remote sensing

MDPI

Article

A Comparison of Three Temporal Smoothing Algorithms to Improve Land Cover Classification: A Case Study from NEPAL

Nishanta Khanal [1,2,*], Mir Abdul Matin [1], Kabir Uddin [1], Ate Poortinga [2,3], Farrukh Chishtie [2,3], Karis Tenneson [2,3] and David Saah [2,3]

1 International Centre for Integrated Mountain Development, Kathmandu GPO Box 3226, Nepal; mir.matin@icimod.org (M.A.M.); Kabir.Uddin@icimod.org (K.U.)

2 Spatial Informatics Group, LLC, 2529 Yolanda Ct., Pleasanton, CA 94566, USA; apoortinga@sig-gis.com (A.P.); fchishtie@sig-gis.com (F.C.); ktenneson@sig-gis.com (K.T.); dsaah@sig-gis.com (D.S.)

3 SERVIR-Mekong, SM Tower, 24th Tower, 979/69 Paholyothin Road, Samsen Nai, Phayathai, Bangkok 10400, Thailand

* Correspondence: khanal.nishant@gmail.com or nkhanal@sig-gis.com

Received: 15 June 2020; Accepted: 14 August 2020; Published: 6 September 2020

Abstract: Time series land cover data statistics often fluctuate abruptly due to seasonal impact and other noise in the input image. Temporal smoothing techniques are used to reduce the noise in time series data used in land cover mapping. The effects of smoothing may vary based on the smoothing method and land cover category. In this study, we compared the performance of Fourier transformation smoothing, Whittaker smoother and Linear-Fit averaging smoother on Landsat 5, 7 and 8 based yearly composites to classify land cover in Province No. 1 of Nepal. The performance of each smoother was tested based on whether it was applied on image composites or on land cover primitives generated using the random forest machine learning method. The land cover data used in the study was from the years 2000 to 2018. Probability distribution was examined to check the quality of primitives and accuracy of the final land cover maps were accessed. The best results were found for the Whittaker smoothing for stable classes and Fourier smoothing for other classes. The results also show that classification using a properly selected smoothing algorithm outperforms a classification based on its unsmoothed data set. The final land cover generated by combining the best results obtained from different smoothing approaches increased our overall land cover map accuracy from 79.18% to 83.44%. This study shows that smoothing can result in a substantial increase in the quality of the results and that the smoothing approach should be carefully considered for each land cover class.

Keywords: remote sensing; temporal smoothing; google earth engine; machine learning; random forest; land cover

1. Introduction

Thematic land cover and forest maps are important sources of information for managers and policy makers for local and national-level policies and development strategies [1–4]. Important decisions regarding land use allocation, food production, hydrological modelling and natural resource management need to be supported by accurate information on spatial and temporal land cover dynamics which is occurring at a tremendous rate [3,5–7]. Remote sensing is a widely-accepted method for land cover classification. Since commercial high resolution images, free medium resolution images and open source cloud computing platforms are becoming more available, using machine learning (ML) techniques to classify land cover types with remotely-sensed data has gained popularity [8,9].

Dealing with noise is one of the main challenges in satellite image classification schemes. Satellite images can contain noise from different sources. Atmospheric effects while the image is taken, sensor degradation and thermal noise, and processing of signals received by sensors are some of the sources that can introduce noise to the images [10]. Noise might propagate through the classification scheme and negatively affect the performance of the model and quality of the outputs [10,11]. Pixel-based classifications in particular were found to be especially sensitive to noise as a considerable proportion of the signal of a pixel might come from the surrounding pixels [12,13]. Various techniques have been proposed to reduce the impact of noise. These methods include pre-processing of the input images or post-processing the results [12].

Even without these processes, random forest, among other machine learning methods, has been found to adapt and reduce the impacts of noise. This is because it uses bootstrapping and random split construction unlike other algorithms [14]. Random forest has been found to perform well in similar classification problems [15]. While convolution neural networks (CNN) has been gaining momentum in the remote sensing community, it is still a budding field and needs more research [16]. Another method to reduce noise and improve classification accuracy is the use of time-series remote sensing data [17,18]. However, one caveat to this technique is that image noise and sensor-introduced noises from individual scenes compound when information from multiple scenes are combined. This is because, if even one scene has noise it can impact the whole series. [19,20]. The issue of noise can also be remedied by smoothing the data either early in the process or before final assemblage of a land cover map so that the noise and inconsistencies are filtered out.

Smoothing is a method of identifying the underlying structure of data such as trends [21,22]. There are multiple studies that explore the use of smoothing algorithms on time-series data in order to reduce noise. For example, Lunetta et al. [20] explored the use of Fourier transformation on normalized difference vegetation index (NDVI) values; Chen et al. [23], Cao et al. [24] applied smoothing to vegetation index data by using a modified Savitzky–Golay filter; Jonsson and Eklundh [25,26] developed the TIMESAT software package, which can assimilate Savitzky–Golay, asymmetric Gaussian, and double-logistic algorithms to smooth NDVI time-series data from different sensors. However, these smoothing algorithms were found to be mathematically complex and computationally intensive. Therefore, various groups proposed methodologies that simplify the process. For example, Sakamoto et al. [27], Geerken et al. [28] explored the use of Fourier transforms for smoothing NDVI data, where the raw data is converted to harmonics and then frequencies corresponding to noise are removed. Eilers [29] introduced the Whittaker smoother, which uses a simple algorithm that runs quickly, yet still balances the fidelity of noisy data with the smoothness of the resulting curve [30]. The Whittaker smoother has also been recently adopted in Google Earth Engine (GEE) to smooth MODIS EVI time series data [31]. Khanal et al. also used the Whittaker smoother to map built-up areas of Kathmandu with improvements in results [32].

Techniques for smoothing remote sensing data have been widely-used; however, there are few studies that compare the smoothing algorithms themselves. Shao et al. [33] focused on smoothing MODIS-NDVI data to support land cover classification. However, since classifications using ML algorithms use more than just NDVI data, studies on how smoothing multiple input features and then training models with this data are needed. Similarly, Atkinson et al. [34] compared Fourier, Whittaker Smoother, Asymmetric Gaussian and Double Logistic smoothing approaches on phenology. Other studies have been done to test the performance of multiple smoothers on MODIS-LAI products [35] and hyper-spectral imagery [36] as well. Performance of some smoothers have also been tested in the context of land cover but with smoothers such as majority filter, Gaussian smoothing and bilateral filtering [37].

This study follows the Regional Land Cover Monitoring System (RLCMS) approach to perform land cover classification [38,39]. RLCMS is a joint USAID and NASA collaborative project that provides support to dedicated development and sustainable landscape projects [40]. This approach introduces the concept of primitives, which can be described as biophysical layers that represent information required to segregate land cover types [38,41,42], for example, tree canopy cover for forest/non-forest classification. The primitives are prepared from composites that are pre-processed with cloud/shadow masking, BRDF correction and terrain correction. ML techniques such as random forest are then used to generate primitives in the form of probability layers [38,40]. These primitives are then used in accordance with land cover typology in a decision tree that outputs required land cover map. While a land cover type can depend on multiple primitives, in this study we focus on definitions that allow single primitives per type as our focus is on studying the impact of smoothing rather than building an extensive land cover map.

Saah et al. [38] applied the Whittaker smoothing algorithm on primitive layers before assembling the final land cover map to stabilize the time-series and remove noise. However, they did not explore and evaluate the impact of including smoothing at a different stage of the image processing. Moreover, specific primitives or land cover classes may show better performance using a different smoothing algorithm. Therefore, this study explores the impact of temporal smoothing on image classification accuracy by following the RLCMS methodology and testing the approaches across all the classes in multiple stages of the workflow. This was done by applying three different smoothing algorithms on eight land cover classes. We compared the final accuracy of applying smoothing during pre-processing and post-processing. This study will help guide future work on land cover mapping on the appropriate choice for smoothing methodologies towards increasing the accuracy of the final results.

2. Materials and Methods

2.1. Study Area

Province No. 1, the easternmost province of Nepal, was selected as the study area. The province stretches from approximately 88.2217°E to 86.1358°E and 28.1310°N to 26.3298°N. This area was chosen because it has extensive topographical and land cover type variation within a short north–south distance: the south contains the flat plains of the Terai region, which progress to hills and snowy mountains in the north. The 2010 land cover map of Nepal (Figure 1) shows that the study region contains all land cover types presented by Uddin et al. [43].

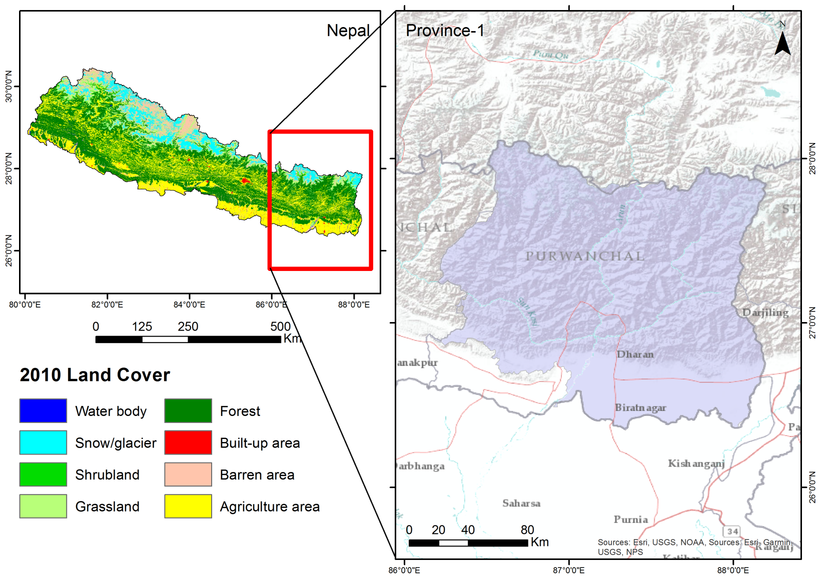


Figure 1. Province 1 (**right**) is located in the eastern part of Nepal (**left**). The different colours indicate the 2010 land cover classes.

2.2. Land Cover Typology

The land cover maps of Nepal prepared by Uddin et al. [43] were used to develop the land cover typology for this study. From the 12 classes presented in the Uddin et al. land cover map, we derived eight classes to further examine. For this study all forests were consolidated into a single class. Additionally, we created a new class "river bed" in order to obtain better segregation from classes such as built-up and barren land. When not separated, river beds have a tendency to be classified as either built-up or barren land with no real consistency. The complete typology used for this study is shown in Table 1.

Table 1. Typology of the land cover classes used for the study.

Class	Description
forest	Land covered with canopy vegetation and shrubs
cropland	Land used for cultivation of crops
settlement	Land containing artificial surfaces
wetland	Areas containing surface water, e.g., lakes and rivers
river bed	Sandy banks of water bodies especially rivers
grassland	Land covered with herbaceous plants that are not trees or shrubs
snow	Land covered in snow
bare	Land without vegetative cover and also not occupied by settlement

2.3. Reference Data Collection

The reference data were labelled using visual interpretation using Collect Earth [44,45] where the plot size was set to equal the spatial resolution of the 30 m Landsat images that were used in land cover mapping. Each plot had 49 points which were labelled according to observed land cover type. The plots were generated on a systematic grid with 2 km between sample plots and additional reference points were added with opportunistic sampling for underrepresented classes. The label land cover type for each plot was derived based on the majority type occurrence within the plot. The training points and validation points were randomly split from this reference data set with 86,093 points being used for training and 4299 for validation. The distributions of these points can be seen on Appendix A. The maximum number of points per class for validation was capped in order to avoid skewed accuracy because of a large number of points in a more accurate class. Because of persistent cloud covers, from 4307 validation data only 3677 points were used to perform accuracy assessment.

2.4. Land Cover Classification

The overall methodology of classification and comparison using various approaches is shown in Figure 2.

2.4.1. Optical Image Processing

Landsat 5, 7 and 8 surface reflectance images in GEE were used to prepare time series composites from the years 2000 to 2018. First, each scene was pre-processed by shadow masking, cloud masking, and then applying BRDF and topographic corrections [40]. The pre-processed scenes were used to create yearly composites. In order to prepare these composites, six bands from each scene were used: blue, green, red, nir, swir1 and swir2. The closest real pixel to the median, referred to as 'medoid', 20 percentile and 80 percentile of those bands were stored in a composite. Additionally, 20 and 80 percentile of NDVI, NDWI were also stored. In addition, information from auxiliary layers such as the JRC global water and SRTM DEM layers were added to the composite. The resulting layer, along with labelled reference data, was used to create primitives for each land cover type. The primitives were generated using a random forest classifier with 100 trees. Feature importance was also calculated to avoid overfitting. These primitives were then run through a decision tree classifier to obtain the final land cover maps.

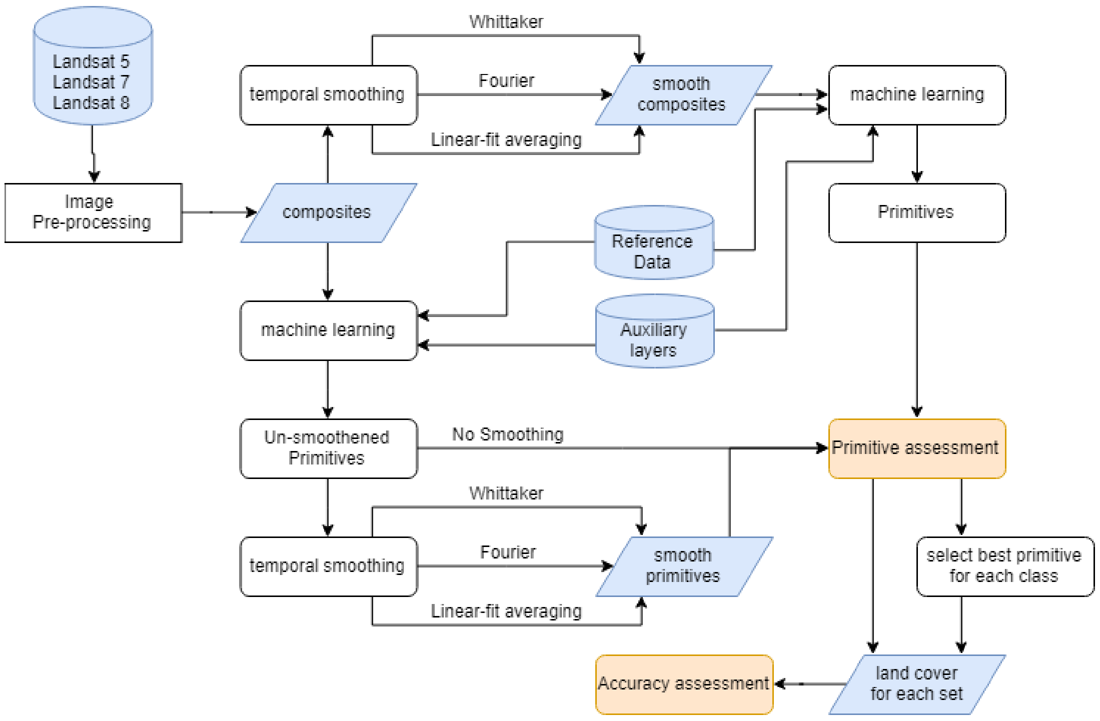


Figure 2. Overall methodology of the study.

2.4.2. Application of Smoothing

Three smoothing algorithms were tested using two different approaches, while everything else was kept constant. In the first approach, smoothing was applied during pre-processing on the composites where all spectral bands were smoothed temporally within our study period starting from 2000 and extending to 2018. These smoothed bands were then combined with the reference data to train a random forest classifier. This classifier was next used to create primitives for each class. This technique was repeated with each of the three smoothing algorithms. In our second approach, primitives for each class were first prepared using composites without smoothing and reference data. Then, each probability layer was smoothed temporally to obtain a smooth time series of primitives for each class. This technique was repeated for each of the three smoothing algorithms. The GEE codes for these implementations can be found in Appendices B.1 and B.2. This process resulted in 7 layers: three for each of the three smoothing algorithms applied on composite image, three for each of the three smoothing algorithms applied on primitives and one original data layer where smoothing was not applied.

The three algorithms that were used are detailed below:

Whittaker Smoothing

The Whittaker smoother is a popular smoothing approach that has been shown to work well with evenly-spaced data [29,46]. The Whittaker smoother can be considered as a special case of B-spline smoothing where the number of knots are equal to the data points [47]. Studies conducted by Hermance et al. [48,49] have successfully demonstrated its use in signal processing. The Whittaker smoother works by fitting a smooth series to a discrete data set. For a noisy y-series and its corresponding smooth z-series, the fidelity of data and roughness of z has to be balanced. Fidelity S is the sum of squares of differences and R is the roughness of the smooth data which is expressed as second order differences. With the goal of minimizing the balanced combination of the two, the combination Q can be expressed as

$$Q = S + \kappa R \tag{1}$$

where,

$$S = \sum_i (y_i - z_i)^2 \tag{2}$$

and

$$R = \sum_i ((z_i - z_{i-1}) - (z_{i-1} - z_{i-2}))^2 \tag{3}$$

κ controls the degree of smoothness where the larger the κ is, the smoother z is. This can be noted in matrix form as in Equation (4).

$$(I + \lambda D'_d D_d)z = y \tag{4}$$

where,

I = Identity matrix

λ = Smoothing degree, the larger λ is, the smoother z will be

D = differential matrix with m-2 rows and m columns with d as the order of differences Each row of the differential matrix D with the order of differences 2 contains the pattern 1 -2 1, shifted in a way where for each row i, $d_{i,i} = 1, d_{i,i+1} = -2, d_{i,i+2} = 1$ and all other values are 0. For example,

$$\begin{vmatrix} 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \end{vmatrix}$$

This matrix equation can be solved to compute the matrix z to obtain the smooth series.

Fourier Smoothing

Fourier smoothing is another popular smoothing approach based on the Fourier transformation of data. There are multiple scenarios where it has been used [25,33,34,50,51]. First, the raw data is converted to frequency domain by implementing a direct Fourier transformation (DFT). The DFT is defined by Equation (5) [52]

$$A_n = \sum_{k=0}^{N-1} X_k e^{(-2\pi ink/N)} \qquad n = 0, 1, ..., N-1 \tag{5}$$

where,

A_n = nth coefficient of the DFT

X_k = kth data in the times series

N = Number of data in the time series

i = $\sqrt{-1}$ It can also be written as (6),

$$A_n = \sum_{k=0}^{N-1} X_k W^{nk} \qquad n = 0, 1, ..., N-1 \tag{6}$$

where,

$$W = e^{(-2\pi i/N)} \tag{7}$$

However, since computations with the complex exponential are not straightforward on GEE yet, Equation (7) is re-expressed using Euler's formula as follows:

$$W = cos(-2\pi nk/N) + isin(-2\pi nk/N) \tag{8}$$

The real and imaginary parts were handled separately. From the results, the data located in indices greater than the specified smoothing degree were set as zero. After reducing the higher harmonics to zero, the resulting values are reverted back to the time domain by applying inverse DFT.

This essentially means that, first we fit the whole data set using a set of cyclic functions and then discarding the higher harmonics to remove the overfit so that we essentially are left with smoothened data set. The Fourier smoother along with Whittaker smoother was found to be two of the best smoothers in terms of classification in a study done by Shao et al. [33].

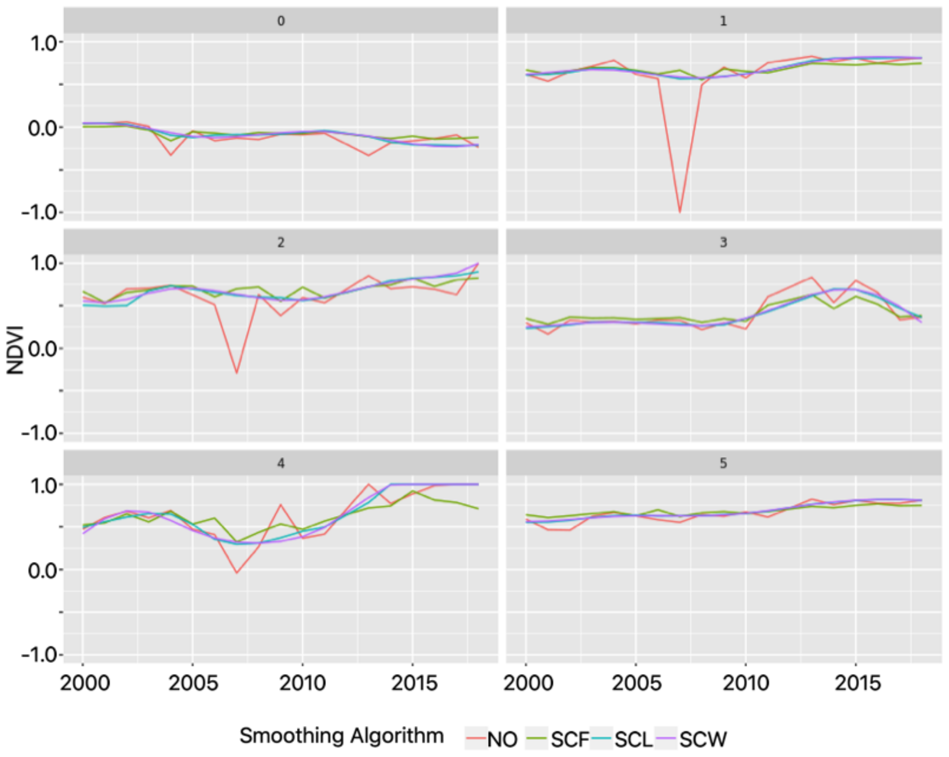
Linear Fit Smoothing

Linear Fit Smoothing is what we refer to as a simple derivative of moving window averaging where the window, a subset of the whole data set is used to average and obtain smooth values [53]. In this case, instead of taking the average over the moving window, data within a window is used to estimate a linear fit which is further averaged to get the final smoothened results.

The nature and smoothing effect of these algorithms can be explored through Figure 3a,b which were explored on different points exhibiting different types of time series data. From the plots it can be seen that, Fourier smoothing tends to preserve the shape of the time series more than others while Whittaker smoothing gave us the most flattened smoothing among the others. The value of smoothing degree of all of these algorithms was decided based on trial and error method of testing multiple values and deciding on the one that gave the best result. The parameters used for these smoothing algorithms are listed below in Table 2.

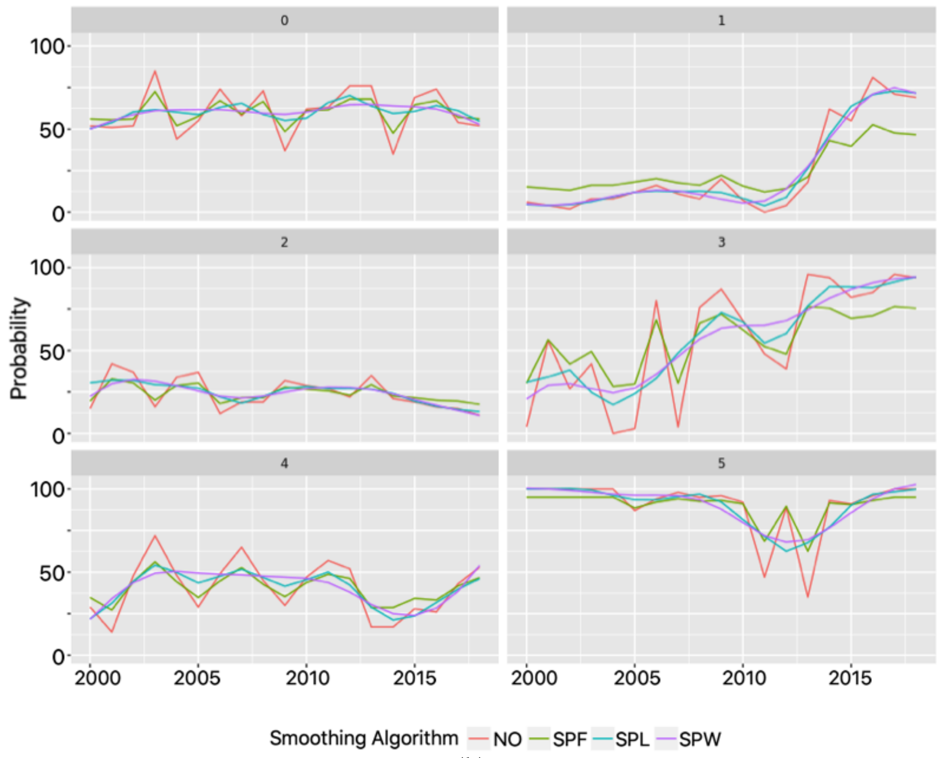
Table 2. Parameters used for smoothing algorithms.

Smoothing Algorithm	Parameters
Whittaker Smoothing	smoothing degree (λ) = 5 order of differences (d) = 3
Fourier Smoothing	smoothing degree (harmonics to preserve) = 2
Linear-Fit Smoothing	window size = 3



(**a**)

Figure 3. *Cont.*



(**b**)

Figure 3. Exploring the nature of smoothing in composites, (**a**) un-smoothed data vs. smoothing applied on composites. NO = No smoothing applied, SCF = Fourier smoothing applied on composite, SCL = Linear-fit smoothing applied on composite, SCW = Whittaker smoothing applied on composite, (**b**) un-smoothed data vs. smoothing applied on primitives. SPF = Fourier smoothing applied on primitive, SPL = Linear-fit smoothing applied on primitive, SPW = Whittaker smoothing applied on primitive.

2.4.3. Final Land Cover Map Generation and Accuracy Assessment

Primitives were assessed based on how well it could distinguish the feature it was supposed to represent vs the others. From the seven sets of primitives, the best primitive for each land cover type was picked based on their performance; this was used to create the eighth set of primitives. Different land cover maps were prepared in an assemblage logic using a decision tree classifier. For each set of primitives, the same assemblage logic was applied to produce land cover maps (see Appendices B.3 and C). The decision tree was kept constant so that only smoothing had an effect on the change in accuracy of the land cover maps. The accuracy of and change in accuracy between each land cover type was compared in order to understand the impact of different algorithms on different land cover types.

3. Results

The primitive assessment (Figure 4) shows the performance of the six different smoothing approaches on each of the eight different land cover types (snow, forest, cropland, wetland, riverbed, settlement, barren and grassland). A primitive is said to have better separability when the two box plots generated from it, one representing the probability distributions of its corresponding land cover type and another representing others, have either narrower plots or larger distance between its two plots. Narrower plots represent more consistent probability distribution while increased distance between the box plots means a clearer separation between the primitive's corresponding type and others. This results in reduced omission and commission when the proper threshold is applied for classification. Therefore, better separation is used to identify better primitives. We found that for snow and forest there is a clear separation between the class itself and other classes, followed by wetlands, settlement and riverbed. For grassland and cropland there is more overlap between the probability distributions.

Figure 4 also shows that Fourier smoothing results in the best separation for snow, wetlands, riverbed, barren land and grassland. For the other land cover types, Whittaker performed better than Fourier. Linear-fit was found to perform better than Fourier for forest, cropland and settlement but not as good as Whittaker. On the other hand, it performed better than Whittaker for other types but did not consistently outperform either Whittaker or Fourier. Furthermore, it was found that smoothing on the composite generally shows narrower distributions for snow, wetland and riverbed classes as compared to primitives. For forest, cropland and settlement the distributions are narrower when smoothing is applied on the primitives as opposed to on composites.

An example of the accuracy gained by smoothing versus visual assessment alone is shown in Figure 5. Here, we show that classification of forest without smoothing results in an inconsistent sudden drop in forest cover for the year 2014 (Figure 5b,f). The seen temporal dynamics of deforestation and regrowth would not be expected considering the short time frame [54]. Our results show a more consistent change after temporal smoothing in land cover types, which is more in line with the expected pathways.

The best performing smoothing algorithms were then used in the land cover assemblage. Figure 6 shows the accuracy of the different land cover classes before smoothing and after combining best primitives post-smoothing. It can be seen that snow and grassland classes show least improvements, while settlement, cropland and forest have larger improvements. It can also be noted that smoothing shows larger improvements for categories with little temporal variability. For example, snow, riverbeds and wetlands can be expected to have more temporal dynamics and show the least improvement.

The final accuracy assessment (Table 3) demonstrates how smoothing improved results. The overall accuracy increased from 79.18% to 83.44%. It is important to note that this result means that using smoothed data sets, given that the smoothing parameters are optimized for the input data set, is better than using no smoothing at all. However, to ensure that smoothing is not introducing errors, the nature of land cover type in question needs to be properly identified and the appropriate smoothing approach needs to be taken. Consequently, to optimize the accuracy of the final maps, our results showed that it is best to use different algorithms based on the land cover type and then combine the best primitives to prepare the final land cover map.

Table 3. Comparison of overall land cover time-series accuracy using different approaches of smoothing.

Smoothing Algorithm	Accuracy
No smoothing	0.7968
Fourier on primitive	0.8268
Whittaker on primitive	0.8297
Linear Fit on primitive	0.8270
Fourier on composite	0.8200
Whittaker on composite	0.8294
Linear Fit on composite	0.8217
Merged Using best Primitives	**0.8344**

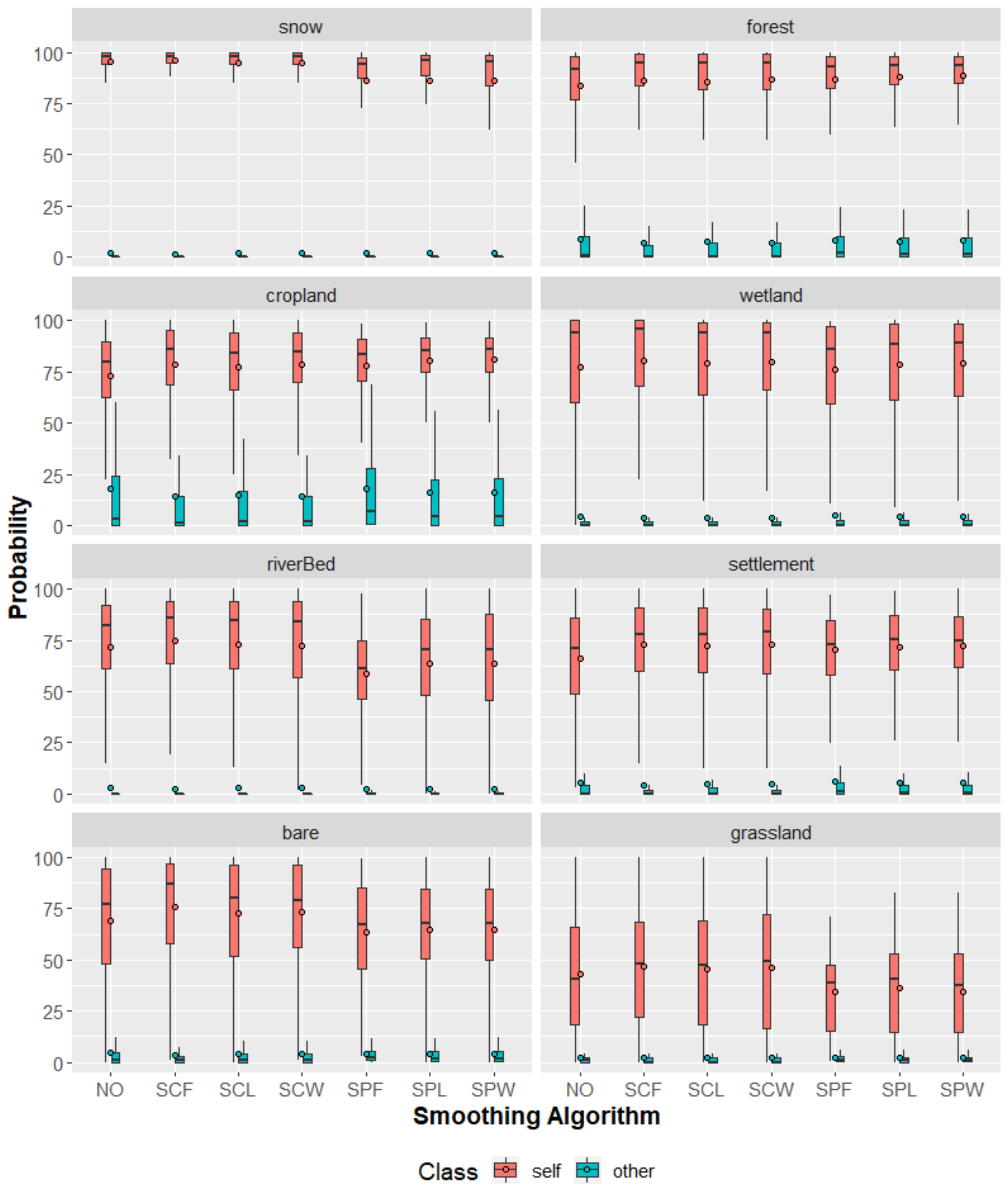


Figure 4. Assessment of primitives prepared using different algorithms. NO = No smoothing applied, SCF = Fourier smoothing applied on composite, SCL = Linear-fit smoothing applied on composite, SCW = Whittaker smoothing applied on composite, SPF = Fourier smoothing applied on primitive, SPL = Linear-fit smoothing applied on primitive, SPW = Whittaker smoothing applied on primitive. The distance between red and blue box plots in the vertical axis shows the inter-class separability of a primitive.

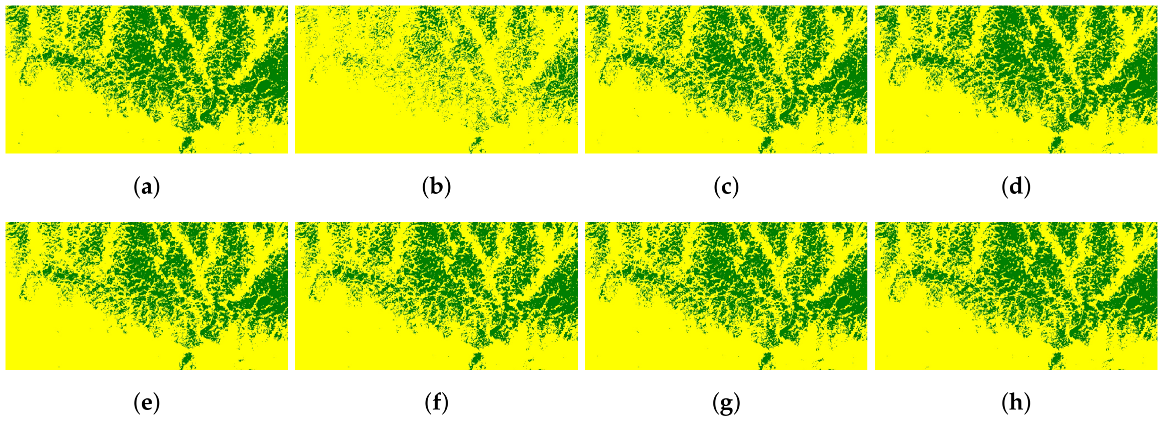


Figure 5. This figure shows forest (green) and other land cover (yellow) from 2013 to 2016 in the southern part of the study region. (**a**–**d**) shows un-smoothened results for 2013, 2014, 2015 and 2016 respectively. Similarly, (**e**–**h**) shows smoothened results for 2013, 2014, 2015 and 2016 respectively. A huge improvement can be seen in 2014. These are the final results generated from the best primitive set, i.e., picking best primitives after applying different algorithms on each primitive.

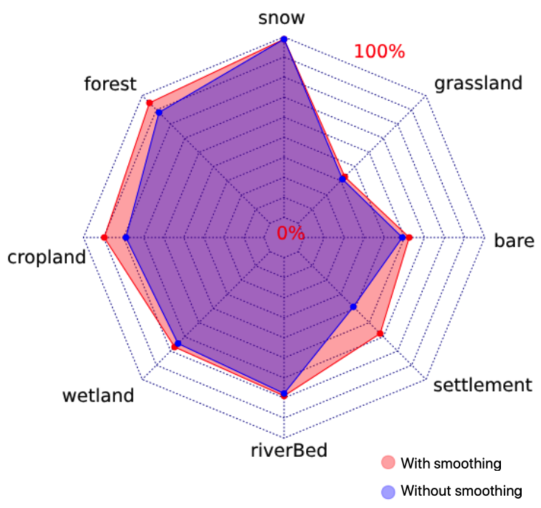


Figure 6. Comparison of class wise accuracy before applying any smoothing and after selecting the best primitives from various smoothing results. In the plot, accuracy extends from 0% (at centre) to 100% (at the outer edges).

4. Discussion

Three different smoothing algorithms were applied on eight different land cover classes. We found that improvements vary amongst different categories. We argue that this difference in performance is caused by the nature of the classes. In the case of Nepal, barren land often interchanges with grassland, and wetland (mainly rivers) often interchanges with riverbeds. Forest, cropland and settlement are more constant classes, which changes either once or infrequently. It therefore appears that Whittaker smoothing works best on land cover types with clear differentiation between them, whereas Fourier can be used on land cover types that more frequently changing. Fourier transformation showed good results for land cover types that frequently change as it breaks down a data series into a combination of cyclic equations; by doing this it preserves many of the highs and lows of the data while still removing noise. Fourier smoothing therefore works best when applied on the composite layer rather than the primitives, as the random forest method can then can use the noise-free composite.

In this study, we explored the use of smoothing on remote sensing composites and primitive layers. However, most studies (e.g., [20,23,24]) apply categorical classification on composites based on indices directly and do not have the intermediate primitive layers as used in this study. Moreover, the study was applied on a relatively small area with distinct climatological and topographic characteristics that determine the spatio-temporal land cover dynamics. More testing is needed to investigate if these findings also apply to other regions. It should also be noted that combining our smoothing algorithms with change detection algorithms such as LandTrendr [55] and Breaks For Additive Seasonal and Trend (BFAST) [56] might help improve land cover mapping accuracy as these algorithms are specifically designed to detect changes.

A main issue in generating land cover maps from annual composites is the number of valid observations (i.e., cloud-free pixels in a year). As such, the intra-annual distribution of pixels varies from year to year and as a consequence, the phenology is not necessarily fully represented in each annual composite. This might be captured in the ML model if multiple years are covered, but smoothing applies some additional noise reduction. This study shows that a smoothing algorithm should be selected based on the dynamics of a class. For example, Whittaker smoothing shows better results for stable classes. Fourier shows good results for classes that are dynamic when applied on composites. In general, it was found that smoothing on the primitives gives better performance than when applied on the composite layers for stable classes. Primitives are confidence layers generated by the ML algorithm so smoothing a set of primitives uses temporal information to improve estimates generated by the ML model rather than observation made by sensors. Stable classes show better performance after smoothing as it flattens out the inter annual dynamics in surface reflectance or class probability. For highly dynamic classes temporal segmentation algorithms might be another alternative approach to capture rapid changes.

We noticed an interesting phenomenon where changes in average probability were largely insignificant, but the lengths of error bars were significantly reduced. This means that smoothing does not necessarily increase the average confidence of overall classification but still improves the inter-class separability by reducing the deviation among pixels. The results line up with Shao et al. [33] while applying the Fourier and the Whittaker algorithms on an NDVI layer that was used for classification. Mingwei et al. [57] applied Fourier transformation on MODIS derived vegetation indices and found good correlation, which seems contradictory to our results. However, it needs to be noted that the study took a look at phenology, where there is a cyclic change in values, while we are looking at annual layers for which we do not expect to see much back and forth changes.

5. Conclusions

This study tested on using temporal smoothing as a means to improve land cover maps that were prepared by using a machine learning algorithm. Here the Fourier, Linear-fit and Whittaker smoothing algorithms were tested on image composites and on image primitives representing individual land cover types over time. Overall, we found that using temporal smoothing with machine learning improves the results compared to no smoothing; however, for classes that frequently undergo changes, Fourier smoothing produces the best results when applied on composite images. Conversely, for classes that do not undergo regular changes, Whittaker gave the best results when applied on primitives. Land cover types that do not undergo frequent changes generally were more improved with smoothing; in some cases, the accuracy of results decreased in classes that show frequent changes. The overall accuracy of the maps produced by our approach improved from 79.18% to 83.44% when we combined the best techniques for each land cover type. We therefore recommend that when creating land cover maps from satellite data, researchers identify the dynamics of the land cover type based on the target temporal resolution and then apply the appropriate algorithm. In cases where smoothing needs to be applied to the whole land cover rather than primitives from just one class, we recommend smoothing each class separately and then using the smoothed layers to prepare the final land cover map.

Author Contributions: Conceptualization, methodology, validation, and formal analysis, N.K., M.A.M. and K.U.; writing—original draft preparation N.K., M.A.M., and A.P.; writing—review and editing, N.K., M.A.M., A.P., K.U,

F.C. and K.T. Finally, all the authors read, edited, and critiqued the manuscript and approved the final version. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: We express our gratitude to David Molden, Director General and Eklabya Sharma, Deputy Director General of ICIMOD for the overall guidance. This study was partially supported by core funds of ICIMOD contributed by the governments of Afghanistan, Australia, Austria, Bangladesh, Bhutan, China, India, Myanmar, Nepal, Norway, Pakistan, Switzerland, and the United Kingdom. This paper has been prepared under the SERVIR-Hindu Kush Himalaya (SERVIR-HKH) initiative funded by the NASA and USAID. Our special gratitude goes to Ghulam Rasul, Regional Programme Manager, MENRIS and Birendra Bajracharya, Chief of Party, SERVIR-HKH for their encouragement and very active support in bringing out this work. We also thank the SERVIR RLCMS team for their assistance. Finally, we would like to thank those who reviewed the articles and provided valuable suggestions to improve the manuscript.

Conflicts of Interest: The authors declare no conflict of interest. The views and interpretations in this publication are those of the authors. They are not necessarily attributable to ICIMOD and do not imply the expression of any opinion by ICIMOD concerning the legal status of any country, territory, city, or area of its authority or concerning the delimitation of its frontiers or boundaries or the endorsement of any product.

Appendix A. Reference Data

Table A1. Training point distribution.

Land Class	Training Points	Validation Points
bare	1272	545
cropland	29,537	600
forest	45,806	600
grassland	378	162
riverBed	1615	600
settlement	1501	600
snow	3565	600
wetland	2419	600

Appendix B. Smoothing Codes

Appendix B.1. Smoothing Composites

```
DESCRIPTION

In order to test the impact of smoothing in landcover classification, we need to
first
check with smoothing the surface reflectance itself as well as smoothing the
intermediate smoothing results. In this step we will be smoothing composites
themself
using different algorithms namely whittaker, fourier and linear fit.

SCRIPT INFO

The users can/should change the following
1. imageCollection - source image collection that needs to be smoothened
2. boundary - the study region
3. smoothingAlgorithm - the algorithm to perform smoothing. can be 'whittaker',
'fourier'
or 'linearFit'
3. startYear - the start year from when to perform temporal smoothing
4. endYear - the end year to which to perform temporal smoothing
5. exportPath - the collection in which smoothened images are exported

DEVELOPER NOTES
1. Make sure the masks in each region is common to avoid errors


```

```
AUTHOR
Nishanta Khanal
ICIMOD, 2019
*/

var imageCollection =
    ee.ImageCollection('projects/servir-hkh/ncomp_yearly_30/compositesV2');
var boundary = ee.FeatureCollection('users/nkhanal/boundaries/p1_buffer_2000');
var smoothingAlgorithm = "fourier";
var startYear = 2000;
var endYear = 2018;
var exportPath = 'users/nkhanal/sm/four/comp';


var addCovariates =
    require("users/khanalnishant/ICIMOD:RLCMS/V2/Training/1_addCovariates_yearly")

var smoothing = require("users/khanalnishant/EETest:Algo/WhittakerSmoothing")
var param = 10;
if(smoothingAlgorithm == 'fourier'){
  smoothing = require("users/khanalnishant/EETest:Algo/FourierSmoothing");
  param = false;
}else if (smoothingAlgorithm == 'linearFit'){
  smoothing = require("users/khanalnishant/Algorithms:LinearFitSmoothing");
  param = 420;
}



// #####################################3
// # perform smoothing
// #####################################3
// #add image attribures for smoothing
function prepareImage(image){
    var imgYear = ee.Number.parse(image.id())
    image = image.unmask(0).clip(boundary).toShort()
    return image.set('year', imgYear,
                     'system:time_start', ee.Date.fromYMD(imgYear, 1, 1).millis());
}
var imageCollection = imageCollection.map(prepareImage)
print(imageCollection.first())

// # get original band names and add fitted at end to select fitted bands
var bandNames = ee.Image(imageCollection.first()).bandNames()
function fittedName(band){
    return ee.String(band).cat('_fitted')
}
var fittedNames = bandNames.map(fittedName)

var smoothingResults = smoothing.runModel(imageCollection, param)
var imageCollection = smoothingResults[0].select(fittedNames, bandNames)
var rmse = smoothingResults[1]

function exportHelper(image, assetID, imageCollectionID){
  image = image.int16()
  // # image = setExportProperties(image, args)

  Export.image.toAsset({
      image:ee.Image(image),
      description:smoothingAlgorithm+"-"+assetID +"-from-js",
      assetId:imageCollectionID+'/'+assetID,
      region:boundary,
      maxPixels:1e13,
      scale: 30
  });
}

for (var year=startYear;year<=endYear;year++){
    var image = ee.Image(imageCollection.filter(ee.Filter.eq('year',year)).first())
                  .clip(boundary)
    exportHelper(image, year, exportPath)
}

Export.image.toAsset({
    image:ee.Image(rmse),
    description:smoothingAlgorithm+"-rmse-from-js",
    assetId:exportPath+'/rmse',
    region:boundary,
    maxPixels:1e13,
    scale: 30
});
```

Appendix B.2. Smoothing Primitives

```
/*
DESCRIPTION

In order to test the impact of smoothing in landcover classification, we need to
first check with smoothing the surface reflectance itself as well as smoothing the
intermediate smoothing results. In this step we will be smoothing primitives using
different algorithms namely whittaker, fourier and linear fit. The export will
generate  an image with timeseries data and rmse on different bands. Different
script is later required to split them into images.

SCRIPT INFO

The users can/should change the following
1. imageCollection - source image collection that contains the unsmoothened
primitives
2. primitive - the primitive to be smoothened
3. boundary - the study region
4. smoothingAlgorithm - the algorithm to perform smoothing. can be 'whittaker',
'fourier' or 'linearFit'
5. startYear - the start year from when to perform temporal smoothing
6. endYear - the end year to which to perform temporal smoothing
7. exportPath - the collection in which smoothened images are exported

DEVELOPER NOTES
1. Make sure the masks in each region is common to avoid errors

AUTHOR
Nishanta Khanal
ICIMOD, 2019
*/

var imageCollectionId = 'users/nkhanal/sm/nosm/prims';
var primitive = 'wetland'
var boundary = ee.FeatureCollection('users/nkhanal/boundaries/p1_buffer_2000');
var smoothingAlgorithm = "linearFit";
var startYear = 2000;
var endYear = 2018;
var exportPath = 'users/nkhanal/sm/line/primtemp';

var addCovariates =
    require("users/khanalnishant/ICIMOD:RLCMS/V2/Training/1_addCovariates_yearly")
```

```

var smoothing = require("users/khanalnishant/EETest:Algo/WhittakerSmoothing")
var param = 10;
if(smoothingAlgorithm == 'fourier'){
  smoothing = require("users/khanalnishant/EETest:Algo/FourierSmoothing");
  param = false;
}else if (smoothingAlgorithm == 'linearFit'){
  smoothing = require("users/khanalnishant/Algorithms:LinearFitSmoothing");
  param = 420;
}


// ######################################3
// # perform smoothing
// ######################################3
// #add image attribures for smoothing
var yearlyPrims = [];
for (var year = startYear; year<=endYear; year++){
  var image = ee.Image(imageCollectionId+'/'+primitive+'-'+year).unmask(0)
            .clip(boundary);
  image = image.set('year', year,
                'system:time_start', ee.Date.fromYMD(year, 1, 1).millis())
  yearlyPrims.push(image);
}
imageCollection = ee.ImageCollection(yearlyPrims);
print(imageCollection.first())

// # get original band names and add fitted at end to select fitted bands
var bandNames = ee.Image(imageCollection.first()).bandNames()
function fittedName(band){
    return ee.String(band).cat('_fitted')
}
var fittedNames = bandNames.map(fittedName)

var smoothingResults = smoothing.runModel(imageCollection, param)
var imageCollection = smoothingResults[0].select(fittedNames, bandNames)
var rmse = smoothingResults[1]

function exportHelper(image, assetID, exportPath){
  image = image.int16()
  Export.image.toAsset({
      image:ee.Image(image),
      description:smoothingAlgorithm+"-"+assetID +"-from-js",
      assetId:exportPath+'/'+primitive+'-'+assetID,
      region:boundary,
      maxPixels:1e13,
      scale: 30
  });
}

var img = ee.Image([])
for (var year=startYear;year<=endYear;year++){
    var image = ee.Image(imageCollection.filter(ee.Filter.eq('year',year)).first())
                  .clip(boundary)
    img = img.addBands(image.rename('b'+year));
}
var img = img.addBands(rmse.rename('rmse')).toInt16();

Export.image.toAsset({
    image:img,
    description:smoothingAlgorithm+"-"+primitive+"-from-js",
    assetId:exportPath+'/'+primitive+'Results',
    region:boundary,
    maxPixels:1e13,
    scale: 30
});
```

Appendix B.3. Assemblage

```
/*
DESCRIPTION

Now that we have satisfactory primitives we will run them through assembler
to obtain a land cover map. The assembler prepares a decision tree based on
user specified thresholds which can be tuned based on visual assessment as
well as primitive assessment plots. The order of primitives in the list denotes
the order in which primitives are placed in the decision tree with the first
primitive placed on the top and so forth. This basically means that if a pixel
has high probability on two primitives (according to the specified threshold)
the final class will be based on the primitive that is higher up on the decision
tree.

SCRIPT INFOS

The users can/should change the following parameters.
1. boundary - this should contain the boundary of region of interest
2. repository - the repository that contains the stack of primitives required
3. scale - the spatial resolution of the resulting landCover
4. exportPath - the path to which to export the resulting landcover
5. primitives - the list of primitives to include in the assembler
6. defaultThresholds - (optional) the default thresholds that the interface
    initiates with.It can be changed in the UI later. This should be in line
    with the list of primitives.
7. year - (optional) the default year to load into the interface

The users need to do the following
1. After the script is run, initiate the export task by going to the "Tasks" tab
------->>>>>>>

*/

/*
Variables - refer above notes section for description on these
*/
var boundary = ee.FeatureCollection("users/nkhanal/boundaries/p1_buffer_2000")
                 .geometry();
var repository = 'users/nkhanal/sm/final/prims/';
var scale = 30;
var exportPath = 'users/nkhanal/sm/final/lc';
var primitives = ['snow','forest','cropland','wetland','riverBed','settlement',
'bare'];
var defaultThresholds = [70, 60, 60, 65, 60, 60, 70];
var year = 2017;

/* ------------------------------------------------------------------------------
WARNING!!!

FUNCTIONS AND PROCEDURES BELOW HERE

edit at your own risk
*/

// container for the classified landclass image
var landClass = ee.Image();

```

```
// **************************************************
// ui objects
// containers for sliders
var sliders = {};
// containers for inspected values
var valLabels = {};
// list of all selectable years
var availableYears = ['2000','2001','2002','2003','2004','2005',
          '2006','2007','2008','2009','2010','2011',
  '2012','2013','2014','2015','2016','2017','2018'];
// dropdown to select years
var yearList = ui.Select(availableYears, 'year', ''+year);
// checkbox to visualize primitives using thresholds
var primThresholds = ui.Checkbox({
  label:'Visualize primitive by thresholds',
  style:{'height':'18px','fontSize':'11px', 'padding':'4px', 'margin':'0px'}
});
// button to refresh display according to parameters
var refreshDisplay = ui.Button({
  label:'Refresh Display',
  onClick: refresh,
  style: {'fontSize':'11px', 'padding':'4px', 'margin':'0px'}
});
// button to export the current LandCover
var exportLC = ui.Button({
  label:'Export current landCover',
  onClick: exportHelper,
  style: {'fontSize':'11px', 'padding':'4px', 'margin':'0px'}
});
// button to export the current LandCover
var exportAllLC = ui.Button({
  label:'Export landcover stack',
  onClick: exportAllHelper,
  style: {'fontSize':'11px', 'padding':'4px', 'margin':'0px'}
});
// button to export problem regions
var exportPA = ui.Button({
  label:'Export Problem Regions',
  onClick: exportAreas,
  style: {'fontSize':'11px', 'padding':'4px', 'margin':'0px'}
});

// **************************************************
// functions

// function to reclassify stack image using decision tree
function reclassify(stackImage, decisionTree){
  var classifier = ee.Classifier.decisionTree(decisionTree);
  var landClass = stackImage.classify(classifier);
  return landClass;
}

// client side list map function to change list of primitives
// to a list of images corresponding to the primitives
// also adds the primitives to the map
function getPrimitiveImages(primitive){
  var image = ee.Image(repository+primitive).select('b'+year)
                                            .multiply(0.01)
                                            .rename(primitive);
  var primLayer = ui.Map.Layer(image, {max:100}, primitive, false, 1);
  if (primThresholds.getValue()){
    primLayer = ui.Map.Layer(image.gte(sliders[primitive].getValue()),
                             {},
                              primitive,
                              false,
                              1);
  }
  Map.layers().set(primitives.indexOf(primitive), primLayer);
  return image;
}

// function to build decision tree from interface
function buildDecisionTree(){
  var values = {
    forest:sliders['forest'].getValue(),
    wetland:sliders['wetland'].getValue(),
    settlement:sliders['settlement'].getValue(),
    cropland:sliders['cropland'].getValue(),
    snow:sliders['snow'].getValue(),
    riverBed:sliders['riverBed'].getValue(),
    bare:sliders['bare'].getValue()
  }

  var DT = ['1) root 9999 9999 9999']
  var base = 1;
  for (var i = 0; i < primitives.length; i++){
    var a = base * 2
    var b = base * 2 + 1
    DT.push(''+a+') '+
            primitives[i]+'>='+values[primitives[i]]+
            ' 9999 9999 '+(i+1)+' *');
    if(i == primitives.length-1){
      DT.push(''+b+') '+primitives[i]+'<'+values[primitives[i]]+' 9999 9999 8 *');
    }else DT.push(''+b+') '+primitives[i]+'<'+values[primitives[i]]+' 9999 9999 9999');
    base = b
  }
  return DT.join('\n')
}

// function to start the process
function process(year){
  var stackImage = ee.Image(primitives.map(getPrimitiveImages)).clip(boundary);
  landClass = reclassify(stackImage, buildDecisionTree());
  var lcLayer = ui.Map.Layer(landClass, imageVisParam, 'land cover', true, 1);
  Map.layers().set(primitives.length, lcLayer);
}

// function to add legend to the map
function addLegend(){
  var UTILS = require('users/khanalnishant/Algorithms:Utilities')
  var palette = imageVisParam.palette
  palette = palette
  var labels = primitives.concat('grassland')
UTILS.addLegend(palette, labels,'Legend',{
    'position':'bottom-right',
    'padding':'8px 16px',
  });
}

// function to add inspect panel to the map
function addInspect(){
  var panel = ui.Panel([],
                        ui.Panel.Layout.Flow('vertical'),
                        {maxHeight:'200px',position:'bottom-center'});
  var inspectLabel = ui.Label('Click to inspect primitive values',
                               {'height':'18px',
                                'fontSize':'11px',
                                'padding':'0px',
                                'margin':'1px'});
  panel.add(inspectLabel)
  for (var i = 0; i <primitives.length;i++){
    var insLabel =  ui.Label(primitives[i],{'width':'60px',
                                             'fontSize':'11px',
                                             'padding':'0px',
                                             'margin':'1px'});
    valLabels[primitives[i]] =  ui.Label('Click Map',{'fontSize':'11px',
                                                       'padding':'0px',
                                                       'margin':'1px'});
    var inspectSubPanel = ui.Panel([insLabel, valLabels[primitives[i]]],
                                   ui.Panel.Layout.Flow('horizontal'));
    panel.add(inspectSubPanel);
  }
  Map.add(panel)
}

// function to update the inspect section once map is clicked
function fetchValues(lonlat){
  var point = ee.Geometry.Point(lonlat.lon, lonlat.lat);
  var stack = ee.Image([]);
  for (var i=0; i<primitives.length;i++){
    var image = Map.layers().get(i).get('eeObject');
    stack = stack.addBands(image);
    valLabels[primitives[i]].setValue('updating values');
  }
  var pointSampled = stack.sample(point,30).first().evaluate(updateValues);
}

//update the labels once the sampling is done
function updateValues(feature){
  // print(feature.properties);
  for (var i=0; i<primitives.length;i++){
    var value = parseFloat(feature.properties[primitives[i]]).toFixed(2);
    valLabels[primitives[i]].setValue(value);
  }
}

// function to export all Land Covers
function exportHelper(){
  Export.image.toAsset({
    image:landClass.toInt8(),
    description:'LandCover-'+year,
    region:boundary,
    scale:scale,
    assetId:exportPath+'/'+year,
    maxPixels:1e10
  });
}

// function to export current Land Cover
function exportAllHelper(){
  print("initiating export");
  var tempYr = year;
  var decisionTree = buildDecisionTree();
  var image = ee.Image([]);
  for (var i = 0; i<availableYears.length;i++){
    year = availableYears[i];
    var stackImage = ee.Image(primitives.map(getPrimitiveImages)).clip(boundary);
    var landClass = reclassify(stackImage, decisionTree)
    image = image.addBands(landClass.rename("b"+year));
  }
  image = image.set('decisionTree',
                    decisionTree,
                    'year',
                    ee.Date.fromYMD(parseInt(year),
                    1,
                    1));
  Export.image.toAsset({
    image:image.toInt8(),
    description:'LandCover-stack',
    region:boundary,
    scale:30,
    assetId:exportPath+'/lcstack',
    maxPixels:1e10
  });
  year = tempYr;
}

// function to export Problem Areas
function exportAreas(){
  var generatedPoints = ee.FeatureCollection.randomPoints(geometry, 100);
  // add latitude and longitude as attributes
  generatedPoints = generatedPoints.map(function(feature){
    var coords = feature.geometry().coordinates();
      return feature.set('latitude',coords.get(1),'longitude', coords.get(0))
  });
  Export.table.toDrive({
    collection:generatedPoints,
    description: 'RLCMS-additionalPoints',
    fileFormat: 'CSV'
  })
}
// function to refresh display based on parameters
function refresh(){
  // get layer shown state
  var layers = Map.layers();
  var shownStat = layers.map(function(layer){
    return layer.getShown();
  });
  // get selected year
  year = yearList.getValue();
  // initiate the process
  process(year);
  // reassign the previous layer shown status
  layers = Map.layers();
  layers.map(function(layer){
   return layer.setShown(shownStat[layers.indexOf(layer)]);
  });
}

// function to initialize the application
function init(){
  var panel = ui.Panel([], ui.Panel.Layout.Flow('vertical'),
  {position:'bottom-left'});
  var yearLabel = ui.Label('Year',{'width':'30px',
                                   'height':'18px',
                                   'fontSize':'11px',
                                   'margin':'15px 0px'});
  var yearSubPanel = ui.Panel([yearLabel, yearList],
                              ui.Panel.Layout.Flow('horizontal'));
  panel.add(yearSubPanel);
  var sliderLabel = ui.Label('Adjust Probability Thresholds',
                             {'height':'18px',
                              'fontSize':'11px',
                              'padding':'0px',
                              'margin':'1px'});
  panel.add(sliderLabel);
  for (var i = 0; i< primitives.length;i++){
    var label = ui.Label(primitives[i],{'width':'50px',
                                        'height':'18px',
                                        'fontSize':'11px',
                                        'padding':'0px',
                                        'margin':'1px'});
    sliders[primitives[i]] = ui.Slider({
      min:0,
      max:100,
      value:defaultThresholds[i],
      style:{'height':'18px','fontSize':'11px', 'padding':'0px', 'margin':'1px'}
    });
    var subPanel = ui.Panel([label, sliders[primitives[i]]],
                            ui.Panel.Layout.Flow('horizontal'),
                            {'padding':'4px'});
    panel.add(subPanel);
  }
  panel.add(primThresholds);
  panel.add(refreshDisplay);
  panel.add(exportLC);
  panel.add(exportAllLC);
  panel.add(exportPA);
  Map.add(panel);
  addLegend();
  addInspect();
  Map.onClick(fetchValues);
}

init();
process(year);
```

Appendix C. Datasets

Here are the list of datasets that were used and/or prepared for this study.

- Non-smoothened primitives : users/nkhanal/sm/nosm/prims
- Non-smoothened land cover : users/nkhanal/sm/nosm/lc
- Final smoothened and best selected primitives: users/nkhanal/sm/final/prims
- Final smoothened and best selected land cover: users/nkhanal/sm/final/lc

These addresses refer to the GEE asset ID of the dataset in question.

References

1. Running, S.W. Ecosystem disturbance, carbon, and climate. *Science* **2008**, *321*, 652–653. [CrossRef] [PubMed]
2. Waldner, F.; Fritz, S.; Di Gregorio, A.; Defourny, P. Mapping priorities to focus cropland mapping activities: Fitness assessment of existing global, regional and national cropland maps. *Remote Sens.* **2015**, *7*, 7959–7986. [CrossRef]
3. Boisvenue, C.; Smiley, B.P.; White, J.C.; Kurz, W.A.; Wulder, M.A. Improving carbon monitoring and reporting in forests using spatially-explicit information. *Carbon Balance Manag.* **2016**, *11*, 23. [CrossRef] [PubMed]
4. Pettorelli, N.; Wegmann, M.; Skidmore, A.; Mücher, S.; Dawson, T.P.; Fernandez, M.; Lucas, R.; Schaepman, M.E.; Wang, T.; O'Connor, B.; et al. Framing the concept of satellite remote sensing essential biodiversity variables: Challenges and future directions. *Remote Sens. Ecol. Conserv.* **2016**, *2*, 122–131. [CrossRef]

5. Turner, B.L.; Lambin, E.F.; Reenberg, A. The emergence of land change science for global environmental change and sustainability. *Proc. Natl. Acad. Sci. USA* **2007**, *104*, 20666–20671. [CrossRef]
6. Bui, Y.T.; Orange, D.; Visser, S.; Hoanh, C.T.; Laissus, M.; Poortinga, A.; Tran, D.T.; Stroosnijder, L. Lumped surface and sub-surface runoff for erosion modeling within a small hilly watershed in northern Vietnam. *Hydrol. Process.* **2014**, *28*, 2961–2974. [CrossRef]
7. Otukei, J.R.; Blaschke, T. Land cover change assessment using decision trees, support vector machines and maximum likelihood classification algorithms. *Int. J. Appl. Earth Obs. Geoinf.* **2010**, *12*, S27–S31. [CrossRef]
8. Tuia, D.; Persello, C.; Bruzzone, L. Domain adaptation for the classification of remote sensing data: An overview of recent advances. *IEEE Geosci. Remote Sens. Mag.* **2016**, *4*, 41–57. [CrossRef]
9. Phiri, D.; Morgenroth, J. Developments in Landsat land cover classification methods: A review. *Remote Sens.* **2017**, *9*, 967. [CrossRef]
10. Landgrebe, D.A.; Malaret, E. Noise in remote-sensing systems: The effect on classification error. *IEEE Trans. Geosci. Remote Sens.* **1986**, 294–300. [CrossRef]
11. Markham, B.; Townshend, J. *Land Cover Classification Accuracy as a Function of Sensor Spatial Resolution*; NASA: Washington, DC, USA, 1981.
12. Song, M.; Civco, D.; Hurd, J. A competitive pixel-object approach for land cover classification. *Int. J. Remote Sens.* **2005**, *26*, 4981–4997. [CrossRef]
13. Townshend, J.; Huang, C.; Kalluri, S.; Defries, R.; Liang, S.; Yang, K. Beware of per-pixel characterization of land cover. *Int. J. Remote Sens.* **2000**, *21*, 839–843. [CrossRef]
14. Pelletier, C.; Valero, S.; Inglada, J.; Champion, N.; Marais Sicre, C.; Dedieu, G. Effect of training class label noise on classification performances for land cover mapping with satellite image time series. *Remote Sens.* **2017**, *9*, 173. [CrossRef]
15. Rahmati, O.; Ghorbanzadeh, O.; Teimurian, T.; Mohammadi, F.; Tiefenbacher, J.P.; Falah, F.; Pirasteh, S.; Ngo, P.T.T.; Bui, D.T. Spatial Modeling of Snow Avalanche Using Machine Learning Models and Geo-Environmental Factors: Comparison of Effectiveness in Two Mountain Regions. *Remote Sens.* **2019**, *11*, 2995. [CrossRef]
16. Ghorbanzadeh, O.; Blaschke, T.; Gholamnia, K.; Meena, S.R.; Tiede, D.; Aryal, J. Evaluation of different machine learning methods and deep-learning convolutional neural networks for landslide detection. *Remote Sens.* **2019**, *11*, 196. [CrossRef]
17. Shao, Y.; Lunetta, R.S. Sub-pixel mapping of tree canopy, impervious surfaces, and cropland in the Laurentian Great Lakes Basin using MODIS time-series data. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2010**, *4*, 336–347. [CrossRef]
18. Franklin, S.E.; Ahmed, O.S.; Wulder, M.A.; White, J.C.; Hermosilla, T.; Coops, N.C. Large area mapping of annual land cover dynamics using multitemporal change detection and classification of Landsat time series data. *Can. J. Remote Sens.* **2015**, *41*, 293–314. [CrossRef]
19. Goward, S.N.; Markham, B.; Dye, D.G.; Dulaney, W.; Yang, J. Normalized difference vegetation index measurements from the Advanced Very High Resolution Radiometer. *Remote Sens. Environ.* **1991**, *35*, 257–277. [CrossRef]
20. Lunetta, R.S.; Shao, Y.; Ediriwickrema, J.; Lyon, J.G. Monitoring agricultural cropping patterns across the Laurentian Great Lakes Basin using MODIS-NDVI data. *Int. J. Appl. Earth Obs. Geoinf.* **2010**, *12*, 81–88. [CrossRef]
21. Wand, M.P.; Jones, M.C. *Kernel Smoothing*; Chapman and Hall/CRC: London, UK, 1994.
22. Simonoff, J.S. *Smoothing Methods in Statistics*; Springer Science & Business Media: Berlin, Germany, 2012.
23. Chen, J.; Jönsson, P.; Tamura, M.; Gu, Z.; Matsushita, B.; Eklundh, L. A simple method for reconstructing a high-quality NDVI time-series data set based on the Savitzky–Golay filter. *Remote Sens. Environ.* **2004**, *91*, 332–344. [CrossRef]
24. Cao, R.; Chen, Y.; Shen, M.; Chen, J.; Zhou, J.; Wang, C.; Yang, W. A simple method to improve the quality of NDVI time-series data by integrating spatiotemporal information with the Savitzky-Golay filter. *Remote Sens. Environ.* **2018**, *217*, 244–257. [CrossRef]
25. Jonsson, P.; Eklundh, L. Seasonality extraction by function fitting to time-series of satellite sensor data. *IEEE Trans. Geosci. Remote Sens.* **2002**, *40*, 1824–1832. [CrossRef]
26. Jönsson, P.; Eklundh, L. TIMESAT—A program for analyzing time-series of satellite sensor data. *Comput. Geosci.* **2004**, *30*, 833–845. [CrossRef]

27. Sakamoto, T.; Yokozawa, M.; Toritani, H.; Shibayama, M.; Ishitsuka, N.; Ohno, H. A crop phenology detection method using time-series MODIS data. *Remote Sens. Environ.* **2005**, *96*, 366–374. [CrossRef]
28. Geerken, R.; Zaitchik, B.; Evans, J. Classifying rangeland vegetation type and coverage from NDVI time series using Fourier Filtered Cycle Similarity. *Int. J. Remote Sens.* **2005**, *26*, 5535–5554. [CrossRef]
29. Eilers, P.H. A perfect smoother. *Anal. Chem.* **2003**, *75*, 3631–3636. [CrossRef]
30. Atzberger, C.; Eilers, P.H. Evaluating the effectiveness of smoothing algorithms in the absence of ground reference measurements. *Int. J. Remote Sens.* **2011**, *32*, 3689–3709. [CrossRef]
31. Kong, D.; Zhang, Y.; Gu, X.; Wang, D. A robust method for reconstructing global MODIS EVI time series on the Google Earth Engine. *ISPRS J. Photogramm. Remote Sens.* **2019**, *155*, 13–24. [CrossRef]
32. Khanal, N.; Uddin, K.; Matin, M.A.; Tenneson, K. Automatic Detection of Spatiotemporal Urban Expansion Patterns by Fusing OSM and Landsat Data in Kathmandu. *Remote Sens.* **2019**, *11*, 2296. [CrossRef]
33. Shao, Y.; Lunetta, R.S.; Wheeler, B.; Iiames, J.S.; Campbell, J.B. An evaluation of time-series smoothing algorithms for land-cover classifications using MODIS-NDVI multi-temporal data. *Remote Sens. Environ.* **2016**, *174*, 258–265. [CrossRef]
34. Atkinson, P.M.; Jeganathan, C.; Dash, J.; Atzberger, C. Inter-comparison of four models for smoothing satellite sensor time-series data to estimate vegetation phenology. *Remote Sens. Environ.* **2012**, *123*, 400–417. [CrossRef]
35. Kandasamy, S.; Baret, F.; Verger, A.; Neveux, P.; Weiss, M. A comparison of methods for smoothing and gap filling time series of remote sensing observations-application to MODIS LAI products. *Biogeosciences* **2013**, *10*, 4055. [CrossRef]
36. Vaiphasa, C. Consideration of smoothing techniques for hyperspectral remote sensing. *ISPRS J. Photogramm. Remote Sens.* **2006**, *60*, 91–99. [CrossRef]
37. Schindler, K. An overview and comparison of smooth labeling methods for land-cover classification. *IEEE Trans. Geosci. Remote Sens.* **2012**, *50*, 4534–4545. [CrossRef]
38. Saah, D.; Tenneson, K.; Poortinga, A.; Nguyen, Q.; Chishtie, F.; San Aung, K.; Markert, K.N.; Clinton, N.; Anderson, E.R.; Cutter, P.; et al. Primitives as building blocks for constructing land cover maps. *Int. J. Appl. Earth Obs. Geoinf.* **2020**, *85*, 101979. [CrossRef]
39. Potapov, P.; Tyukavina, A.; Turubanova, S.; Talero, Y.; Hernandez-Serna, A.; Hansen, M.; Saah, D.; Tenneson, K.; Poortinga, A.; Aekakkararungroj, A.; et al. Annual continuous fields of woody vegetation structure in the Lower Mekong region from 2000-2017 Landsat time-series. *Remote Sens. Environ.* **2019**, *232*, 111278. [CrossRef]
40. Poortinga, A.; Tenneson, K.; Shapiro, A.; Nquyen, Q.; San Aung, K.; Chishtie, F.; Saah, D. Mapping Plantations in Myanmar by Fusing Landsat-8, Sentinel-2 and Sentinel-1 Data along with Systematic Error Quantification. *Remote Sens.* **2019**, *11*, 831. [CrossRef]
41. Poortinga, A.; Nguyen, Q.; Tenneson, K.; Troy, A.; Bhandari, B.; Ellenburg, W.L.; Aekakkararungroj, A.; Ha, L.T.; Pham, H.; Nguyen, G.V.; et al. Linking earth observations for assessing the food security situation in Vietnam: A landscape approach. *Front. Environ. Sci.* **2019**, *7*, 186. [CrossRef]
42. Poortinga, A.; Aekakkararungroj, A.; Kityuttachai, K.; Nguyen, Q.; Bhandari, B.; Thwal, N.S.; Priestley, H.; Kim, J.; Tenneson, K.; Chishtie, F.; et al. Predictive Analytics for Identifying Land Cover Change Hotspots in the Mekong Region. *Remote Sens.* **2020**, *12*, 1472. [CrossRef]
43. Uddin, K.; Shrestha, H.L.; Murthy, M.; Bajracharya, B.; Shrestha, B.; Gilani, H.; Pradhan, S.; Dangol, B. Development of 2010 national land cover database for the Nepal. *J. Environ. Manag.* **2015**, *148*, 82–90. [CrossRef]
44. Bey, A.; Sánchez-Paus Díaz, A.; Maniatis, D.; Marchi, G.; Mollicone, D.; Ricci, S.; Bastin, J.F.; Moore, R.; Federici, S.; Rezende, M.; et al. Collect earth: Land use and land cover assessment through augmented visual interpretation. *Remote Sens.* **2016**, *8*, 807. [CrossRef]
45. Saah, D.; Johnson, G.; Ashmall, B.; Tondapu, G.; Tenneson, K.; Patterson, M.; Poortinga, A.; Markert, K.; Quyen, N.H.; San Aung, K.; et al. Collect Earth: An online tool for systematic reference data collection in land cover and use applications. *Environ. Model. Softw.* **2019**, *118*, 166–171. [CrossRef]
46. Atzberger, C.; Eilers, P.H. A time series for monitoring vegetation activity and phenology at 10-daily time steps covering large parts of South America. *Int. J. Digit. Earth* **2011**, *4*, 365–386. [CrossRef]
47. Chountasis, S.; Katsikis, V.N.; Pappas, D.; Perperoglou, A. The whittaker smoother and the moore-penrose inverse in signal reconstruction. *Appl. Math. Sci.* **2012**, *6*, 1205–1219.

48. Hermance, J.F.; Jacob, R.W.; Bradley, B.A.; Mustard, J.F. Extracting phenological signals from multiyear AVHRR NDVI time series: Framework for applying high-order annual splines with roughness damping. *IEEE Trans. Geosci. Remote Sens.* **2007**, *45*, 3264–3276. [CrossRef]
49. Hermance, J.F. Stabilizing high-order, non-classical harmonic analysis of NDVI data for average annual models by damping model roughness. *Int. J. Remote Sens.* **2007**, *28*, 2801–2819. [CrossRef]
50. Berman, M. Automated smoothing of image and other regularly spaced data. *IEEE Trans. Pattern Anal. Mach. Intell.* **1994**, *16*, 460–468. [CrossRef]
51. Schmidt, K.; Skidmore, A. Smoothing vegetation spectra with wavelets. *Int. J. Remote Sens.* **2004**, *25*, 1167–1184. [CrossRef]
52. Cochran, W.T.; Cooley, J.W.; Favin, D.L.; Helms, H.D.; Kaenel, R.A.; Lang, W.W.; Maling, G.C.; Nelson, D.E.; Rader, C.M.; Welch, P.D. What is the fast Fourier transform? *Proc. IEEE* **1967**, *55*, 1664–1674. [CrossRef]
53. Guiñón, J.L.; Ortega, E.; García-Antón, J.; Pérez-Herranz, V. Moving average and Savitzki-Golay smoothing filters using Mathcad. *Pap. ICEE* **2007**, *2007*.
54. Nagendra, H.; Karmacharya, M.; Karna, B. Evaluating forest management in Nepal: Views across space and time. *Ecol. Soc.* **2005**, *10*, 24. [CrossRef]
55. Kennedy, R.E.; Yang, Z.; Cohen, W.B. Detecting trends in forest disturbance and recovery using yearly Landsat time series: 1. LandTrendr—Temporal segmentation algorithms. *Remote Sens. Environ.* **2010**, *114*, 2897–2910. [CrossRef]
56. Verbesselt, J.; Hyndman, R.; Newnham, G.; Culvenor, D. Detecting trend and seasonal changes in satellite image time series. *Remote Sens. Environ.* **2010**, *114*, 106–115. [CrossRef]
57. Mingwei, Z.; Qingbo, Z.; Zhongxin, C.; Jia, L.; Yong, Z.; Chongfa, C. Crop discrimination in Northern China with double cropping systems using Fourier analysis of time-series MODIS data. *Int. J. Appl. Earth Obs. Geoinf.* **2008**, *10*, 476–485. [CrossRef]