Belief Propagation, Classifying MODIS time-series, and Fast-Fourier Transform

Gunther Tonitz, *Stellenbosch University*, 25040863
Steffan Schoonbee, *Stellenbosch University*, 23535237
Dirk Hoffmann, *Stellenbosch University*, 24758477
Franco Uys, *Stellenbosch University*, 23583584
Shriyan Singh, *Stellenbosch University*, 24902098

***Abstract*—This report discusses belief propagation as a form of error-correction coding. It presents an analysis of techniques for classifying MODIS time-series data, focusing on vegetation and settlement land cover types in the Gauteng province of South Africa. The study employs dimensionality reduction using the Fast Fourier Transform (FFT), followed by classification using Logistic Regression and Naïve Bayes methods. Additionally, a Sequential Probability Ratio Test (SPRT) model is applied to capture time-varying patterns in the dataset. The report demonstrates the comparative performance of these methods, particularly highlighting the efficacy of FFT for reducing data complexity and improving classification accuracy. The results underscore the importance of accurate land-cover classification in monitoring settlement expansion, which is critical for sustainable development.**

***Index Terms*—land cover classification, hamming codes, dimension reduction**

I. Introduction

Hamming codes are a form of systematic error-correcting codes. They are easy to understand and simple to implement. Error-correcting codes need to be used when noise can be expected in the transmission channel. Satellites send data over a vast distance and a reasonable amount of noise can be expected in their transmission channel. Error-correcting codes, such as Hamming codes, thus have a clear use case in Remote sensing.

Different classification techniques are compared on a hypertemporal Moderate Resolution Imaging Spectroradiometer (MODIS) dataset to classify between vegetation and settlement time-series. The first involves dimensionality reduction on several observations of the same pixel. Logistic Regression and Naïve Bayes classifiers are then fitted to the reduced dataset to classify land cover type. The second technique applies the Sequential Probability Ratio Test (SPRT) along with a time-varying model to classify land cover type.

Classification of land cover types is an important study in Southern Africa since settlement expansion is very prominent. Monitoring settlement growth can help enforce sustainable development to reduce negative environmental impact [6].

Fourier Transforms are an important component in signal processing by converting time-series data into the frequency domain. This is crucial for analyzing periodic patterns in the data. The Fast Fourier Transform (FFT), a computationally efficient algorithm, allows for faster and more efficient transformations, making it ideal for processing large datasets such as the MODIS time-series.

The remainder of the report is structured as follows: Section II provides definitions on different techniques and thus provides the necessary background to understand what follows. Section III contains formulas used to apply the techniques to the specific domain. Section IV details pseudocode for some of the implementations. Section V presents figures and tables to convey research results. Finally, Section VI concludes the report with a summary of the findings.

II. Definitions

A. Belief Propagation

The Sum-Product Algorithm can be applied to a factor graph of a function to calculate the marginal function. the update rule works as follows: The message sent from a node v on edge e is the product of the function at v with the messages received at v on all edges other than e, summed over all variables except those associated with e. Belief Propagation is when the sum-product update rule is applied to the factor graph of a sparse graph code [1].

B. Hamming Codes

Originally proposed by R. W. Hamming, Hamming Codes is a form of error-detecting and correcting codes. Hamming Codes are *systematic* codes which means k parity bits are added to the original message [5]. In the case of the *Hamming(7,4,3)* format the message length is 7 and 3 parity bits are added. These parity bits can be used to correct errors with the belief propagation algorithm [1].

C. Dataset

A hypertemporal time-series is a time-series that consists of frequent equal-spaced observations [6]. The dataset used in this case study comprises 230 km^2 of the Gauteng province of South Africa. There are 925 observed pixels (592 vegetation, 333 settlement) and 368 samples. Since 45 yearly measurements are drawn, we conclude that the dataset spans just over eight years. Each pixel also consists of eight time-series corresponding to the first seven bands of MODIS and the Normalised Difference Vegetation Index (NDVI) [2]. The NDVI is obtained by [6],

$$\text{NDVI} = \frac{\text{Band } 2 - \text{Band } 1}{\text{Band } 2 + \text{Band } 1}. \tag{1}$$

D. Dimension Reduction

Dimensionality reduction is the process of converting a set of features into a set of fewer features, which best capture the data characteristics. Such techniques have proven useful in many applications [9]. One such technique reduces the dataset into two harmonic features [2]. In signal processing, a frequency domain can be decomposed into a fundamental frequency and its harmonics. These harmonics are components having frequencies that are integer multiples of the fundamental frequency and can help capture the periodicity in the data. To transform the data from a time-domain to a frequency domain, the Fast Fourier Transform (FFT) is applied which represents the signal as functions of sine and cosine waves [8]. A MODIS pixel can be reduced to its mean and seasonal component for each band by utilising this technique. [2].

E. Time-varying Model

In a statistical framework, time-varying models consider data at different points in time to be generated by different underlying distributions. Thus, unlike previous attempts to classify land-cover types, sequential time-varying models can accurately capture the seasonality contained in hyper-temporal data [7].

F. Sequential Detection

The sequential probability ratio test (SPRT) is an extension of Neyman Pearson for sequential detection. Given a set of i.i.d. observations in the sequence $\mathbf{z}_k = \{z_k\}_{k=1,2,...}$ the goal is to decide between two hypotheses

$$H_0 : z_k \sim Q_0, k = 1, 2, ...$$

versus

$$H_1 : z_k \sim Q_1, k = 1, 2, ...$$

where Q_0 and Q_1 are two probability distributions with densities q_0 and q_1. We can resolve this task by taking a cumulative sum of the log-likelihood ratios as the sequence progresses and classifying once an appropriate exit threshold is breached [6].

G. Fast Fourier Transform

The Fast Fourier Transform (FFT) is an algorithm that efficiently computes the Discrete Fourier Transform (DFT) [10] with a computational complexity of $O(N \log_2 N)$, as opposed to the $O(N^2)$ complexity of the DFT. The FFT uses recursion to break down a large problem into smaller subproblems and combines their results, making it ideal for large datasets.

H. Radix-2 Decimation in Time

The Radix-2 Decimation in Time (DIT) FFT is a specific version of the FFT algorithm that recursively splits the input into even and odd indexed elements. This approach allows the FFT to compute the DFT more efficiently for input sizes that are powers of two ($N = 2^n$).

I. Twiddle Factors

Twiddle factors are complex exponential terms of the form $e^{-2\pi ik/N}$ that are used in FFT algorithms to combine the results of recursive subproblems. They capture the periodicity of the signal and are essential in merging the even and odd parts of the input during the FFT process.

III. Formulas

A. Additive White Gaussian Noise

Noise is added to the bits when transmitting it over the communication channel with $N(0, \sigma^2)$ where σ^2 is:

$$\sigma^2 = \frac{1}{2 \cdot (R_c) \cdot 10^{\frac{\text{SNR}}{10}}} \tag{2}$$

where R_c is the code ratio and SNR is the signal-to-noise ratio. For *Hamming(7,4)* the R_c is $\frac{4}{7}$.

B. Hamming codes

H is the parity check matrix. For Hamming(7,4,3) the H matrix is defined as:

$$H = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \tag{3}$$

The Generator matrix G is defined as:

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \tag{4}$$

To encode a codeword α with Hamming codes the dot product is computed between α and the G matrix.

C. Belief Propagation

For the belief propagation algorithm three different update rules are defined, check- and code-node updates and the final update step. The check- and code-node update steps are intermediate update steps. The log-likelihoods are initialised as [1]:

$$\Lambda(r_l \mid t_l) = -r_l \alpha_l \frac{4}{N_0} \tag{5}$$

For readability's sake:

$$k' = \forall i \in \{1, 2, \ldots, n\} \setminus \{k\} \tag{6}$$

Now the check-nodes are updated with the following formula [1]:

$$\Lambda(X_k \mid Y_{k'}) = -2 \tanh^{-1} \prod_{k'} - \tanh\left(\frac{\Lambda(X_i \mid Y_i)}{2}\right) \tag{7}$$

The code-nodes are updated with the following formula [1]:

$$\Lambda(X_k \mid C_{k'}) = \sum_{k'} \Lambda(X_k \mid C_i) \tag{8}$$

The algorithm ends on a check-node update after which the following final update is made to obtain the final marginal values [1]:

$$\Lambda(X_k \mid C_{\forall i \in \{1,2,\ldots,n\}}) = \sum_{\forall i \in \{1,2,\ldots,n\}} \Lambda(X_k \mid C_i) \quad (9)$$

For belief propagation, an early stopping criteria was defined. If a code-node update has been performed, and the codeword can be decoded into a valid keyword we can stop the iterations [4]. First, the original message bits α are extracted and we compute $\alpha \dot{H}$. If this is equal to the zero matrix $\mathbf{0}$, the iterations can cease.

D. Harmonic Features

To obtain the harmonic features for each band of a modus pixel, we apply the FFT to the time-series and obtain the mean and seasonal component via:

$$[\mathbf{Y}_2]_{ij} = \begin{cases} |\mathcal{F}[x_i(t)][0]| & \text{if } j = 1 \\ 2|\mathcal{F}[x_i(t)][f_s]| & \text{if } j = 2 \end{cases} \quad (10)$$

Here $\mathbf{Y}_2$ is an $n \times 2$ matrix where n is the number of pixels, $x_i(t)$ denotes the time-series of MODIS pixel i in band b, and $\mathcal{F}\{\}$ denotes the Fourier Transform. Frequency 0 is the first or fundamental component and represents the mean of the time-series data. Frequency $f_s = \frac{1}{45}$ returns the seasonal component, since 45 observations are made per year, we expect every 45 observations apart to repeat in season [3].

E. Time-varying Sequential Model

Consider an observed MODIS pixel $Z^c = \{\mathbf{z}_k^c\}_{k=1,2,\ldots}$ which belongs to class $c \in C$. Let us further assume a single band $b \in B$ for $z_k^{c,b}$. Since the pixels can be considered a stochastic process, we can determine the first-order statistical description equivalent to a set of probability density functions ${q_k^{c,b}}_{k=1,2,\ldots}$. Assuming the MODIS pixel is periodic with 45 observations per year, we have that $q_k^{c,b} = q_{k+45}^{c,b}$. Thus, seasonality repeats annually, over 45 lags. Therefore, the pixels can be represented by 45 underlying distributions corresponding to annual observations. Given that there are 8 bands, the total is 720 underlying distributions per class. To estimate these distributions, the observations belonging to the same period are grouped together with

$$G_i^{c,b} = \underset{i=45n}{\text{pr}} Z^{c,b}, n = 0, 1, \ldots, N, 1 \leq i \leq 45 \quad (11)$$

where pr is the projection operator and N is the number of years. Thus, we simply group together all observations of the same season, band and class and use this data to estimate the underlying distributions [3]. To estimate each underlying distribution, a Gaussian Mixture Model (GMM) with a single component is used.

F. SPRT algorithm

The classification of land cover is considered to be equivalent to deciding between the following two hypotheses:

$$H_0 : z_k^b \sim Q_k^{c_0,b}, k = 1, 2, \ldots$$

versus

$$H_1 : z_k^b \sim Q_k^{c_1,b}, k = 1, 2, \ldots$$

where H_0 is settlement and H_1 is vegetation [3]. The SPRT algorithm calculates a cumulative sum of the log-likelihood ratios (LLR) to decide between hypotheses. Thus,

$$S_k = \sum S_i, \quad S_i = \ln \frac{q_i^{c_1,b}(z_i^b)}{q_i^{c_0,b}(z_i^b)}$$

Unlike the SPRT algorithm, an exit threshold is not used, since the observations are not independently and identically distributed. The likelihood is calculated by using the applicable time-varying model, given the season and band of the pixel being classified. Classification is made after completing the sum of the LLR's for the entire span of observations where $S_k > 0$ classifies to H_1 and H_0 otherwise [7].

G. Discrete Fourier Transform

The Discrete Fourier Transform (DFT) transforms a signal from the time domain to the frequency domain. The DFT is defined as:

$$Y_k = \sum_{n=0}^{N-1} x_n e^{-2\pi i k n / N}$$

where N is the number of points in the signal, x_n are the time-domain samples, and Y_k are the corresponding frequency-domain values [10].

IV. Pseudocode

A. Belief Propagation

For the uncoded binary phase shift keying the following steps need to be followed for every value of SNR:

1) Pick a random four-bit keyword
2) Modulate the keyword using BPSK
3) Inject random noise with AWGN using the SNR
4) Demodulate the noise bits with BPSK
5) Compare the Demodulated bits to the original keyword

For belief propagation with Hamming encoding, only two steps need to be added. Parity bits need to be added to the keywords and the received bits are decoded using the belief propagation algorithm 2.

This process is repeated until a total of 10,000 errors have occurred to ensure reliability. This process is extremely slow since the experiment has to be repeated many times to obtain 10,000 errors. Parallelization and precomputing the modulated keywords are the two main speed-ups implemented. Originally each SNR value's BER was computed in parallel. It was, however, found to be more effective to repeat the experiment in parallel for a single SNR value at a time. The early stopping criteria for belief propagation also resulted in a significant

Algorithm 1: Hamming Parallel BER Computation

Input: SNR
Output: total_errors, total_bits
Initialize: $rc = \frac{4}{7}$, total_errors = 0, total_bits = 0;
Compute $eb\_N0 = 2 \times rc \times 10^{SNR/10}$;
Compute $\sigma^2 = \frac{1}{eb\_N0}$;
Initialize lookup_table as an empty dictionary;
for $i = 0$ **to** 15 **do**
    Convert i to 4-bit binary array, *data_bits*;
    Encode *data_bits* using Hamming $(7, 4)$;
    Modulate *encoded_bits* using BPSK ;
    Store $(encoded\_bits, tx\_signal)$ in lookup_table;
end
while *total_errors* $<$ *1250* **do**
    Randomly select *data_bits* from lookup_table;
    Retrieve $(encoded\_bits, tx\_signal)$ from lookup_table;
    Pass *tx_signal* through AWGN channel with σ^2 to get *rx_signal*;
    Decode *rx_signal* using belief propagation;
    Extract original 4-bit word from *decoded_bits*;
    Compute bit errors $|$*data_bits* $-$ *decoded_data*$|$;
    Update total_errors: *total_errors* += bit_errors ;
    Update total_bits: *total_bits* += 4;
end
Return: total_errors, total_bits;

Algorithm 2: Belief Propagation Decode

Input: rx, σ^2, H, $max\_iterations$
Output: Decoded block int_bits
Initialize:
- $llr \leftarrow \frac{-4 \cdot rx}{2 \cdot \sigma^2}$
- $check\_node \leftarrow 0$;
- $code\_node \leftarrow llr$

for $iteration \leftarrow 1$ ***to*** $max\_iterations$ **do**
    foreach *check node* i **do**
        $mask \leftarrow$ Each code node connected to i
        Precompute
        $tanh\_values \leftarrow \tanh(\frac{m\_code\_to\_check[mask]}{2})$
        foreach *code node* j *connected to* i **do**
            $check\_node[i, j] \leftarrow$
            $2 \cdot \tanh^{-1}(\prod tanh\_values)$
    foreach *code node* j **do**
        $mask \leftarrow$ Each check node connected to j
        Perform the final update step;
        $code\_node[j] \leftarrow llr[j] + \sum check\_node[mask]$;
        $bits \leftarrow$ original message from $code\_nodes$;
        Check for early stopping;
        if $H \cdot bits \mod 2 == 0$ **then**
            return $bits$
        Change it to an intermediate update step;
        $code\_node[j] -= check\_node[i, j]$
return

speed-up. The maximum iterations used for belief propagation were defined as 10 [1].

Since the algorithm obtains the BER for the uncoded and Hamming codes only the Hamming codes pseudocode is given 1. For the uncoded version simply omit and encoding or belief propagation steps.

B. *Estimating GMM for time-varying model* - *Algorithm 3*

The following algorithm estimates the underlying Gaussian mixture models for use in the sequential time-varying model. It uses only the first eight years of data since the ninth is incomplete.

Algorithm 3: Fit GMMs for Time-Varying Model

Input: X, y, num_time_steps, num_bands, $num\_classes$
Output: gmm_models
Initialize:
- Split $X\_reduced$ and y into training and test sets;
- Initialize gmm_models as a list of lists of lists with dimensions $[num\_classes][num\_time\_steps][num\_bands]$, filled with `None`;

for $time\_step \leftarrow 0$ ***to*** $num\_time\_steps - 1$ **do**
    for $band \leftarrow 0$ ***to*** $num\_bands - 1$ **do**
        for $class \leftarrow 0$ ***to*** $num\_classes - 1$ **do**
            $data \leftarrow$ Current $time\_step, band,$ and class;
            $gmm \leftarrow$ Initialize and fit a GMM on $data$;
            $gmm\_models[class][time\_step][band] \leftarrow gmm$;
        end
    end
end
Return: gmm_models;

C. *SPRT Test with Time-varying Model* - *Algorithm 4*

The following algorithm shows a general procedure to classify a single MODIS pixel using each of the 8 bands. The previously estimated GMMs are utilised within the algorithm to calculated the log-likelihood ratio within a given season.

D. *Radix-2 Decimation-in-Time FFT* - *Algorithm 5:*

- Recursively splits input into even and odd parts.
- Computes FFT on each part and combines results using twiddle factors.
- Efficient for inputs of size 2^n, with complexity $O(N \log_2 N)$.

E. *Generic FFT for Any Length Input* - *Algorithm 6*

- Handles arbitrary input sizes.
- Uses Radix-2 DIT FFT if input length is divisible by 2.
- Falls back to matrix-based DFT for non-power-of-2 inputs.

Algorithm 4: Classify Pixel Using SPRT

Input: $pixel\_data$, gmm_veg, gmm_sett, num_bands, num_time_steps
Output: classification

Initialize: $llr\_per\_band \leftarrow$ a list of empty lists with length num_bands;
for $time\_step \leftarrow 0$ ***to*** $num\_time\_steps - 1$ **do**
  for $band \leftarrow 0$ ***to*** $num\_bands - 1$ **do**
    Get Models and Observation:
    $gmm\_veg \leftarrow gmm\_veg[time\_step][band]$;
    $gmm\_sett \leftarrow gmm\_sett[time\_step][band]$;
    $observation \leftarrow pixel\_data[time\_step, band]$;
    Calculate Log-Likelihood Ratios:
    $llr\_veg \leftarrow gmm\_veg.\texttt{score\_samples}(observation)[0]$;
    $llr\_sett \leftarrow gmm\_sett.\texttt{score\_samples}(observation)[0]$;
    $llr \leftarrow llr\_veg - llr\_sett$;
    Append llr to $llr\_per\_band[band]$;
  end
end
Decision Making:
- $final\_llr \leftarrow$ Accumulate llr_per_band over time for each band;
- **Classify:** $classification \leftarrow 1$ if $final\_llr > 0$ else 0;

Return: $classification$;

Algorithm 5: Radix-2 Decimation-in-Time FFT

Input: x (input array of size 2^n)
Output: FFT of x

Function ditrad2(x):
- $N \leftarrow$ len(x)
- Convert x to complex type
- **if** $N == 1$ **then**
  return x
- Split x into even and odd parts:
  - $even \leftarrow ditrad2(x[:: 2])$
  - $odd \leftarrow ditrad2(x[1 :: 2])$
- Pre-allocate result array
- **for** $k = 0$ **to** $N/2 - 1$ **do**
  Compute $twiddle\_factor = e^{-2\pi ik/N} \times odd[k]$
  $result[k] \leftarrow even[k] + twiddle\_factor$
  $result[k + N/2] \leftarrow even[k] - twiddle\_factor$
- **return** result

Algorithm 6: Generic FFT for Any Length Input

Input: x (input array of arbitrary length)
Output: FFT of x

Function generalFFT(x):
- Convert x to complex type
- $N \leftarrow$ len(x)
- **if** $N == 1$ **then**
  return x
- **if** N *is divisible by 2* **then**
  return ditrad2(x)
- Else, perform matrix-based DFT:
  - Compute k and n as indices
  - Compute $exponent \leftarrow e^{-2\pi ikn/N}$
  - **return** $exponent \cdot x$

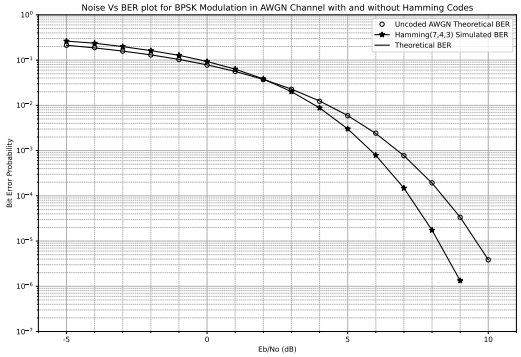


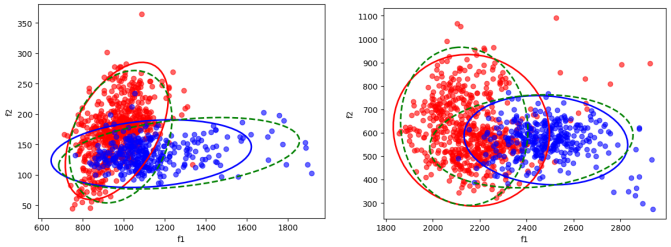
Fig. 1: Bit error probability at different noise levels

V. Figures

Figure 1 represents the bit error rates at different levels of noise. The simulated BER for uncoded bits match the theoretical BER extremely well. Repeating the experiment until a total of 10,000 errors had occurred was clearly sufficient. The Hamming codes performed better than the uncoded AWGN at low levels of noise. At high levels of noise, the Hamming codes start performing worse than the uncoded bits. Hamming codes can only correct single-bit errors [5]. When multiple bit errors occur the Hamming codes can not correct it and the belief propagation algorithm does not work.

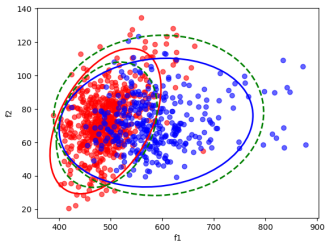
Figure 2 represents the feature space after applying dimensionality reduction to obtain the harmonic features: mean and seasonality. Gaussian densities were also fitted and appear to accurately encapsulate the respective classes. Components obtained from the two-component GMM, consider the dataset as unlabeled and are thus not as accurate. We see that certain bands are more separable than others, indicated by the visual scatter and the overlap between Gaussian and GMM confidence intervals, which should be reflected in the classification accuracies.

Figure 3 indicates the classification accuracy obtained for each model in each band when applying the techniques to

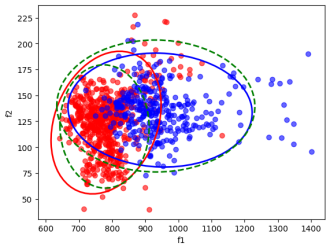


(a) Band 1

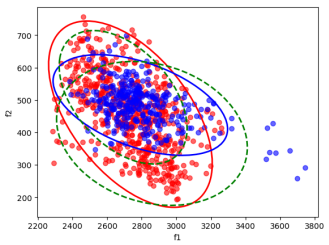
(b) Band 2



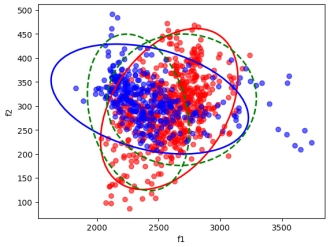
(c) Band 3



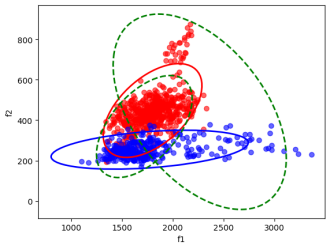
(d) Band 4



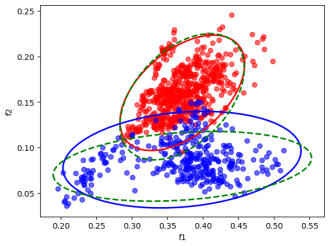
(e) Band 5



(f) Band 6



(g) Band 7



(h) NDVI

Fig. 2: The reduced feature space obtained after running FFT on the dataset. The mean of each pixel is indicated on the y-axis and the seasonal component on the x-axis. The features are plotted with corresponding labels (blue for settlements, red for vegetation). The ellipses are 95% confidence intervals for both Gaussian densities (solid line) and GMM mixture components (dashed line).

a single data split of 50% training and 50% testing. In most cases, the Logistic Regression and Naïve Bayes models that were applied to the reduced dataset, consisting of the two harmonic components, outperform the SPRT time-varying model applied to the full feature-space. It is also clear that certain bands yield better classification scores. Particularly NDVI and band 7 report the highest scores, which is supported by the class separability of these bands indicated by figure 2. Furthermore, the classification accuracies for these bands are high ($> 90\%$) indicating that the problem of land cover

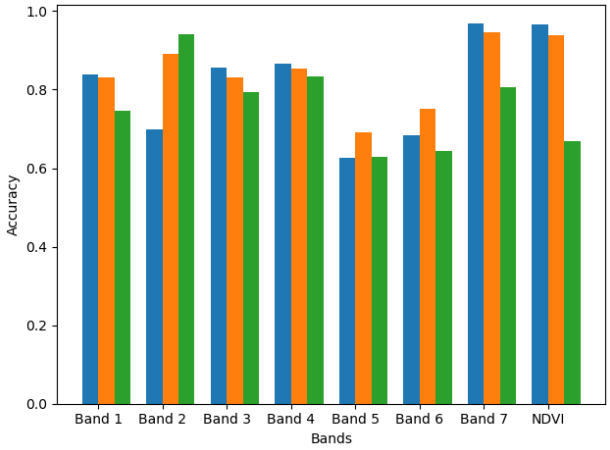


Fig. 3: The per-band classification accuracy obtained on the test data for Logistic Regression (blue), Naïve Bayes (Orange), and SPRT (Green).

classification is successfully addressed.

Table I contains the average classification accuracy and standard deviation for each technique taken over 20 runs. In each run, a random seed is generated and applied during the splitting process for both the reduced and full feature space. Thus ensuring that all techniques are trained and tested on the same permutation of observations. Again, the data is split into 50% for testing and 50% for training. The SPRT time-varying model reported the worst performance in all bands except for band 2, where it stood out as the best classifier with an accuracy of 93.9%, performing markedly better than Logistic Regression. Naïve Bayes displayed the best overall performance across all bands with an accuracy of 84.78%, which is slightly better than that of logistic regression. Returning to figure 2 this fact is apparent given the large overlap between classes making it difficult to separate land cover types with a linear boundary. However, logistic regression reported the highest performance across all techniques and bands, achieving an accuracy of 96.93% and a standard deviation of only 0.5% in the NDVI band. Thus, the techniques applied to the reduced dataset outperform the time-varying sequential model in almost every band. This indicates that seasonality and trend are well summarised by harmonic features. Furthermore, the use of logistic regression as a linear classifier achieves an average accuracy of over 96% in the NDVI band indicating effective classification of land-cover types.

The performance of several FFT and DFT implementations is compared in table II. Due to its $O(N^2)$ complexity, the Double Loop DFT is the slowest, lasting more than 73 milliseconds. With vectored operations, the Matrix DFT greatly enhances performance; however, the One Layer FFT, which makes use of the FFT structure, demonstrates even higher efficiency. The custom Radix-2 FFT improves the speed even further, getting close to the **numpy** FFT's performance, which is the fastest because of its optimized libraries. Power-of-2 and non-power-of-2 inputs are handled by the General FFT with

Band	Logistic Regression Mean	Logistic Regression Std	Naive Bayes Mean	Naive Bayes Std	SPRT Mean	SPRT Std
1	0.839849	0.012837	0.838013	0.010907	0.740173	0.016048
2	0.786393	0.108396	0.893413	0.009928	0.939093	0.011542
3	0.845896	0.013424	0.822246	0.015932	0.789957	0.013312
4	0.850216	0.012242	0.845464	0.014656	0.825378	0.013424
5	0.632937	0.011859	0.659503	0.014843	0.630994	0.015949
6	0.705832	0.021841	0.753780	0.012741	0.639741	0.018749
7	0.959827	0.008623	0.948380	0.006758	0.763175	0.031935
NDVI	0.969330	0.005228	0.941577	0.007651	0.656371	0.033785
Average	0.823785	0.024306	0.837797	0.011677	0.748110	0.019343

TABLE I: Contains averages for accuracy and standard deviation across 20 runs for three classification models: Logistic Regression, Naïve Bayes, SPRT. It also displays a final average across all bands for each classifier.

Method	Time per loop (µs)	Std. Dev. (µs)
Double Loop DFT	73,300	1,440
Matrix DFT	1,580	29.9
One Layer FFT + Matrix DFT	892	36.2
numpy FFT	7.88	0.151
Custom FFT	48.4	3.2
General FFT (Power of 2 input)	1,520	33.6
General FFT (Non-power of 2 input)	72.7	4.29

TABLE II: Performance Comparison of FFT and DFT Implementations

minor reduction in performance for the latter.

VI. Conclusion

This study applied several classification and signal processing techniques to MODIS time-series data, focusing on distinguishing between vegetation and settlement land cover types. The use of belief propagation for error correction, in combination with Hamming codes, showcased a robust approach to handling noise in data transmission, significantly improving the reliability of classification results, especially in noisy environments.

Additionally, Fast Fourier Transform (FFT) was instrumental in reducing the dimensionality of the dataset. By converting the time-series data to the frequency domain, FFT effectively captured periodic patterns and allowed for improved classification accuracy, particularly in the NDVI and certain spectral bands. This transformation facilitated the extraction of meaningful harmonic components, leading to more efficient and accurate land-cover classification when applied to Logistic Regression and Naive Bayes models.

Comparatively, the SPRT model, although effective in capturing time-varying patterns, underperformed relative to the reduced data approach. The FFT-based dimensionality reduction, coupled with traditional classifiers, provided a faster and more scalable solution for processing the large MODIS dataset.

The findings highlight the practical utility of combining FFT for data reduction and belief propagation for error correction in the classification of environmental data. This approach has valuable applications in monitoring settlement expansion and supporting sustainable development efforts. Future work could explore further optimizations in belief propagation and additional refinements in FFT processing for even greater accuracy and efficiency.

References

[1] T. L. Grobler, "Fountain Codes and Their Typical Application in Wireless Standards Like EDGE," M.Eng. thesis, Dept. of Electrical, Electronic and Computer Engineering, Univ. of Pretoria, Pretoria, South Africa, 2008.

[2] T. L. Grobler, W. Kleynhans and B. P. Salmon, "Empirically Comparing Two Dimensionality Reduction Techniques – PCA and FFT: A Settlement Detection Case Study in the Gauteng Province of South Africa," IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium , Yokohama, Japan, 2019, pp. 3329-3332.

[3] T. L. Grobler, W. Kleynhans and B. P. Salmon, "Sequential Classification of MODIS time-series," IGARSS 2012 - 2012 IEEE International Geoscience and Remote Sensing Symposium , Munich, Germany, 2012, pp. 6236-6239.

[4] O. Gazi, "Syndrome Decoding and Some Important Linear Block Codes," in *Forward Error Correction via Channel Coding*, Springer, Cham, 2020, pp. 57–89.

[5] R. W. Hamming, "Error detecting and error correcting codes," in The Bell System Technical Journal, vol. 29, no. 2, pp. 147-160, April 1950, doi: 10.1002/j.1538-7305.1950.tb00463.x.

[6] T.L. Grobler, "Sequential and Non-sequential Hypertemporal Classification and Change Detection of MODIS Time-series", Ph.D. dissertation, Dept. Electrical, Electronic and Computer Engineering, Uni. Pretoria, Pretoria, 2012

[7] E. Ackermann, "Sequential Land Cover Classification", M.S. Thesis, Dept. Electrical, Electronic and Computer Engineering, Uni. Pretoria, Pretoria, 2011

[8] T. Thomas, P. Michael, "Signal Processing Methods for Harmonic Analysis", *Engineering Reports*, Jan 2020, available: https://www.authorea.com/users/294862/articles/423318-signal-processing-methods-for-harmonic-analysis

[9] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning with Applications in R*. New York: Springer, 2013.

[10] Douglas L. Jones, *Digital Signal Processing: A User's Guide*. OpenStax CNX 2022.