Mehmet Özgür Türkoglu for his continuous support in the supervision of this project.
- The Swiss Federal Office for Agriculture (FOAG) which partially funded this research through project *DeepField*.

Contents

List of Figures

List of Tables

1 Introduction

Monitoring of agricultural lands is important to manage food production, biodiversity, and forestry, among others. An increasing world population and changes in consumer habits require new agricultural areas or intensification of existing ones: ≈38% of the Earth's land surface is already covered with crops and pastures (Foley et al. 2005). Cropland expansion and more intensive agriculture are linked to ecological problems like deforestation, biodiversity loss, and soil degradation. Crop monitoring supports land management to minimize such negative impacts.

With the increasing availability of high-resolution satellite data and annotations, crop classification with machine learning methods has made great progress. Like vegetation in general, agricultural crops have class-specific spectral properties depending on soil structure (soil brightness, roughness), vegetation architecture (LAI, leaf angle, etc.), and leaf biochemistry (chlorophyll, water content, nitrogen content, etc.). Importantly, the reflectance of agricultural crops varies strongly due to their pronounced phenological cycle (Garnot et al. 2019; Hu et al. 2017), so time series modeling is essential to achieve good classification. Recurrent neural network (RNN) architectures are a powerful tool for sequence modeling, and several state-of-the-art crop mapping methods are based on RNNs (M Rustowicz et al. 2019; Rußwurm and Körner 2017, 2018; Turkoglu et al. 2019). A main challenge when working with satellite time series in the optical spectrum (e.g, Sentinel-2 or Landsat) are atmospheric effects, in particular occlusion by clouds. If the region of interest is overcast during image acquisition, then the corresponding observations do not contain any signal from the crop on the ground, see Figs. 1 and 2. RNNs can, in principle, learn to discard uninformative parts of the input. However, their performance degrades (Che et al. 2018; Tan et al. 2020) when uninformative observations are frequent and arbitrarily scattered across the time series (as often the case in repeat-pass satellite data). One reason for this behavior is that RNNs are designed for discrete data sampled at regular intervals – in our case with equal temporal spacing. RNNs regard the individual observations as an ordered sequence, but beyond the sequence order they are unaware of time, as they do not have a "clock". They cannot represent the dynamics of a system that is observed with variable temporal spacing, since the state update depends only on the previous state and observation, without any notion when that state is reached.

In this project, we deal with the classification of spectral time series into crop types, in the presence of varying cloud cover. To handle the changing and uneven data gaps due to clouds, we employ a method based on Neural Ordinary Differential Equations (NODE, Chen et al. 2018). NODEs encode continuous dynamics that are, by themselves,

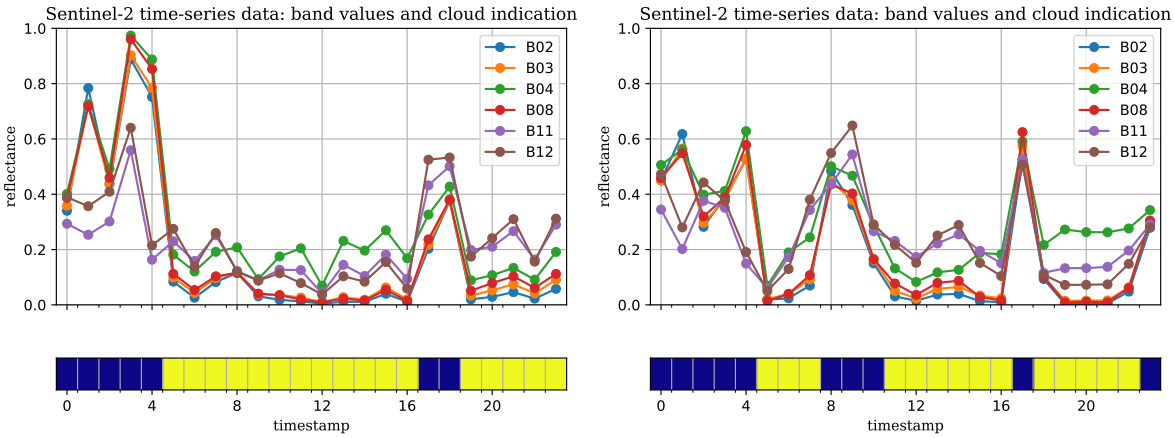


Figure 1: Examples of Sentinel-2 time series data from TUM dataset (left for *winter wheat*, right for *corn*). Observations obscured by clouds are marked in blue at the bottom. Note their irregular distribution.

independent of observations, and have a metric notion of time, so that observations can be included with conventional RNN operations whenever they are available. Hence, uninformative observations can simply be skipped. Furthermore, we expand the model with convolutional layers to model the spatial dimension of the satellite images. We validate the proposed method on two different datasets, in combination with two different (arguably, the two most popular) recurrent architectures, namely Long Short-Term Memory (LSTM, Hochreiter et al. 1997) units and Gated Recurrent Units (GRU, Chung et al. 2014). To the best of our knowledge, our work is the first to apply neural ordinary differential equations for time series analysis in remote sensing. We show that the application of NODEs to satellite image time series is fairly straight-forward, opening up a host of potential applications. We believe that NODEs may be beneficial for many remote sensing tasks beyond crop mapping, since the problem of uninformative observations and/or data gaps is a ubiquitous, general problem of optical earth observation.

Figure 2: Example of Sentinel-2 image time series from TUM dataset (visualization using bands B02, B03, B04). The last image is the ground-truth map where different colors correspond to different crop types.

2 Related Work

Classification of crops from satellite image data has been studied intensively in the remote sensing literature. In early work, rule-based systems (Conrad, Fritsch, et al. 2010) or classical machine learning (Inglada et al. 2015; Wardlow et al. 2008) were used. Such methods mostly relied on handcrafted features such as the Normalized Difference Vegetation Index (NDVI) (Conrad, Fritsch, et al. 2010; Foerster et al. 2012; Peña-Barragán et al. 2011; Ustuner et al. 2014). To model the temporal dynamics, methods used beyond consensus voting or concatenation, also employed floating windows (Conrad, Dech, et al. 2014), hidden Markov models (Bailly et al. 2018; Siachalou et al. 2015) and dynamic warping (Belgiu et al. 2018).

More recent approaches tend to avoid handcrafted features and employ deep learning. Rußwurm and Körner 2017 investigate the use of Long Short-Term Memory (LSTM) networks to model the temporal evolution of different crop types' spectral characteristics. In follow-up work, a convolutional version of LSTM is used to learn spatio-temporal patterns of the crop classes (Rußwurm and Körner 2018). Other popular neural architectures have also been used to represent temporal dependencies; such as temporal convolutional networks (TCNs, Lea et al. 2016), which employ convolutions along the time dimension (Pelletier et al. 2019), a 3D convolutional neural network that models both spatial and temporal dimension (Ji et al. 2018), and transformers, which instead use a learned attention mechanism to focus on informative time steps in the sequence (Rußwurm, Lefèvre, et al. 2019).

A recurring question when working with time series of observations is how to handle missing data. A frequent strategy is imputation, i.e., explicitly filling the gaps by predicting the missing values. Many different variants exist, including simple (e.g., linear or spline) interpolation, supervised learning methods such as k-nearest neighbours (Batista et al. 2002) or Random Forests (Stekhoven et al. 2012), and algebraic methods for low-rank matrix completion (Koren et al. 2009; Mazumder et al. 2010). These methods have also been used in remote sensing. In Nguyen et al. 2018 the authors used a k-NN imputation techniques to estimate the forest biomass from Landsat time series, similarly Brooks et al. 2012 predicted the missing data by leveraging Fourier analysis.

There have been attempts to extend recurrent networks and propagate information through data gaps. In the simplest case, missing values are replaced by the last observed value or the dataset mean, but it has also been proposed to gradually decay the input towards the mean (Che et al. 2018). That work also suggests appending the time that has passed since the previous observation to the input. Tresp et al. 1998 integrate an RNN with a Kalman filter, whereas GRU-D uses exponential decays to model the behavior of the variables over time. For remote sensing images, it has been proposed to reconstruct the missing data with a convolutional encoder-decoder network, exploiting the fact that gaps with associated ground truth are trivial to simulate for training (Zhang et al. 2018).

Neural Ordinary Differential Equations (NODEs) (Chen et al. 2018) can be used as a generic building block to integrate a self-contained dynamic model into deep learning architectures, see below. By themselves, NODEs do not ingest observations, but they can be combined with standard recurrent units to form ODE-RNNs, where a NODE represents the dynamics and some RNN unit serves to update the hidden state when input is available at irregular intervals (Rubanova et al. 2019). Extensions of that idea include a GRU-like version of NODE, where the sporadic input of observations is interpreted as a Bayes update, termed ODE-GRU-Bayes (De Brouwer et al. 2019); using Controlled Differential Equations to obtain smooth temporal dynamics without jumps when the hidden state is updated with new input data (Kidger et al. 2020); and the Vid-ODE architecture which consists of a convolutional version of ODE-RNN as an encoder for tasks like video interpolation and extrapolation (Park et al. 2020).

3 Methodology

3.1 Recurrent Neural Networks

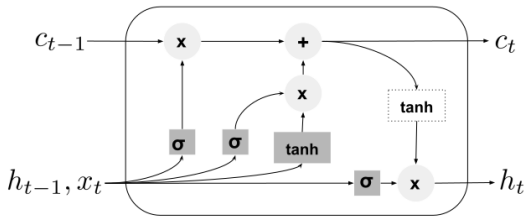
Recurrent neural networks have established themselves as a powerful tool for modelling sequential data. They have brought significant progress in a variety of applications, in recent years notably language processing and speech recognition (A. Graves et al. 2013; Nallapati et al. 2016; Sak et al. 2014). RNNs are feed-forward neural networks that iteratively update a hidden *state vector* $\boldsymbol{h}$ with the same computational unit (or "cell"). At a given time t, a new observation (input) is combined with the previous state $\boldsymbol{h}_{t-1}$ in a non-linear mapping that outputs the new state $\boldsymbol{h}_t$.

$$\boldsymbol{h}_t = f(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}, \boldsymbol{W}) \,, \tag{1}$$

with $\boldsymbol{W}$ being the trainable parameters of the cell. The input sequence has an overall length T, which can vary from sample to sample. The final hidden state $\boldsymbol{h}_T$ can be seen as a "cumulative encoding" that summarises the complete sequence, and serves as input for a decoder that maps it to the desired output, for instance a class label or a forecast of some biophysical variable. To fit the model to the training data, its parameters $\boldsymbol{W}$ are adjusted to minimise a suitable loss function that measures the prediction error of either the final states or all the intermediate states of the training sequences. Usually that minimisation uses some form of stochastic gradient descent (Linnainmaa 1976; Rumelhart et al. 1986). RNNs can in principle handle sequences of arbitrary and varying length; however, in their most basic form,

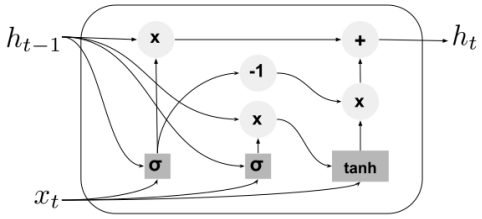
$$\boldsymbol{h}_t = \tanh(\boldsymbol{W}_x \boldsymbol{x}_t + \boldsymbol{W}_h \boldsymbol{h}_{t-1} + \boldsymbol{b}) \tag{2}$$

they struggle to capture long-range dependencies in the input, the error signal is diluted more and more with increasing distance (the "vanishing gradient" problem). To address this limitation, gated architectures have been proposed, most prominently Long Short-Term Memory (LSTM, Hochreiter et al. 1997) and Gated Recurrent Units (GRU, Chung et al. 2014). They use *gates* to store and retain information over longer time intervals, so as to mitigate the vanishing gradient problem, see Fig. 3. Gates are modulation functions that control the information flow into hidden state variables (note, LSTM has a second such variable, the "cell state"). Both LSTM and GRU are able to capture long- and short-range dependencies in the data (Alex Graves et al. 2013; Sutskever et al. 2014).



$$
\begin{aligned}
\boldsymbol{i}_t &= \sigma(\boldsymbol{W}_{xi}\boldsymbol{x}_t + \boldsymbol{W}_{hi}\boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \\
\boldsymbol{f}_t &= \sigma(\boldsymbol{W}_{xf}\boldsymbol{x}_t + \boldsymbol{W}_{hf}\boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \\
\boldsymbol{o}_t &= \sigma(\boldsymbol{W}_{xo}\boldsymbol{x}_t + \boldsymbol{W}_{ho}\boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \\
\boldsymbol{z}_t &= \tanh(\boldsymbol{W}_{xz}\boldsymbol{x}_t + \boldsymbol{W}_{hz}\boldsymbol{h}_{t-1} + \boldsymbol{b}_z) \\
\boldsymbol{c}_t &= \boldsymbol{f}_t \circ \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \circ \boldsymbol{z}_t \\
\boldsymbol{h}_t &= \boldsymbol{o}_t \circ \tanh(\boldsymbol{c}_t)
\end{aligned}
$$

(a) LSTM



$$
\begin{aligned}
\boldsymbol{f}_t &= \sigma(\boldsymbol{W}_{xf}\boldsymbol{x}_t + \boldsymbol{W}_{hf}\boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \\
\boldsymbol{r}_t &= \sigma(\boldsymbol{W}_{xr}\boldsymbol{x}_t + \boldsymbol{W}_{hr}\boldsymbol{h}_{t-1} + \boldsymbol{b}_r) \\
\boldsymbol{z}_t &= \tanh(\boldsymbol{W}_{xz}\boldsymbol{x}_t + \boldsymbol{W}_{hz}(\boldsymbol{r}_t \circ \boldsymbol{h}_{t-1}) + \boldsymbol{b}_z) \\
\boldsymbol{h}_t &= \boldsymbol{f}_t \circ \boldsymbol{h}_{t-1} + (1 - \boldsymbol{f}_t) \circ \boldsymbol{z}_t
\end{aligned}
$$

(b) GRU

Figure 3: RNN cell structures and dynamics of LSTM and GRU. σ is the Sigmoid function, $\circ$ denotes the Hadamard (element-wise) product, and $\boldsymbol{i}_t$, $\boldsymbol{f}_t$, $\boldsymbol{o}_t$ are respectively the *input* gate, *forget* gate, and *output* gate activations. $\boldsymbol{z}_t$ is the update. LSTM additionally has a separate cell state $\boldsymbol{c}_t$, whereas GRU has a *reset* gate $\boldsymbol{r}_t$ instead of the input and output gates.

3.2 Neural Ordinary Differential Equations

RNNs update their hidden state in discrete steps that are implicitly assumed to be evenly spaced, as the state change between two adjacent steps does not depend on their time difference. However, natural phenomena usually follow some underlying process that evolves continuously and longer time intervals correspond to larger changes. The hidden state of an RNN instead, cannot be sampled at arbitrary times, but only at every discrete update. Neural Ordinary Differential Equations (NODE) as proposed by Chen et al. 2018 can be seen as a continuous version of RNNs. Instead of parameterising the update of the hidden state as in equation 1, its continuous dynamics is parameterised by a neural network f_θ:

$$\frac{d\boldsymbol{h}(t)}{dt} = f_\theta(\boldsymbol{h}_t, t, \boldsymbol{W}) \tag{3}$$

where θ are the network parameters. This differential equation corresponds to an initial value problem with input $\boldsymbol{h}_0$ and solution $\boldsymbol{h}_T$, hence it can be used as a generic feed-forward building block. The NODE approach has several desirable properties:

- As it defines a continuous transformation of the latent function, it can be sampled at arbitrary, irregular times, which means that the model assumptions permit missing data points in the time series.
- The hidden state depends explicitly on a time parameter t, endowing the model with an awareness of time (respectively, rate of change), as opposed to only ordering (respectively, change per step).
- One can train the model with the adjoint method (Pontryagin 1962) instead of classical backpropagation, which is more memory efficient, since one need not store intermediate values during the forward pass.
- The formulation is not tied to a specific ODE solver, so one can trade off computational cost vs. numerical accuracy by choosing an appropriate solver.

3.3 Convolutional Layers

A common method to process image-based data is to use convolutional layers (LeCun et al. 1995). They are the core building blocks of various ground-breaking architectures for computer vision tasks (He et al. 2016; Krizhevsky et al. 2012; Simonyan et al. 2014; Szegedy et al. 2015). In contrast to fully-connected layers, convolutional layers are built on the concept of local connectivity. This means that the number of parameters is reduced considering only connections within the same spatial neighborhoods. The parameters of this transformation are stored in kxk matrices (called filters), where k is usually chosen as 3. Weight sharing further reduces the number of trainable weights by applying the same filter to all neighborhoods in the image. This procedure brings the advantage of translation invariance. In practice, a convolution layer consists of several filters applied to the same input. Moreover, convolutional neural networks (CNNs) are built by sequentially applying convolutional layers and non-linearity functions (activation functions) to achieve higher levels of abstraction.

3.4 ODE-RNN: Combining NODE and RNN

By itself, NODE updates the state vector $\boldsymbol{h}$ without taking into account any observations $\boldsymbol{x}$, i.e., it only represents the system dynamics. See equation 3. To inject observations whenever they become available, we incorporate conventional recurrent units (we test both LSTM or GRU), similar to Rubanova et al. 2019. The proposed approach alternates between two steps:

1. state prediction with NODE, and
2. state update with RNN, based on the observed data.

At time $t = 0$ the hidden state $\boldsymbol{h}$ is initialized with Gaussian noise of small magnitude, $\boldsymbol{h}_0 \sim \mathcal{N}(0, \sigma^2)$ with $\sigma = 10^{-4}$. Starting from there, the hidden state at time t_i is predicted with the ODE solver and then updated with an RNN only if an observation $\boldsymbol{x}_{t_i}$ is available for the time step t_i. If no observation is available, that second step is skipped. More formally, these two steps are expressed as follows:

$$\text{Prediction:} \quad \boldsymbol{h}_{t_i} := \mathsf{ODESolver}\big(f_\theta, \boldsymbol{h}_{t_{i-1}}, (t_{i-1}, t_i)\big) \tag{4}$$

$$\text{Update:} \quad \boldsymbol{h}_{t_i} := \mathsf{RNN}(\boldsymbol{h}_{t_i}, \boldsymbol{x}_{t_i}) \tag{5}$$

We use a multi-layer perceptron (MLP) with two hidden layers for the ODE network f_θ, which is solved with the Euler method. The Euler method performs numerical integration with an explicit first-order approximation and is considered the simplest explicit solver for initial value problems on ODEs. We also tried more sophisticated solvers (the 4^{th}-order Runge-Kutta and the adaptive Dopri5), but noticed neither an increase in performance nor a reduction of the variance as shown in Figure 4. The recurrent unit RNN in our experiments is a LSTM or GRU. Figure 5 schematically illustrates the evolution of the hidden state. The state $\boldsymbol{h}_T$ at the final time T, after seeing all observations, serves as a feature encoding of the complete time series and is fed into a fully-connected network to map it to class probabilities for the different crop types.

3.5 CONV-ODE-RNN: Combining CNN, NODE, and RNN

The architecture of CONV-ODE-RNN closely follows the one for ODE-RNN in Section 3.4. The main difference is the replacement of the fully-connected layers by convolutional layers. Note, that in fully-connected layers the input vector is multiplied by a fully-occupied matrix. When dense layers are replaced by the convolutional ones, the only change in terms of architecture is that the fully-occupied matrix is replaced by a sparse matrix,

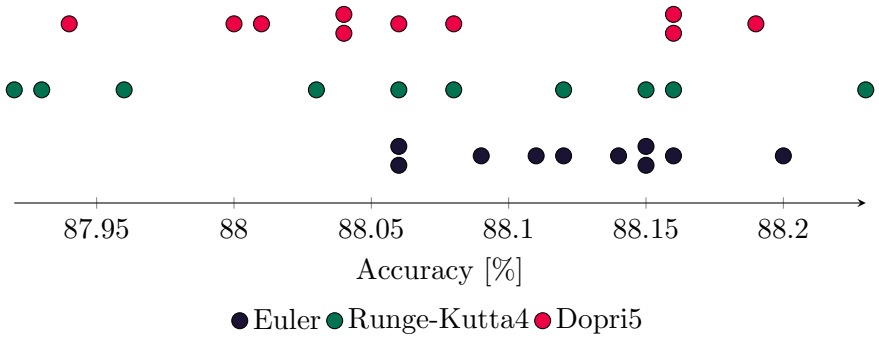


Figure 4: Validation accuracy on TUM dataset from 10 runs with different random seeds using the 1^{st}-order Euler, the 4^{th}-order Runge-Kutta, and the adaptive Dopri5 (Dormand et al. 1980) respectively as ODE solvers. Using the Euler method, the model achieved a mean score of 88.12% with a standard deviation of 0.04%. The Runge-Kutta method achieved a mean score of 88.06% with a standard deviation of 0.10%. The Dopri5 method achieved a mean score of 88.07% with a standard deviation of 0.08%.

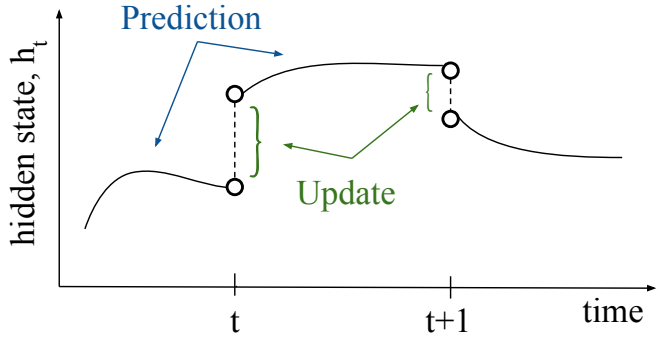


Figure 5: Illustration of the hidden state trajectory. Observations are available at $t_i = t$ and $t_i = t + 1$.

which is constructed by converting the filters into Toeplitz matrices (Toeplitz 1911). This replacement also includes the fully-connected layers inside the RNN cells (see Figure 3) and the classifier after the last hidden state. Although changing the ODE network f_θ from a fully-connected to a classical convolutional network is technically possible, filters larger than 1×1 seem unreasonable. Since ODEs can be seen as stacking for incremental updates, the receptive field grows tremendously fast by using 3×3 kernels. Instead, 1×1 kernels are used, which act as pixel-wise transformations of the hidden trajectories.

Many datasets come with pixels-wise occlusion masks. In this architecture, we use this information and use the output of the update step (Eq. 5) only for pixels that are not occluded. For the occluded pixels, the value of the hidden state before the update is just copied. In other words, we only update the hidden state when the pixel is observed.

4 Datasets

In this work, two different dataset are used: The TUM dataset and the SwissCrop dataset. The location of these are depicted in Figure 6.

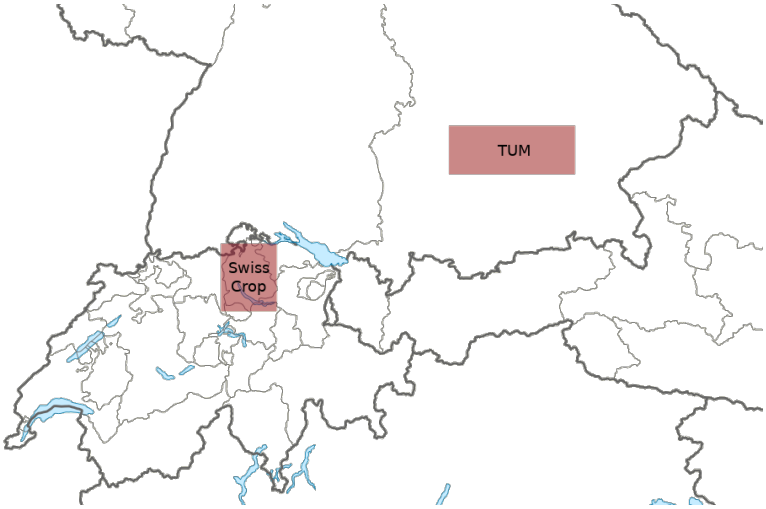


Figure 6: Location of the two datasets.

4.1 TU Munich Crop Data

The TUM dataset was generated by Rußwurm and Körner 2017. It comprises a time series of up to 26 multi-spectral Sentinel-2A satellite images with a ground resolution of 10 m. The data is collected over a $102 \times 42\,\mathrm{km}^2$ area north of Munich, Germany (Fig. 6), between December 2015 and August 2016, and comes with ground truth annotations with 19 classes (18 different crop types, plus a rejection class "other"). The data has been atmospherically corrected with the standard settings of the *Sen2cor* toolbox (Louis et al. 2016) and comprises six bands, namely B2 (blue), B3 (green) B4 (red) B8 (near infrared) B11 and B12 (both short-wave infrared). Lower-resolution bands are upsampled to 10 m with nearest-neighbour interpolation. The intensities in each band were normalised to have mean zero and unit energy. This dataset consists of patches of 3×3 pixels centred at the pixel of interest, and without further spatial information. We only use this data to for the non-spatial models (e.g. RNNs and ODE-RNNs) by flattening the neighbourhoods into 54-dimensional vectors.

4.2 SwissCrop Data

For the Swiss dataset, we have collected a time series of 71 multi-spectral Sentinel-2 satellite images with a ground resolution of 10 m collected over a 50×48 km^2 area over the Swiss cantons of Zurich and Thurgau (Fig. 6) between January 2019 and December 2019. As for TUM, the data were atmospherically corrected with the *Sen2cor* toolbox (Louis et al. 2016) and normalized to mean zero and unit energy per channel. We use 9 bands, B2 (blue), B3 (green), B4 (red), B5, B6, B7 (all vegetation red edge), B8 (near infrared), B11, B12 (both short-wave infrared). As above, lower-resolution bands are upsampled to 10 m with nearest-neighbor interpolation. For the non-spatial models (e.g. RNNs and ODE-RNNs), intensities from a 3×3 neighborhood around each pixel of interest are flattened into 81-dimensional input vectors. For the spatial models (e.g. CONV-ODE-RNNs), patches of 24×24 pixels are taken along with their corresponding 24×24 label patches. The crop labels are provided by the Swiss Federal Office of Agriculture. The original nomenclature includes a large number of crop classes with an extremely imbalanced, long-tailed distribution. We collect all rare classes with a total of $< 50'000$ pixels per class into a common rejection class "No Label", leaving us with 13 classes for the dominant crop types.

5 Experiments

5.1 Setup

We test two different variants of the proposed method, using either LSTM or GRU to include the observation data. These methods are then compared to the respective baselines without the NODE part. For quantitative performance evaluation we use the F1-score (harmonic mean between precision and recall) and the overall accuracy (fraction of correctly classified total pixels). As a main metric we focus on the F1-score, since it is less influenced by varying class frequencies, whereas the overall accuracy tends to be biased towards the most dominant classes. Uninformative observations (clouds, water) are detected with the corresponding tools of *Sen2cor* (Louis et al. 2016) and removed from the input time series.

5.2 Training Details

We implement the ODE network (f_θ in Eq. 3 and 4) as a multi-layer perceptron with two hidden layers. The number of neurons per layer is set to 255 for TUM, respectively to 220 for SwissCrop, reducing the capacity to account for the smaller number of classes. In the convolutional version, we use one hidden layer with 1×1-convolutions and dimension 1024. Otherwise, we use exclusively 3×3 filters for the rest of the convolutional architecture. The size of the hidden state vector $\boldsymbol{h}$ is set to 80 for both datasets. For the CONV-ODE-RNN, we use a hidden size of 70. A classifier with two hidden layers of 300 units each is used to map the final state $\boldsymbol{h}_T$ to class probabilities. For the CONV-ODE-RNN model, this classifier consists of convolutional layers. Batch normalization (Ioffe et al. 2015) of the hidden state is performed before every LSTM/GRU update and before the classification layer. The model is trained using the Adamax optimizer (Kingma et al. 2014) with mini-batch size 600 for TUM, respectively 500 for SwissCrop. A batch size of 6 (e.g. 6 patches of size 24×24) is used for the CONV-ODE-RNN model. The learning rate of the ODE-RNN models is initially set to 0.07 and then decreased by a constant factor 0.9995 after each batch iteration, corresponding to a half-life of 1400 batches. For the CONV-ODE-RNN, an initial learning rate of 0.002 is used and a decay factor such that the half-life corresponds to $20'000$ batches. On the TUM dataset, the model is trained for 12 epochs or a total of $\approx 6'000$ batch iterations. For the SwissCrop dataset, the models are trained for 50 epochs, except for the CONV-ODE-RNN model, which is trained for 80 epochs.

We compare our model to several baselines that do not employ NODE. The implementations of the RNN baselines (LSTM and GRU) without ODE backbone closely follow

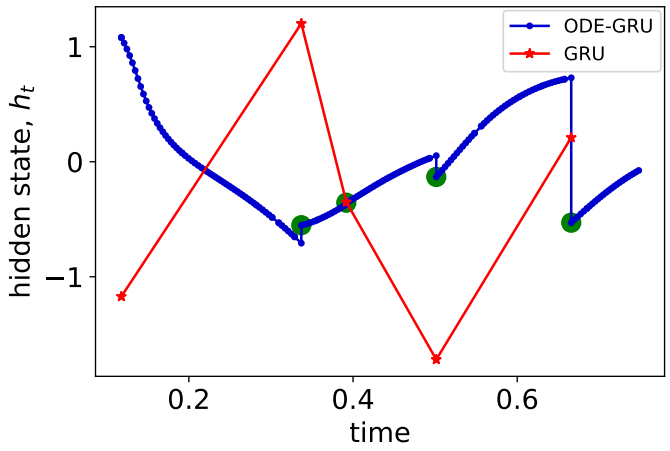


Figure 7: Examples of hidden state trajectories (reduced to 1 dimension with Principal Component Analysis). The blue and green markers denote prediction and update steps of ODE-GRU, respectively. Red markers denote GRU updates.

those of ODE-RNN, so as to rule out implementation differences and make them comparable. We run two different versions of the RNN baselines: *B-I* leaves out the ODE prediction step (equation 4), such that we are left with a conventional LSTM, respectively GRU model; *B-II* also leaves out the ODE prediction, but adds the time interval δt since the previous observation to the input vector. This trick gives the models the chance to learn how to handle varying delays between observations. Empirically this variant, denoted by the suffix $-\delta t$, does slightly improve the performance for both LSTM and GRU. A separate hyper-parameter search was done for each model. We found that, for both *B-I* and *B-II*, for GRU, a hidden state of size 150 worked best. For LSTM, the size of the cell state was also set to 150. The total number of learnable parameters per model are listed in Table 1. As optimiser for the baseline models we used Adam (Kingma et al. 2014) with batch size 300, which we empirically found to work best. As a final baseline for regularly sampled time series, without recurrence, we also run a temporal convolutional network (TCN), with hyper-parameter settings similar to those recommended by Pelletier et al. 2019. All models were implemented in Pytorch. Source code is available at `https://github.com/nandometzger/CONVODEcrop`.

	Method	#Parameters
B-II	LSTM-δt	346k
	GRU-δt	316k
NODE	ODE-LSTM	238k
	ODE-GRU	256k
CONV-NODE	CONV-ODE-LSTM	1758k
	CONV-ODE-GRU	1351k

Table 1: Number of trainable parameters per model (differences between *B-I* and *B-II*, respectively between TUM and SwissCrop, are <5k). To achieve the best performance, the NODE model needs significantly fewer parameters.

5.3 Results

In this section, the results of the conducted experiments are presented and discussed. In Section 5.3.1, the absolute performance of the ODE-RNN models is compared to the baseline model. The experiment of Section 5.3.2 examines the gain in performance when the spatial component is modeled as well (e.g. using CONV-ODE-RNN models). Furthermore, this chapter present experiments examine the usefulness of NODE-based models in special scenarios in terms of availability data. Namely: for the case of a smaller dataset size (see Section 5.3.3), the case of shorter time series, e.g. early season classification (see Section 5.3.4), and the case of sparser sampled time series (see Section 5.3.5).

5.3.1 Performance Comparison

Quantitative classification results for the TUM and SwissCrop datasets are reported in Tables 2 and 3, respectively. The proposed NODE method brings a consistent improvement over pure RNN models, across both performance metrics, although having fewer parameters (see Table 1). For all further experiments, we compare to GRU-δt, since it is the best-performing baseline. Note that all RNN baselines have been tuned for optimal performance on the specific task and are already very strong, e.g., they significantly outperform TCN (Pelletier et al. 2019), on both datasets. For a more detailed picture, we show the full confusion matrix of ODE-GRU for the TUM data in Figure 8. Note the consistently high accuracy across almost all crop classes. Significant miss-classifications happen only between the most ambiguous classes, namely: fallows (i.e., unproductive areas *not* currently used for crops) are confused with meadows and with the diffuse "other" class; and different types of winter grains with similar phenotype and similar phenological cycle are sometimes

confused. We also visualize the difference between the confusion matrix of ODE-GRU and that of the best baseline GRU-δt. For most classes, ODE-GRU increases the rate of correct classifications and reduces confusions, in some cases up to >7 percent points. There is only a single class out of 19 where ODE-GRU performs noticeably worse (peas). Furthermore, we provide a visualization of the predictions for the validation data of the SwissCrop in Figure 9.

	Method	**F1-score (%)**	**Accuracy (%)**
B-I	TCN (Pelletier et al. 2019)	69.0	85.7
	LSTM	77.0	87.1
	GRU	77.7	87.6
B-II	LSTM-δt	76.7	87.3
	GRU-δt	78.9	87.6
NODE	ODE-LSTM	78.7	87.8
	ODE-GRU	**79.6**	**88.1**

Table 2: Performance comparison on TUM dataset.

	Method	**F1-score (%)**	**Accuracy (%)**
B-I	TCN (Pelletier et al. 2019)	70.0	80.1
	LSTM	72.6	85.2
	GRU	74.6	85.5
B-II	LSTM-δt	73.7	85.2
	GRU-δt	75.5	85.6
NODE	ODE-LSTM	74.6	85.6
	ODE-GRU	**76.1**	**85.9**

Table 3: Performance comparison on SwissCrop dataset.

5.3.2 Convolution Model

In this experiment, we demonstrate the benefits of incorporating spatial information using CONV-ODE-RNN. We conduct this experiment using the SwissCrop dataset, as it contains the required georeferences of the pixels. We only take patches that have at maximum 10% unlabeled of the pixels to ensure that the spatial component of the data is present. Note, the class "Other" is not joined into the class "No Label". Instead, the class "Other" is treated as a separate target class and the class "No Label" is ignored by the model. Practically, this

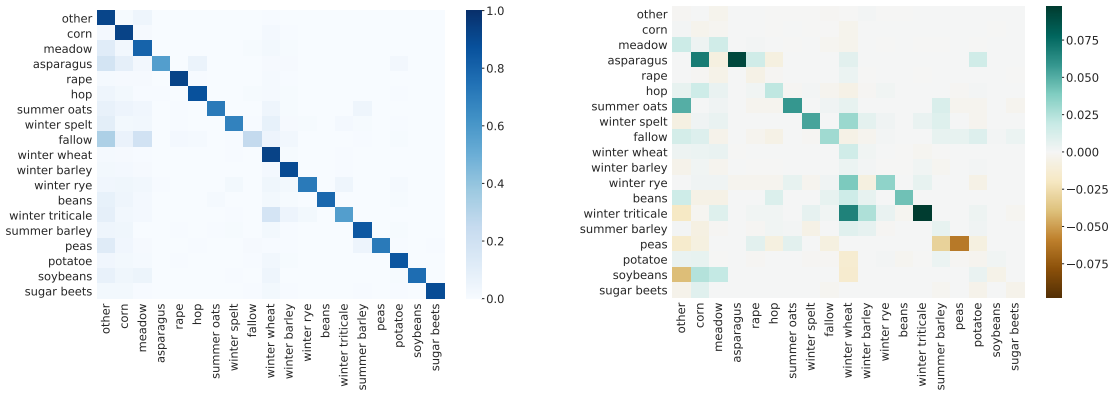


Figure 8: Per-class results for TUM data. **Left:** Normalised confusion matrix of ODE-GRU. Rows denote the true classes, columns the predicted ones. **Right:** Benefit of ODE-GRU over GRU-δt. Green denotes margins in favour of ODE-GRU (higher correctness on the diagonal, respectively lower confusion off the diagonal), brown denotes margins in favour of GRU-δt.

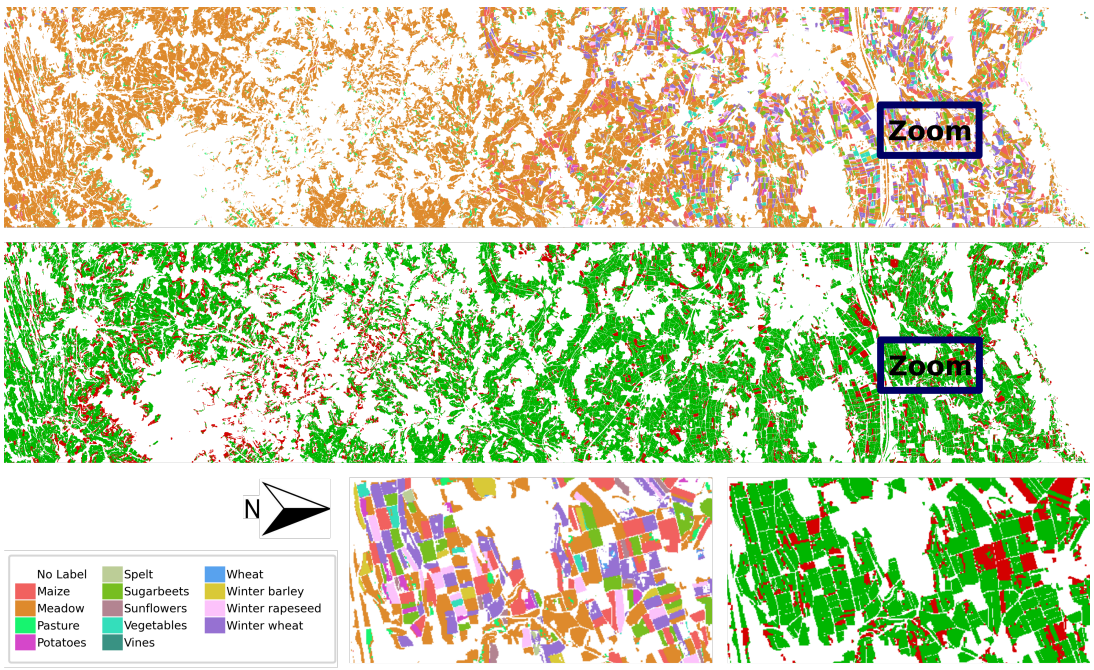


Figure 9: Crop map prediction by ODE-GRU for the complete validation area of Swisscrop, and corresponding accuracy map (red denotes wrongly classified pixels).

is achieved by assigning a weight of 0 to this class when calculating the loss. We assess the performance of the GRU-based models since the results of the experiments in Section 5.3.1 shows that GRU-based models generally perform better than the LSTM-based versions. Since the GRU-based models consistently perform better than the LSTM-based models. The quantitative results of Table 4 show a significant improvement of more than 5% to F1 Score by using the convolutional model instead of the non-spatial models. The outcome further shows that this improvement does not compromise accuracy.

	Method	**F1-score (%)**	**Accuracy (%)**
B-I	GRU	54.8	82.2
B-II	GRU-δt	55.1	**82.3**
NODE	ODE-GRU	55.2	82.2
CONV-NODE	CONV-ODE-GRU	**60.8**	**82.3**

Table 4: Performance comparison on the modified SwissCrop dataset with patches of at least 90% labeled pixels.

5.3.3 Dataset Size

The NODE model can be seen as a better prior on the temporal dynamics that accounts for continuity of the process and for an isotropic time scale. In order to show that this prior on the hidden state indeed makes sense and imposes a meaningful inductive bias, we amplify its influence by reducing the amount of training data. We note that this small-data regime is a relevant situation in agricultural remote sensing, where access to ground truth labels is often the bottleneck. We randomly sub-sample the training set to 10%, respectively 1% of its original size, train the models on the reduced set, and evaluate them on the same (full) test set as before. As expected, performance decreases for both ODE-GRU and the GRU-δt baseline if they are presented fewer training samples. However, the relative advantage due to NODE increases as the model sees less data and must rely more on the prior, as shown in Table 5. This supports our claim that the a-priori assumptions inherent in NODE better match the temporal evolution of agricultural crops, respectively their spectral responses.

5.3.4 Early Classification

The architecture as proposed in Section 3.4 is capable of extrapolating the hidden trajectory beyond the last data point in time, whereas conventional RNNs (including GRU) cannot extrapolate without observations – this is illustrated on the right end of the trajectories in Fig. 7. In the next experiment we investigate this difference, by comparing our ODE-GRU

	F1-score (%)			Accuracy (%)		
Method	100%	10%	1%	100%	10%	1%
GRU-δt	78.9	70.8	51.7	87.6	85.2	79.8
ODE-GRU	**79.6**	**72.0**	**53.0**	**88.1**	**85.8**	**80.9**
Difference	+0.8	+1.2	+1.3	+0.5	+0.6	+1.1

Table 5: Performance comparison for small-data regime (TUM dataset), using 100%, 10%, and 1% of the available training sequences.

to the GRU-δt baseline on the task of early classification. That task corresponds to the practically important scenario of forecasting the area of each crop from a shorter time series covering only the early part of the growing season, before most crops have been harvested. In the context of agriculture, forecasting is a practically important scenario, for instance to ensure food security or to inform policies for sustainable agriculture (Rußwurm, Tavenard, et al. 2019). To simulate that setting, we truncate the time series of the test set and keep only the leading 75%, respectively 50% of all time steps. For the conventional GRU, this means that we have to classify from only the early part of the growing season, whereas with ODE-GRU we can simply let the prediction step run beyond the last time step without observations. We also tried to extrapolate with the conventional GRU, by feeding it the global channel-wise mean values as input, but this did not improve the scores. Obviously classification performance drops if only a part of the seasonal cycle is shown to the model (Table 6). However, the relative improvement due to NODE increases quite significantly as the time series ends earlier. This demonstrates the value of explicitly modeling latent trajectories and being able to extrapolate them into the future beyond the last observation.

	F1-score (%)			Accuracy (%)		
Method	100%	75%	50%	100%	75%	50%
GRU-δt	78.9	49.0	20.2	87.6	74.9	48.7
ODE-GRU	**79.7**	**53.9**	**24.5**	**88.1**	**79.8**	**60.7**
Difference	+0.8	+4.9	+4.3	+0.5	+4.9	+12.0

Table 6: Performance comparison for early season classification (TUM dataset).

5.3.5 Missing Data Rate

In this experiment, the input image time series are also shortened, but this time by randomly subsampling along the time dimension rather than truncation. I.e., the model

receives fewer observations (100/75/50/25% of the original number) per time series, but those are still distributed over the entire phenological cycle, just sparser. Classification performance is tracked for three different scenarios: sparser sequences for training, but full density at test time; dense sequences for training, sparser ones at test time; and training and testing with the same amount of sparsity. Table 7 shows that moderately subsampling the training sequences does not greatly affect performance. In fact, for ODE-GRU we even observe a slight increase, which we attribute to a data augmentation effect: By randomly dropping a small portion of the images we increase the variability in the training set in terms of temporal gaps. Apparently that added diversity outweighs the slightly sparser sampling per sequence and helps to learn a better dynamical model. Naturally, the performance degrades for both models when downsampling only the test sequences, making them different from the sequences presented during training and thus creating a domain shift. Still, ODE-GRU degrades slower than the GRU-δt baseline, resulting in a larger relative edge. In the scenario where training and test data have progressively higher (but matched) sparsity, performance decreases monotonically, see diagonal in Table 7. In this setting ODE-GRU has an even (slightly) bigger edge as sparsity increases, showing that the stronger dynamical model of NODE manages to preserve a meaningful hidden state across longer temporal gaps. Across all the three scenarios, ODE-GRU not only consistently outperforms the baseline without NODE dynamics, but also achieves larger gains in the scenarios with fewer observations. The better handling of sparse time series with few input images is consistent with our hypothesis that NODE can represent the temporal evolution more faithfully and is therefore able to bridge longer and more irregular data gaps.

		F1 (%)				Accuracy (%)			
	Training Set	100%	75%	50%	25%	100%	75%	50%	25%
Testing Set	**GRU-δt**								
	100%	78.9	78.6	78.1	69.4	87.6	87.9	87.6	85.3
	75%	71.9	76.2	—	—	85.3	86.8	—	—
	50%	60.0	—	70.8	—	79.3	—	85.0	—
	25%	32.4	—	—	52.9	62.5	—	—	77.6
	ODE-GRU								
	100%	79.6	**80.2**	79.2	74.6	88.1	**88.5**	88.2	86.6
	75%	74.5	77.7	—	—	86.3	87.6	—	—
	50%	64.0	—	73.1	—	82.1	—	86.1	—
	25%	36.2	—	—	58.5	66.6	—	—	80.8
	Difference								
	100%	+0.7	+1.6	+1.1	+5.2	+0.5	+0.6	+0.6	+1.3
	75%	+2.6	+1.5	—	—	+1.0	+0.8	—	—
	50%	+4.0	—	+2.3	—	+2.8	—	+1.1	—
	25%	+3.8	—	—	+5.6	+4.1	—	—	+3.2

Table 7: Performance comparison for sparser time series (TUM dataset) with input observations subsampled to 100%, 75%, 50%, 25% of the original density.

6 Conclusion

We have studied the use of neural ordinary differential equations (NODE) in combination with conventional recurrent units to assimilate image observations, in order to model the latent dynamics of spectral signatures over time. Our target application is crop classification from optical satellite time series, with missing data due to cloud cover. We have shown that the ODE-RNN model performs consistently better than conventional recurrent network architectures that lack an explicit, isotropic notion of time. Moreover, we propose the CONV-ODE-RNN, which embeds the spatial characteristics of satellite images into the ODE-RNN model using convolutional layers. This further improves the F1 score by more than 5%. Under favorable conditions with frequent revisits and low to moderate cloud cover, conventional RNN models already perform quite well, yet we have observed consistent improvements. The NODE prior becomes particularly useful under challenging conditions, for instance when only little training data is available, when the available time series are sparse, and when the time series cover only a part of the growth cycle. In these situations NODE exhibits significant benefits in our experiments, reaching up to 10% higher F1-score for very sparse time series and up to 20% higher overall accuracy for the

early classification scenario. As a side effect, we also found that the ability to represent irregularly sampled time series and assimilate observations at arbitrary points in time allows for a convenient form of data augmentation by randomly dropping some observations from the training sequences.

So far, we have chosen the most straight-forward way to update the hidden dynamics of NODE when an observation becomes available. However, this procedure leads to discontinuous jumps of the hidden state with each update. We suspect that these jumps adversely affect the model because they violate the ODE's preference for a continuously evolving state vector. In future work, this limitation could be addressed through tighter integration between the prediction and update mechanisms or regularization of the update step.

Beyond our specific application, we believe that the NODE framework has great potential for time series analysis in remote sensing and Earth observation, where data gaps are ubiquitous, not only due to clouds but also caused by irregular acquisition patterns in space or time, sensor failures and exchanges, transmission limits, etc.

Bibliography

Bailly, Simon et al. (2018). "Crop-rotation structured classification using multi-source sentinel images and LPIS for crop type mapping". In: *IGARSS*.

Batista, Gustavo EAPA and Maria Carolina Monard (2002). "A Study of K-Nearest Neighbour as an Imputation Method". In: *Int'l Conference on Hybrid Intelligent Systems*.

Belgiu, Mariana and Ovidiu Csillik (2018). "Sentinel-2 cropland mapping using pixel-based and object-based time-weighted dynamic time warping analysis". In: *Remote Sensing of Environment* 204, pp. 509–523.

Brooks, Evan B et al. (2012). "Fitting the multitemporal curve: A Fourier series approach to the missing data problem in remote sensing analysis". In: *IEEE Transactions on Geoscience and Remote Sensing* 50.9, pp. 3340–3353.

Che, Zhengping et al. (2018). "Recurrent neural networks for multivariate time series with missing values". In: *Scientific Reports* 8.1, pp. 1–12.

Chen, Tian Qi et al. (2018). "Neural ordinary differential equations". In: *NeurIPS*.

Chung, Junyoung et al. (2014). "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *NeurIPS Workshops*.

Conrad, Christopher, Stefan Dech, et al. (2014). "Derivation of temporal windows for accurate crop discrimination in heterogeneous croplands of Uzbekistan using multitemporal RapidEye images". In: *Computers and Electronics in Agriculture* 103, pp. 63–74.

Conrad, Christopher, Sebastian Fritsch, et al. (2010). "Per-field irrigated crop classification in arid Central Asia using SPOT and ASTER data". In: *Remote Sensing* 2.4, pp. 1035–1056.

De Brouwer, Edward et al. (2019). "GRU-ODE-Bayes: Continuous modeling of sporadically-observed time series". In: *NeurIPS*.

Dormand, John R and Peter J Prince (1980). "A family of embedded Runge-Kutta formulae". In: *Journal of computational and applied mathematics* 6.1, pp. 19–26.

Foerster, Saskia et al. (2012). "Crop type mapping using spectral–temporal profiles and phenological information". In: *Computers and Electronics in Agriculture* 89, pp. 30–40.

Foley, Jonathan A et al. (2005). "Global consequences of land use". In: *Science* 309.5734, pp. 570–574.

Garnot, V Sainte Fare et al. (2019). "Time-Space tradeoff in deep learning models for crop classification on satellite multi-spectral image time series". In: *IGARSS*.

Graves, A., A. Mohamed, and G. Hinton (2013). "Speech recognition with deep recurrent neural networks". In: *ICASSP*.

Graves, Alex, Abdel-rahman Mohamed, and Geoffrey Hinton (2013). "Speech recognition with deep recurrent neural networks". In: *ICASSP*.

He, Kaiming et al. (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural Computation* 9.8, pp. 1735–1780.

Hu, Qiong et al. (2017). "How do temporal and spectral features matter in crop classification in Heilongjiang Province, China?" In: *Journal of Integrative Agriculture* 16, pp. 324–336.

Inglada, Jordi et al. (2015). "Assessment of an operational system for crop type map production using high temporal and spatial resolution satellite optical imagery". In: *Remote Sensing* 7.9, pp. 12356–12379.

Ioffe, Sergey and Christian Szegedy (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *ICML*.

Ji, Shunping et al. (2018). "3D convolutional neural networks for crop classification with multi-temporal remote sensing images". In: *Remote Sensing* 10.1, p. 75.

Kidger, Patrick et al. (2020). "Neural Controlled Differential Equations for Irregular Time Series". In: *arXiv preprint arXiv:2005.08926*.

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *ICLR*.

Koren, Yehuda, Robert Bell, and Chris Volinsky (2009). "Matrix factorization techniques for recommender systems". In: *IEEE Computer Journal* 42.8, pp. 30–37.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*, pp. 1097–1105.

Lea, Colin et al. (2016). "Temporal convolutional networks: A unified approach to action segmentation". In: *European Conference on Computer Vision*. Springer, pp. 47–54.

LeCun, Yann, Yoshua Bengio, et al. (1995). "Convolutional networks for images, speech, and time series". In: *The handbook of brain theory and neural networks* 3361.10, p. 1995.

Linnainmaa, Seppo (1976). "Taylor expansion of the accumulated rounding error". In: *BIT Numerical Mathematics* 16.2, pp. 146–160.

Louis, Jérôme et al. (2016). "Sentinel-2 Sen2Cor: L2A processor for users". In: *ESA Living Planet Symposium*.

M Rustowicz, Rose et al. (2019). "Semantic Segmentation of Crop Type in Africa: A Novel Dataset and Analysis of Deep Learning Methods". In: *CVPR Workshops*.

Mazumder, Rahul, Trevor Hastie, and Robert Tibshirani (2010). "Spectral regularization algorithms for learning large incomplete matrices". In: *Journal of Machine Learning Research* 11, pp. 2287–2322.

Nallapati, Ramesh et al. (2016). "Abstractive text summarization using sequence-to-sequence RNNs and beyond". In: *arXiv preprint arXiv:1602.06023*.

Nguyen, Trung H et al. (2018). "A comparison of imputation approaches for estimating forest biomass using Landsat time-series and inventory data". In: *Remote Sensing* 10.11, p. 1825.

Park, Sunghyun et al. (2020). "Vid-ODE: Continuous-Time Video Generation with Neural Ordinary Differential Equation". In: *arXiv preprint arXiv:2010.08188*.

Pelletier, Charlotte, Geoffrey I Webb, and François Petitjean (2019). "Temporal convolutional neural network for the classification of satellite image time series". In: *Remote Sensing* 11.5, p. 523.

Peña-Barragán, José M et al. (2011). "Object-based crop identification using multiple vegetation indices, textural features and crop phenology". In: *Remote Sensing of Environment* 115.6, pp. 1301–1316.

Pontryagin, Lev Semenovich (1962). *Mathematical theory of optimal processes*. Routledge.

Rubanova, Yulia, Ricky TQ Chen, and David Duvenaud (2019). "Latent ODEs for irregularly-sampled time series". In: *NeurIPS*.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). "Learning representations by back-propagating errors". In: *Nature* 323.6088, pp. 533–536.

Rußwurm, Marc and Marco Körner (2017). "Temporal vegetation modelling using long short-term memory networks for crop identification from medium-resolution multi-spectral satellite images". In: *CVPR Workshops*.

– (2018). "Multi-temporal land cover classification with sequential recurrent encoders". In: *ISPRS International Journal of Geo-Information* 7.4, p. 129.

Rußwurm, Marc, Sébastien Lefèvre, and Marco Körner (2019). "BreizhCrops: A Satellite Time Series Dataset for Crop Type Identification". In: *ICML Workshops*.

Rußwurm, Marc, Romain Tavenard, et al. (2019). "Early classification for agricultural monitoring from satellite time series". In: *International Conference on Machine Learning (ICML) Workshop*.

Sak, Haşim, Andrew Senior, and Françoise Beaufays (2014). "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition". In: *arXiv preprint arXiv:1402.1128*.

Siachalou, Sofia, Giorgos Mallinis, and Maria Tsakiri-Strati (2015). "A hidden Markov models approach for crop classification: Linking crop phenology to time series of multi-sensor remote sensing data". In: *Remote Sensing* 7.4, pp. 3633–3650.

Simonyan, Karen and Andrew Zisserman (2014). "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556*.

Stekhoven, Daniel J and Peter Bühlmann (2012). "MissForest – non-parametric missing value imputation for mixed-type data". In: *Bioinformatics* 28.1, pp. 112–118.

Sutskever, Ilya, Oriol Vinyals, and Quoc V Le (2014). "Sequence to sequence learning with neural networks". In: *NeurIPS*.

Szegedy, Christian et al. (2015). "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9.

Tan, Qingxiong et al. (2020). "DATA-GRU: Dual-Attention Time-Aware Gated Recurrent Unit for Irregular Multivariate Time Series". In: *AAAI*.

Toeplitz, Otto (1911). "Zur Theorie der quadratischen und bilinearen Formen von unendlichvielen Veränderlichen". In: *Mathematische Annalen* 70.3, pp. 351–376.

Tresp, Volker and Thomas Briegel (1998). "A solution for missing data in recurrent neural networks with an application to blood glucose prediction". In: *NeurIPS*.

Turkoglu, Mehmet Ozgur et al. (2019). "Gating Revisited: Deep Multi-layer RNNs That Can Be Trained". In: *arXiv preprint arXiv:1911.11033*.

Ustuner, M et al. (2014). "Crop type classification using vegetation indices of RapidEye imagery". In: *ISPRS Archives* 40.7.

Wardlow, Brian D and Stephen L Egbert (2008). "Large-area crop mapping using time-series MODIS 250m NDVI data: An assessment for the US Central Great Plains". In: *Remote Sensing of Environment* 112.3, pp. 1096–1116.

Zhang, Qiang et al. (2018). "Missing data reconstruction in remote sensing image with a unified spatial-emporal-spectral deep convolutional neural network". In: *IEEE Transactions on Geoscience and Remote Sensing* 56.8, pp. 4274–4288.

Declaration of Originality

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor .

Title of work (in block letters):

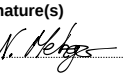
Crop Classification with Neural Ordinary Differential Equations

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Metzger	Nando

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work .

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
18.12.2020	

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.