# An Efficient Solution to Generate Meta-features for Classification with Remote Sensing Time Series

**Roberto U. Paiva**[1,2], **Savio S. T. Oliveira**[1], **Luiz M. L. Pascoal**[1,2], **Leandro L. Parente**[2], **Wellington S. Martins** [1]

[1]Institute of Informatics - Federal University of Goias (UFG)

[2]Image Processing and Geoprocessing Laboratory (LAPIG), UFG

urzedabr@ufg.br, wellington@inf.ufg.br, {savioteles,luizmlpascoal,leal.parente}@gmail.com

***Abstract.*** *Over the last years, the volume of Earth observation (EO) data increased significantly due to the large number of satellites orbiting the planet. These data are being used by automatic classification approaches to generate land-use and land-cover (LULC) products for different landscapes around the world. Dynamic Time Warping (DTW) is a classical method used to measure the similarity between two time series. In this context, DTW-based algorithms are an efficient approach to handle EO time series. These algorithms can be used to generate meta-features (i.e., new features automatically derived from the original features) to improve the performance of classification models. However, these algorithms have a long processing time and depends on large computational resources, making it difficult to use in large data volumes. Seeking to address this limitation, this work presents a full scalable parallel solution to optimize the construction of remote sensing meta-features. Additionally, a new classification strategy is presented, in which, the meta-features generated were used to train and evaluate a Random Forest model. Our results shows that both approaches leads to improvement in execution time and overall accuracy when compared to traditional methods.*

## 1. Introduction

The land-use and land-cover (LULC) presents several change dynamics that can be monitored through the analyzes of remote sensing time series [Foody 2002]. Due to the large number of satellites orbiting the planet, in general, these monitoring initiatives are using a huge volume of Earth observation (EO) data to produce mapping products for large areas of earth's surface, seeking to assist decisions related to food security, environmental conservation, sustainability, greenhouse gases emissions and deforestation [Nepstad et al. 2014, Bonan 2008, Bala et al. 2007, Dewan and Yamaguchi 2009].

The Dynamic Time Warping (DTW) [Sakoe and Chiba 1978], is a classic computer science algorithm introduced in the 1970s. It makes use of dynamic programming to measure the similarity between two time series. In the remote sensing context, some DTW-based algorithms were developed to map LULC changes consistently across the years [Guan et al. 2016, Romani et al. 2010, Maus 2016]. Among these algorithms, the Time-Weighted Dynamic Time Warping (TWDTW) [Maus et al. 2016] stands out as a method sensitive to seasonal climatic changes in the natural and cultivated vegetation. The TWDTW method is currently used jointly with the $k$-Nearest

Neighborhood ($k$-NN) algorithm, to generate meta-features for LULC classification [Dadi 2019, Oliveira et al. 2019, Manabe et al. 2018].

Although the TWDTW is effective in analyzing time series, its face some problems that impact on their performance, which makes it difficult do apply to large volumes of data. In addition, in some regions, the integration between $k$-NN and TWDTW presents low accuracy for LULC classifications [Dadi 2019]. More recent works are exploiting parallel processing to generate meta-features based in the TWDTW, seeking to obtain better performance when dealing with large datasets. These works also perform the classification using more sophisticated machine learning algorithms (e.g. Random Forest) to obtain improvements in the classification's accuracy [Oliveira et al. 2018, Oliveira et al. 2019, Paiva et al. 2020].

A parallel version of the TWDTW algorithm, called SP-TWDTW, obtained a speedup of $246$ times in relation to the original version of TWDTW in R language and $11$ times compared to the sequential version in C ++ [Oliveira et al. 2018]. However, this version presents a low thread usage and scalability limitations. SP-TWDTW is restricted to the size of the time series, which negatively impacts its scalability. Considering the technological advances for new space sensors and the emergence of satellites with high temporal resolution (e.g. PlanetScope), this limitation may undermine the usage of the SP-TWDTW algorithm in the future.

This work uses efficient parallelization strategies, using GPU/CUDA architecture, to propose a new DTW-based algorithm called Rapid-DTW, that computes part of the input data in windows of changeable size. The Rapid-DTW algorithm allows the workload of each thread to be variable according to the input data, thus reducing the idleness of each thread, decreasing the cost of synchronization, and removing the limitation regarding the size of the data input. The experiments carried out showed that the Rapid-DTW obtained better results in terms of performance when compared to the SP-TWDTW. Thus, it was possible to compare large sets of time series with various LULC patterns, in a short time, producing measures of similarity (meta-features). In addition, the result of a Random Forest model classification is presented using the meta-features generated by the Rapid-DTW. The Random Forest model showed accuracy improvements in relation to the classic $k$-NN learning algorithm used with SP-TWDTW in [Oliveira et al. 2019].

The remaining of this paper is organized as follows. Section 2 presents a brief discussion of remote sensing time series and related work. Section 3 discusses the classification of LULC using meta-features. Section 4 presents the proposed strategy, the Rapid-DTW, for computing the dynamic programming matrix in parallel. Section 5 presents the results of the tests and experiments carried out. Finally, in section 6 we present the conclusions and future work.

## 2. Time Series Processing: TWDTW and SP-TWDTW

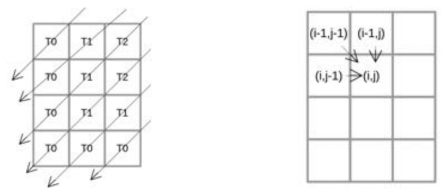
The TWDTW is an extension of the DTW algorithm, and it was designed to work with remote sensing time series. The TWDTW presents a logistical function capable of dealing with different seasonality in data, for example, the seasonality during a crop cultivation. This function is responsible for creating a penalty score in similarities between time series and recognized patterns that are displaced in time. The TWDTW computes two matrices: a matrix of weights $\Psi$ and a dynamic programming matrix, called $D$ matrix.

The weight matrix $\Psi$ is computed through a logistic function based on the difference (in days) between the identified patterns and their time series. From the result of the weight matrix $\Psi$, the algorithm calculates the $D$ matrix, using a recursive sum of the minimum dynamics, according to Equation 1. Then the algorithm uses the matrix $D$ to find the path with the lowest cost, thus generating the measure of similarity between the pattern and the time series.

$$d_{i,j} = \Psi_{i,j} + min\{d_{i-1,j}, d_{i-1,j-1}, d_{i,j-1}\} \tag{1}$$

With the increase in the volume of remote sensing time series data, the computational demand for TWDTW has also increased as its original version was designed to work sequentially. This makes it difficult to analyze large areas, given the greater volume of data to be processed [Oliveira et al. 2018].

Due to the high computational cost to run the TWDTW algorithm, a parallel solution (SP-TWDTW) was proposed in [Oliveira et al. 2018]. This solution is based on the traditional strategy of computing elements in diagonals in parallel wavefront (Figure 1(a)). Each diagonal is processed in parallel, given the dependency on previous elements. In this matrix, the computation of each $(i, j)$ element depends on the $(i - 1, j)$, $(i, j - 1)$ and $(i - 1, j - 1)$ elements previously calculated, as illustrated in Figure 1(b).



(a) Diagonal strategy. (b) Data dependence.

**Figure 1. SP-TWDTW [Oliveira et al. 2018]**

SP-TWDTW target a highly multi-threaded GPU, and calculates one element per thread at each step when computing the $D$ matrix. To ensure correctness, the number of threads in each block is determined as the size of the main diagonal, which is equal to the minimum value of the size of the pattern and the size of the time series. Since this operation is very simple and presents a very low workload for each thread (i.e., each thread computes only one element of the matrix per step of the algorithm), it causes a large amount of processing idleness during its execution, thus decreasing performance. Also, the GPU/CUDA programming architecture is currently limited to 1024 threads for each block [NVIDIA 2018]. Given this limitation, the SP-TWDTW is not able to work for values of similarity measures between patterns and time series when both are greater than 1024.

## 3. Classification Based on Meta-features

The use of machine learning algorithms for classification of LULC is essential to create models that enable automated and recurrent mapping. The DTW-based algorithms can generates similarity measures that can be used as meta-features to carry out classification of LULC. These meta-features are agnostic to the classification model and have been used with some well known classification algorithms.

In most studies, these meta-features are traditionally used as input to the $k$-NN [Belgiu and Csillik 2018] algorithm. However, some studies have reported that the algorithm presents low accuracy when applied in regions with higher data variability within each class [Dadi 2019]. Recent approaches presents the use of meta-features in more sophisticated machine learning algorithms, such as Random Forest and Support Vector Machines (SVM), and have obtained satisfactory results in terms of improving accuracy [Oliveira et al. 2019, Paiva et al. 2020]. In recent years, the Random Forest [Breiman 2001] and SVM [Vapnik 1995] algorithms have become a reference for good remote sensing classifiers [Rodriguez-Galiano et al. 2012, Belgiu and Drăguţ 2016].

In the field of remote sensing, the Random Forest has become the most used algorithm for classification of LULC, as it presents a simple configuration and obtains good accuracy [Pal 2005, Gislason et al. 2006, Belgiu and Drăguţ 2016]. According to [Pal 2005] Random Forest's popularity occurs due to its ability to achieve an accuracy similar to SVM, along side the ease of usage, with few parameters to be configured by the user and trivial adaptation to remote sensing data. Some recent works have used Random Forest for mapping large areas, showing its effectiveness when working with a large volume of data [Ayala-Izurieta et al. 2017, Parente and Ferreira 2018, Tsai et al. 2018].

## 4. Rapid-DTW

The Rapid-DTW, a new DTW-based algorithm, computes the elements of the dynamic programming matrix using windows of changeable size. The idea is to choose the workload performed by the threads at each step of the algorithm, allowing the method to experiment with different window sizes to obtain an ideal configuration for a specific instance of the problem. This decreases the idleness of threads and the cost of synchronization, which improves the use of GPU resources, enabling better performance of the algorithm.

The Rapid-DTW changes the flow in which the $D$ matrix is computed. Rather than performing the computation of the elements in a diagonal flow, as the SP-TWDTW does [Oliveira et al. 2018], the window strategy performs the computation in elements within a given window. This results in the computation flow to be carried out vertically, allowing the dependence of data internally in each window in a sequential manner while guaranteeing the dependence of data between the windows of elements in parallel. Figure 2 illustrates the computation flow given the new strategy.

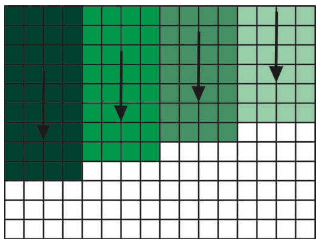


**Figure 2. Computation flow of the $D$ matrix with Rapid-DTW.**

It is easy to notice that, the larger the window is set, the greater is the workload of each thread, which implies fewer idle threads in each step of the algorithm. As illustrated in Figure 3, in each window the thread runs sequentially, ensuring data dependency, while windows with the same color can be processed in parallel. The window size variability allows adaptation of the number of threads to be aligned to the size of the input problem. This variation enables to calculate similarity measures between patterns and time series for any given size of data entry.

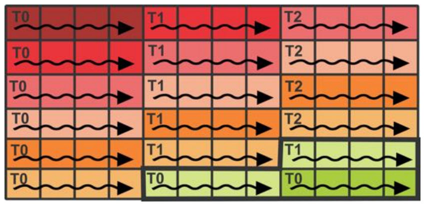


**Figure 3. Behavior of threads inside the window.**

Due to data dependency inherent to the dynamic programming algorithm, the number of synchronizations to maintain the correctness of the algorithm is very high. Therefore, the Rapid-DTW also works with smaller number of threads, thus significantly reducing the number of synchronizations.

The strategy for computing the programming matrix $D$ is described in Algorithm 1. In lines 1-4, constant variables are defined. In line 2, the window size is calculated according to the number of threads. Two auxiliary variables, $auxj$ in line 7 and $aux$ in line 23, are used to locate the elements of each window, while the $base$ variable, declared in line 6, is responsible for keeping each thread within the specified window size.

The $for$ loop in lines 5 to 18 computes the upper part of the matrix, and similarly the $for$ loop in lines 21 to 35 computes the lower part. The conditionals in lines 8 and 25 control the number of threads working in parallel, while lines 17 and 34 perform the synchronization. The inner $for$ loops in lines 10 to 15 and 27 to 32 are used to locate where the elements of the windows are computed in parallel. Finally, an auxiliary function, $update\_element(i, j)$, is used to update each element according to the presented Equation 1.

## 5. Experiments and Results

In this section, we will detail the conditions for carrying out the experiments, as well as the results obtained. In this paper we defined two experiments. The first experiment was designed to evaluate the performance of Rapid-DTW when compared to SP-TWDTW and TWDTW in C++. In the second experiment, we applied the generated meta-features to train and evaluate a Random Forest model for LULC classification. All experiments were performed on a computer with an Intel Core i7-9700 processor (3.2 GHz and 8 MB Cache), 16GB DDR4 RAM, and NVIDIA GeForce GTX 1660 Ti video card with 6 GB GDDR6 of memory with Turing architecture, 1536 CUDA cores, 1770 MHz of frequency.

In the first experiment, the implementations for the Rapid-DTW, SP-TWDTW, and TWDTW algorithms were executed 10 times in each scenario and the average computed. Moreover, significance tests (paired t-tests) were run to test for significant differences, with 95% confidence. The TWDTW and SP-TWDTW codes were obtained directly from the work published in [Oliveira et al. 2018]. The three algorithms are used to generate the meta-features, therefore, this experiment was designed to assess the impact in terms of execution time when different number of observations patterns and time series are applied. Thus, in order to compare the execution time of the algorithms and carry out the experiments with time series of different sizes, input data was generated synthetically from the MODIS13Q1 database presented in [Maus et al. 2016].

**Algoritmo 1:** Rapid-DTW - Computation of $D$ matrix

```
Data: $\Psi$ Weight matrix
$y$: number of lines
$x$: number of columns
$num\_threads$: number of threads
Result: $D$ Matrix
$tid \leftarrow$ thread id
$windowSize \leftarrow x/num\_threads$
$tidWindow \leftarrow tid * windowSize$
$tidWindowaux \leftarrow tid * (windowSize - 1)$
for $(si = 0; si < y; si++)$ do
    $base \leftarrow tidWindow + (si - tid) * x$
    $auxj \leftarrow tidWindowaux$
    if $(tid \leq min(si, x - 1))$ then
        All threads in paralalel do:
        for $(index = base; index < base + windowSize; index++)$ do
            $i \leftarrow si - tid$
            $j \leftarrow tid + auxj$
            $update\_element(i, j)$
            $auxj \leftarrow auxj + 1$
        end
    end
    $sync\_barrier$
end
$si \leftarrow (y - 1 - tid) * x$
$auxj \leftarrow 0$
for $sj \leftarrow ((x/windowSize) - 2; sj \geq 0; sj--)$ do
    $base \leftarrow tidWindow + si + windowSize + auxj$
    $aux \leftarrow 0$
    $auxj \leftarrow auxj + windowSize$
    if $(tid \leq min(sj, y - 1))$ then
        All threads in paralalel do:
        for $(index = base; index < base + windowSize; index++)$ do
            $i \leftarrow y - tid - 1$
            $j \leftarrow x - (windowSize * sj) - windowSize + aux + tidWindow$
            $update\_element(i, j)$
            $aux \leftarrow aux + 1$
        end
    end
    $sync\_barrier$
end
```

Initially, tests were performed to evaluate the performance of the Rapid-DTW on datasets with sizes commonly used in the literature for classification using the MODIS13Q1. In this test, 50 patterns and 1000 time series were generated, with 24 observations for the patterns and 50 observations for the time series. Then, to carry out the experiment on larger time series, with the objective to demonstrate our method's ability to process time series for the next generation of satellites, another set of 50 patterns, and 1000 time series with sizes ranging from 48 to 768 observations for patterns and 100 to 1600 observations for the time series was generated. A last data set was created to test patterns and time series simulating spatial sensors with really high temporal resolution. In this set, 5 patterns and 10 time series were generated, with sizes ranging from 1536 to 12288 observations for patterns and 3200 to 24800 observations for the time series. The SP-TWDTW was not used in this last dataset, due to its scalability limitation.

In order to generate the meta-features, the database presented in [Picoli et al. 2018] was used. This database presents MODIS13Q1 data extracted from the region of Mato Grosso - Brazil, to classify nine different classes of LULC. The samples are distributed as follows: Cerrado (400), Cotton-fallow (34), Forest (138), Pasture (370), Soy-corn (398), Soy-cotton (399), Soy-fallow (88), Soy-millet (235) and Soy-sunflower (53), totaling 2115 samples with 23 observations of NIR, MIR, EVI and NVDVI bands each sample. Te meta-features were generated for each point within the samples using the Rapid-DTW. The meta-features present the distances for each class (i.e., nine values for each point) bearing in mind that each class represents a pattern.

The Random Forest model was trained using 500 trees, as there was no improvement in the results with the increase in the number of trees. To estimate accuracy, all experiments were performed using the K-Fold cross-validation technique, with K = 5. Finally, a matrix of confusion was generated with the producer accuracy (PA), the user accuracy (UA), and overall accuracy (OA) from the results obtained. The Kappa coefficient was also calculated in order to analyze a baseline classification result.

## 5.1. Performance Analysis of Rapid-DTW

Using patterns of size 24 and time series of size 45 all multiple values of the main diagonal, which in this case has size 24, were tested. Figure 4 presents the results of this experiment, in which the Rapid-DTW presented the lower response time overall, during the computation of the $D$ matrix, specifically $2.4$ times faster than SP-TWDTW and $5.3$ times faster than TWDTW. The best results were obtained when a 2 elements window size is applied, which is the smallest multiple of 24. This occurs because the number of threads per block in this input set is less than 32, which is favored since the CUDA architecture makes use of warps, with minimum sets of 32 threads that share instructions and memory. Therefore, for this scenario, reducing drastically the number of threads ends up harming more than the gain obtained with the decrease of idleness and synchronization cost, causing a negative result in performance.

The next test was designed to present the execution time of the three aforementioned algorithms, performing all the necessary steps to calculate the similarity measure. The results are illustrated in Figure 5(a). Next, Figure 5(b) presents a new set of tests with pattern and time series sizes over 1024, in which the SP-TWDTW was unable to perform due to limitation of threads per block in CUDA architecture. We omit confidence intervals in both graphs for the sake of clarity.

Overall, the performance of Rapid-DTW is better when the sizes of patterns and time series increase. With the results obtained in the experiments, it was possible to compare Rapid-DTW versus SP-TWDTW, in which the best performance was obtained

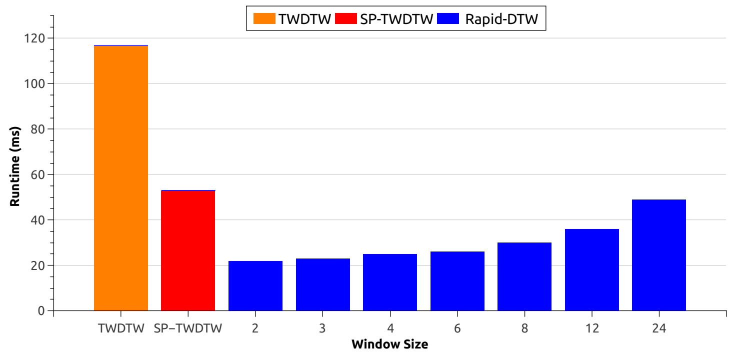
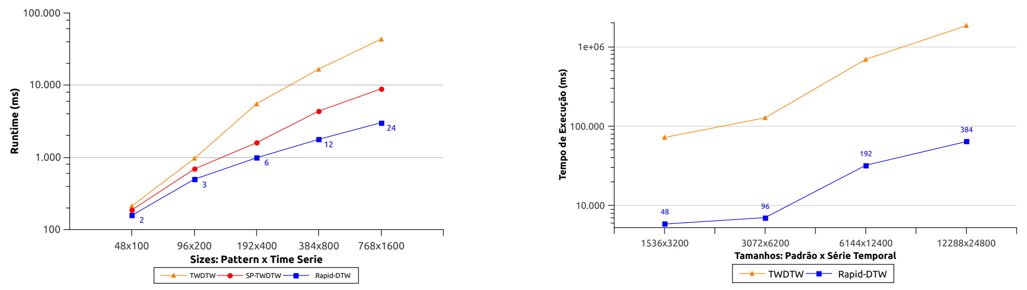


**Figure 4. Execution time of the $D$ matrix by TWDTW, SP-TWDTW and Rapid-DTW with patterns size 24 and time series size 50.**

using the largest data set ($768x1600$), with an improvement in runtime of $2.9$ times faster. Compared to the sequential version, Rapid-DTW achieved a speedup of up to $28.8$ times on the largest data set ($12288x24800$). This improvement is due to the computation time of the dynamic programming matrix, which takes significantly longer than the rest of the steps, totaling $87\%$ of the runtime.



(a) TWDTW x P-TWDTW x Rapid-DTW with window size ranging from 2 to 24.

(b) TWDTW x Rapid-DTW with window sizes ranging from 48 to 384.

**Figure 5. Runtime of all steps of the algorithms. The blue numbers correspond to the window size used for the input set. Axis $y$ in logarithmic scale.**

## 5.2. Meta-features classification

The second experiment aims to evaluate the classification accuracy using the meta-features generated by the Rapid-DTW. In this scenario, the meta-features were applied as input to the Random Forest algorithm. Each point in the sample presents 9 meta-features according to the classes to be classified. The Table 1 shows an example of the algorithm entry for three pixels.

| Pixel | Cerrado | Fallow_Cotton | Forest | Pasture | Soy_Corn | Soy_Cotton | Soy_Fallow | Soy_Millet | Soy_Sunflower |
|---|---|---|---|---|---|---|---|---|---|
| **1** | 3,53 | 4,03 | 4,00 | 2,67 | 3,62 | 4,45 | 4,34 | 3,29 | 4,07 |
| **2** | 2,62 | 4,27 | 2,91 | 2,22 | 3,70 | 4,83 | 4,59 | 3,33 | 4,19 |
| **3** | 1,70 | 4,08 | 3,34 | 1,72 | 3,31 | 4,71 | 4,15 | 2,70 | 3,79 |

**Table 1. Example of the meta-features generated by Rapid-DTW as an input vector for Random Forest.**

Table 2 presents the confusion matrix with the results of this experiment. The Random Forest model showed an overall accuracy of 84.02% using the 9 meta-features (Kappa 0.81). The same experiment on the same data set was carried out in the work of [Oliveira et al. 2019], in which was proposed a combination of SP-TWDTW and $k$-NN that obtained an overall accuracy of 78%. Random Forest was able to obtain a better result, in terms of accuracy, of 6.2% in relation to $k$-NN, with no addition of new features.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | **UA(%)** |
|---|---|---|---|---|---|---|---|---|---|---|
| **1 Cerrado** | **375** | 0 | 7 | 16 | 0 | 0 | 0 | 0 | 0 | 94,25% |
| **2 Fallow_Cotton** | 0 | **13** | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 85,29% |
| **3 Forest** | 11 | 0 | **126** | 4 | 0 | 0 | 0 | 0 | 0 | 89,13% |
| **4 Pasture** | 14 | 0 | 5 | **343** | 2 | 0 | 0 | 7 | 0 | 92,43% |
| **5 Soy_Corn** | 0 | 2 | 0 | 1 | **326** | 48 | 0 | 44 | 35 | 67,34% |
| **6 Soy_Cotton** | 0 | 17 | 0 | 1 | 22 | **333** | 0 | 6 | 2 | 87,97% |
| **7 Soy_Fallow** | 0 | 1 | 0 | 0 | 0 | 1 | **81** | 7 | 0 | 89,77% |
| **8 Soy_Millet** | 0 | 1 | 0 | 5 | 46 | 12 | 7 | **171** | 7 | 66,81% |
| **9 Soy_Sunflower** | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | **9** | 96,23% |
| **PA(%)** | 93,75% | 38,24% | 91,30% | 92,70% | 81,91% | 83,46% | 92,05% | 72,77% | 16,98% | **OA = 84,02%** |

**Table 2. Confusing matrix of the application of meta-features to Random Forest.**

Still in Table 2, it can be seen that according to the producer's accuracy the Random Forest obtained good results (PA over 90%) with Cerrado, Forest, Pasture and Soy_Fallow classes. The agriculture-related classes present similar behavior in their time series, thus hampering the classifier in differentiating them and consequently obtaining good accuracy results [Picoli et al. 2018, Paiva et al. 2020].

Finally, given the similarities between the agriculture classes, a new experiment was conducted by merging the classes Fallow_Cotton, Soy_Corn, Soy_Cotton, Soy_Fallow, Soy_Sunflower into a single class called Agriculture. In this experiment a better result is observed using only 9 meta-features, in which the classifier managed to obtain an OA of 96.50% (Kappa 0.94). Regarding the new class of Agriculture, a PA of 99.59% was obtained. In scenarios where there is no need to typify the classification of agriculture, the use of this simplification is encouraged given its significant results. Table 3 shows the confusion matrix for this experiment.

| | 1 | 2 | 3 | 4 | **UA (%)** |
|---|---|---|---|---|---|
| **1 Cerrado** | **375** | 7 | 16 | 0 | 94,25% |
| **2 Forest** | 11 | **127** | 3 | 0 | 89,86% |
| **3 Pasture** | 14 | 4 | **337** | 5 | 93,78% |
| **4 Agriculture** | 0 | 0 | 14 | **1202** | 98,84% |
| **PA(%)** | 93,75% | 92,03% | 91,08% | 99,59% | **OA = 96,50%** |

**Table 3. Confusion matrix of the application of meta-features simplifying the agriculture classes.**

## 6. Conclusions and Future Work

Given the large number of satellites being constantly launched into Earth's orbit and their increasingly powerful sensors, it is expected that the sizes and quantities of remote sensing time series continue to increase in the near future. However, the computational cost to process this volume of data should also increase proportionally to the size and quantity of time series. The future of Remote Sensing may depends on the exploitation of high performance computing for the efficient processing of this huge volume of data [Houborg and McCabe 2018, Hansen and Loveland 2012, Plaza 2008, Camara et al. 2016]

In this work, the Rapid-DTW algorithm was presented, exploiting parallel processing for dealing with remote sensing data and applying the generated meta-features into the classification of land use and cover, demonstrating the potential of high performance computing techniques in the area of remote sensing. The conducted experiments shows that the Rapid-DTW presents significant improvement on runtime over traditional methods regarding the generation of meta-features.

Additionally, this work also proposed the application of the generated meta-features into the Random Forest, a state-of-the-art classifier. The experiments reported an improvement of 6.2% in overall accuracy when compared to the traditional $k$-NN. The results also encouraged a new experiment in which the agriculture related classes were merged into a single class, which lead to a significant overall accuraccy of 96.5% using the nine generated meta-features.

The use of Rapid-DTW in conjunction with the Random Forest algorithm allowed the analysis of a classification methodology capable of generating good accuracy results with few characteristics. Rapid-DTW is the first step towards creating an application capable of generating meta-features for various data from satellite time series. Once meta-features are generated for a given set of data, they can be used in various classification analyzes. These meta-features can also be used in several other classification methodologies, being incorporated into the input vectors of the machine learning algorithms.

However, there are some limitations to this work. This methodology does not deal directly with the satellite images. Therefore, all inputted data must be pre-processed in order to create a table with the time series data and their patterns. In future work, we propose the application of our method in different regions in order to verify its performance in runtime and accuracy. For that, we intend to create a method to directly process the satellite images and input its results to Rapid-DTW into a more complete methodology.

## References

Ayala-Izurieta, J. E., Márquez, C. O., García, V. J., Recalde-Moreno, C. G., Rodríguez-Llerena, M. V., and Damián-Carrión, D. A. (2017). Land cover classification in an ecuadorian mountain geosystem using a random forest classifier, spectral vegetation indices, and ancillary geographic data. *Geosciences*, 7(2):34.

Bala, G., Caldeira, K., Wickett, M., Phillips, T., Lobell, D., Delire, C., and Mirin, A. (2007). Combined climate and carbon-cycle effects of large-scale deforestation. *Proceedings of the National Academy of Sciences*, 104(16):6550–6555.

Belgiu, M. and Csillik, O. (2018). Sentinel-2 cropland mapping using pixel-based and object-based time-weighted dynamic time warping analysis. *Remote sensing of environment*, 204:509–523.

Belgiu, M. and Drăguţ, L. (2016). Random forest in remote sensing: A review of applications and future directions. *ISPRS Journal of Photogrammetry and Remote Sensing*, 114:24–31.

Bonan, G. B. (2008). Forests and climate change: forcings, feedbacks, and the climate benefits of forests. *science*, 320(5882):1444–1449.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Camara, G., Assis, L. F., Ribeiro, G., Ferreira, K. R., Llapa, E., and Vinhas, L. (2016). Big earth observation data analytics. In *Proceedings of the 5th ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data - BigSpatial 16*. ACM Press.

Dadi, M. M. (2019). Assessing the transferability of random forest and time-weighted dynamic time warping for agriculture mapping. Master's thesis, University of Twente, Enschede.

Dewan, A. M. and Yamaguchi, Y. (2009). Land use and land cover change in greater dhaka, bangladesh: Using remote sensing to promote sustainable urbanization. *Applied geography*, 29(3):390–401.

Foody, G. M. (2002). Status of land cover classification accuracy assessment. *Remote Sensing of Environment*, 80(1):185–201.

Gislason, P. O., Benediktsson, J. A., and Sveinsson, J. R. (2006). Random forests for land cover classification. *Pattern Recognition Letters*, 27(4):294–300.

Guan, X., Huang, C., Liu, G., Meng, X., and Liu, Q. (2016). Mapping rice cropping systems in vietnam using an ndvi-based time-series similarity measurement based on dtw distance. *Remote Sensing*, 8(1):19.

Hansen, M. C. and Loveland, T. R. (2012). A review of large area monitoring of land cover change using landsat data. *Remote Sensing of Environment*, 122:66–74.

Houborg, R. and McCabe, M. F. (2018). A cubesat enabled spatio-temporal enhancement method (CESTEM) utilizing planet, landsat and MODIS data. *Remote Sensing of Environment*, 209:211–226.

Manabe, V. D., Melo, M. R., and Rocha, J. V. (2018). Framework for mapping integrated crop-livestock systems in mato grosso, brazil. *Remote Sensing*, 10(9):1322.

Maus, V. (2016). *Land Use and Land Cover Monitoring Using Remote Sensing Image Time Series*. PhD thesis, PhD thesis, Instituto Nacional de Pesquisas Espaciais, Sao José dos Campos.

Maus, V., Câmara, G., Cartaxo, R., Sanchez, A., Ramos, F. M., and De Queiroz, G. R. (2016). A time-weighted dynamic time warping method for land-use and land-cover mapping. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 9(8):3729–3739.

Nepstad, D., McGrath, D., Stickler, C., Alencar, A., Azevedo, A., Swette, B., Bezerra, T., DiGiano, M., Shimada, J., da Motta, R. S., et al. (2014). Slowing amazon deforestation through public policy and interventions in beef and soy supply chains. *science*, 344(6188):1118–1123.

NVIDIA, C. (2018). Cuda c programming guide, version 9.1. *NVIDIA Corp.*

Oliveira, S. S., Cardoso, M. d. C., Bueno, E., Rodrigues, V. J., and Martins, W. S. (2019). Exploiting parallelism to generate meta-features for land use and land cover classification with remote sensing time series. *Brazilian Symposium on Geoinformatics (GeoInfo)*, pages 135–146.

Oliveira, S. S., Pascoal, L. M., Ferreira, L., Cardoso, M. d. C., Bueno, E., Rodrigues, V. J., and Martins, W. S. (2018). Sp-twdtw: A new parallel algorithm for spatio-temporal analysis of remote sensing images. *Brazilian Symposium on Geoinformatics (GeoInfo)*, pages 46–57.

Paiva, R., Oliveira, S., Martins, W., and Parente, L. (2020). Análise de metacaracterísticas para classificação de uso e cobertura do solo utilizando random forest. In *Anais do XI Workshop de Computação Aplicada à Gestão do Meio Ambiente e Recursos Naturais*, pages 71–80. SBC.

Pal, M. (2005). Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, 26(1):217–222.

Parente, L. and Ferreira, L. (2018). Assessing the spatial and occupation dynamics of the brazilian pasturelands based on the automated classification of modis images from 2000 to 2016. *Remote Sensing*, 10(4):606.

Picoli, M. C. A., Camara, G., Sanches, I., Simões, R., Carvalho, A., Maciel, A., Coutinho, A., Esquerdo, J., Antunes, J., Begotti, R. A., et al. (2018). Big earth observation time series analysis for monitoring brazilian agriculture. *ISPRS journal of photogrammetry and remote sensing*, 145:328–339.

Plaza, A. (2008). *High performance computing in remote sensing*. Chapman & Hall/CRC, Boca Raton, FL.

Rodriguez-Galiano, V. F., Ghimire, B., Rogan, J., Chica-Olmo, M., and Rigol-Sanchez, J. P. (2012). An assessment of the effectiveness of a random forest classifier for land-cover classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, 67:93–104.

Romani, L. A., Goncalves, R., Zullo, J., Traina, C., and Traina, A. J. (2010). New dtw-based method to similarity search in sugar cane regions represented by climate and remote sensing time series. In *2010 IEEE International Geoscience and Remote Sensing Symposium*, pages 355–358. IEEE.

Sakoe, H. and Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49.

Tsai, Y. H., Stow, D., Chen, H. L., Lewison, R., An, L., and Shi, L. (2018). Mapping vegetation and land use types in fanjingshan national nature reserve using google earth engine. *Remote Sensing*, 10(6):927.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer New York.