# CLASSIFICATION OF SATELLITE IMAGES BASED ON THEIR CLASS OF TERRAIN

**David G. Shatwell**
Department of Electrical Engineering
University of Engineering and Technology–UTEC
Lima 15063, Peru
david.shatwell@utec.edu.pe

**Alejandro Weston**
Department of Mechanical Engineering
University of Engineering and Technology–UTEC
Lima 15063, Peru
alejandro.weston@utec.edu.pe

**Oscar Ramos**
Department of Mechatronics Engineering
University of Engineering and Technology–UTEC
Lima 15063, Peru
oramos@utec.edu.pe

**ABSTRACT**

Satellite image classification and analysis has many important applications in the real world, such as tracking deforestation and desertification levels, glacier movement, and even urban expansion. In this paper, we propose a satellite image analysis algorithm based on sub-image classification. The proposed algorithm can identify up to five different classes of terrain: city, sand, vegetation, mountain and water. In order to find the best possible image analysis algorithm, we trained and compared the classification performance of four common machine learning algorithms that require manual feature extraction and one deep learning algorithm with automatic feature extraction. All sub-images used to train and test the models were acquired from Google Earth at different heights. The models were tested using sub-images that come from the same images as the training set and also with sub-images that come from other images not used for training. After evaluating the performance of all models, the best one was selected for the task of satellite image analysis.

***Keywords*** Satellite images · Terrain classification · Color and texture analysis · Deep learning

## 1 Introduction

Satellite image analysis consists of using a set of techniques, usually based on image processing and machine learning algorithms, used to extract useful information from the image. For example, in the past, satellite image analysis has been used to track deforestation levels using econometric techniques [Nelson and Hellerstein, 1997], the Haar wavelet transform [Menaka et al., 2013], multispectral imaging [Lu et al., 2017], and deep learning [Bragilevsky and Bajić, 2017, Irvin et al., 2020, Lee et al., 2020]. Glacier size and movement tracking have also benefited from satellite image analysis. For example, Scambos et al. [1992] used image cross-correlation to estimate glacier velocity with sub-pixel accuracy, while Baumann et al. [2021] used spectral band ratios and maximum likelihood classification to update the inventory of glacier ice in New Zealand. Lastly, urban expansion has also been analyzed by employing satellite images with great success. Long et al. [2018] evaluated the impact of high-speed rails on the development of cities using nighttime light satellite images, while Boulila et al. [2021] proposed a novel CNN-LSTM-based approach to predict urban expansion.

Most of the satellite image analysis methods employed in the literature use at least one of following techniques: (i) manual classification, (ii) automatic classification based on multispectral imaging, (iii) automatic classification based on feature extraction and classic machine learning algorithms, or (iv) automatic classification based on deep learning. From these four techniques, only (iii) and (iv) can be applied to images acquired by color cameras, which are are usually

Table 1: Number of sub-images per terrain class

| Terrain class | Number of sub-images |
|---|---|
| Vegetation | 20,501 |
| City | 24,832 |
| Sand | 18,098 |
| Mountain | 19,155 |
| Water | 24,037 |
| Total | 105,804 |

cheaper, easier to operate, and posses a higher spatial resolution than hyperspectral, thermal, or other more specialized types of cameras.

In the last decades, manual feature extraction algorithms have been used extensively to represent high-dimensional data structures, such as color or gray-scale images, as smaller feature vectors that contain only the most most important characteristics of the object being represented. A common approach to produce feature vectors is to extract color and texture features from the images. Color features can be computed using the channel statistics, such as the mean and variance of each color channel or multiway principal component analysis (MPCA) [Nomikos and MacGregor, 1994]. More than one color space can be used to increase the amount of features and potentially improve classification accuracy. Similarly, several texture feature extraction methods have been developed in the past. Haralick et al. [1973] proposed a set of 14 second-order statistical parameters computed from the gray-level coocurrance matrix (GLCM). Haralick texture features are used to describe the relationship between the intensity of adjacent pixels, which is associated to the texture of images. Other common methods include Gabor filters [Turner, 1986], wavelet texture analysis [Chang and Kuo, 1993], and multiscale AM–FM texture analysis [Loizou et al., 2010].

Manual feature extraction is usually followed by a classification stage, where a predictive model is used to map the feature vectors to one or more classification values, depending on the algorithm and number of classes. Some of the most well-known classification algorithms are support vector machines (SVM), multilayer perceptrons (MLP), random forests (RF), and naive Bayes (NB). While classic machine learning algorithms have produced excellent results over the years in many classification tasks, more recently, deep learning algorithm have become more prominent. For example, convolutional neural networks (CNN) are part of a family of deep learning algorithms that employ convolutions with trainable kernels in order to generate feature vectors [LeCun et al., 1995]. The advantage of CNNs over classic machine learning algorithms is that they don't require manual manual feature engineering, thus, reducing development time. Also, the convolution layers in the CNNs may produce features that more accurately represent the images than those produced using manual feature engineering, which is a highly desirable trait in classification systems.

In this paper, we propose a satellite image analysis algorithm based on terrain classification of sub-images. We compare the performance of four different algorithms using manual feature extraction, and one classification algorithm based on deep learning. All algorithms were trained and tested using the same dataset, which consists of satellite images obtained from Google Earth Pro at different heights. Additionally, we performed tests with sub-images from different images than the ones used for training and that contain multiple classes of terrain. By doing this, it was possible to determine which classification algorithm has better generalization capabilities.

## 2 Dataset

Google Earth Pro was used to acquire images of five different terrain types: city, sand, vegetation, mountain and water. The images used JPEG lossy compression and have dimensions of $4,800 \times 2,823$ pixels. Each image was subdivided into 814 square sub-images with side length of 128 pixels. In total, the dataset contained $106,623$ sub-images of the five terrain classes, where 80% were used for training and validation, and the remaining 20% were used for testing (test set 1). The number of sub-images per class is shown on table 1. Additionally, the models were also tested using five new satellite images with multiple terrain classes (test set 2). The new images contain the same terrain classes found on the original train and test sets. However, because the new sub-images come from different images as the ones used for training, the distribution of their features is slightly different. In total, test set 2 contains 4,070 sub-images.

## 3 Methods

The proposed satellite image analysis algorithm consists of three main stages: (i) sub-image partition, which was covered in section 2, (ii) feature extraction, and (iii) classification. Classic machine learning algorithms usually need a feature extraction stage in order to reduce the number of dimensions of the original images, which can be done by extracting color and texture features. Color features are computed using channel statistics and multiway principal component analysis, while texture features are computed using Haralick features applied over the gray-level co-occurrence matrix. In contrast, when using the convolutional neural network as the classification algorithm, the filter coefficients of the convolutional layers are found when training the algorithm. In the classification stage, five different models are trained and used to classify the sub-images: support vector machines (SVM), multi-layer perceptrons (MLP), random forests (RF), naive Bayes (NB), and convolutional neural networks (CNN). The complete classification algorithm is shown on figure 1.

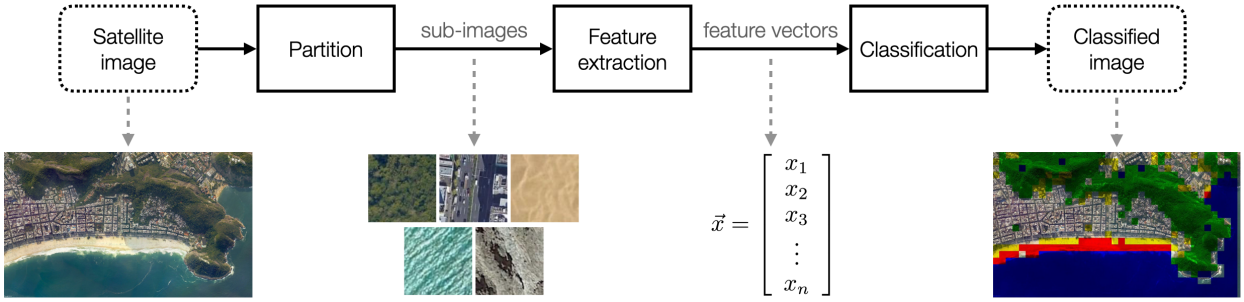


Figure 1: Block diagram of the proposed algorithm, which consists of image partitioning, feature extraction and classification stages.

### 3.1 Color features

Before computing the color and texture features, each sub-image is transformed from the red-green-blue (RGB) to the hue-saturation-value (HSV) color space. Using two color spaces instead of only one is a straightforward way to produce more features, which has the potential to improve classification accuracy. The HSV color space is commonly used in computer vision applications because it describes color in a way similar to how humans interpret it. Also, the hue channel is invariant to changes in lightning, which is highly desirable in many image classification problems. After color transformation, the algorithm computes first-order statistical features from both color spaces, particularly, the mean, variance, skewness and kurtosis. Because there are three color channels in each color space, the algorithm computes 24 channel statistics in total.

The second set of color features is computed with the multiway principal component analysis (MPCA) algorithm. MPCA is similar to the classic PCA algorithm, but uses a pre-processing step that transforms the color image into a single matrix. In MPCA, each pixel, which has three intensity values corresponding to the three color channels, is treated as one row of the data matrix $\mathrm{X_{PCA}}$. If the original sub-image $S$ has $M$ rows, $N$ columns, and 3 channels, then $\mathrm{X_{PCA}}$ has $M \times N$ rows and 3 columns, as shown on figure 2.

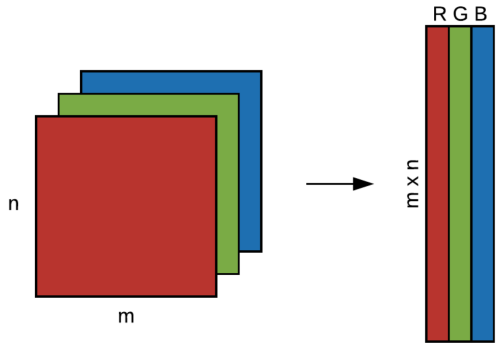


Figure 2: Graphical representation of multiway principal component analysis (MPCA)

After forming the matrix $X_{PCA}$, PCA is computed by finding the eigenvectors of its covariance matrix. The result is a set of three eigenvectors, ordered from highest to lowest eigenvalue, that are orthogonal and represent the directions of highest variance in the pixels. However, because the set composed of the first and second eigenvectors usually usually explain over 99% of the variance, as shown on Tessier et al. [2007], Geladi and Grahn [2006], and Bharati and MacGregor [1998], the third eigenvector can be discarded without significant losses. The first and second eigenvectors produce, in total, six new color features, because each of them consists of three elements. Unlike the color statistics, which provide features of each channel individually, the eigenvectors can be interpreted as overall color descriptors of an image [Yu and MacGregor, 2003]. Thus, the combination of both sets of features provides a complete representation of the color of the image.

### 3.2 Texture features

The visual texture of an image is defined as the local variations in intensity that are generated by the roughness or unevenness in the surface of an object [Mirmehdi, 2008]. In other words, when light strikes a rough surface, it scatters randomly, increasing or decreasing the local reflectance of the surface in the viewing direction. For a well-defined texture, the intensity changes are usually regularly distributed with random variations, which means that the local patterns in the image may have definite characteristics of periodicity, directionality or orientation.

One of the most widely used methods to quantify the roughness of a texture is the gray-scale co-occurrence matrix (GLCM), which is based on computing second-order statistical parameters of the distribution of pixel intensities. Rosenfeld and Troy [1970] showed that first order statistics alone are not good enough descriptors for texture classification. Therefore, Haralick et al. [1973] proposed the use of second order statistics, computed by considering the spatial relationship between pairs of pixels. More formally, the GLCM is used to express the probability $P(i, j|d, \theta)$ that two given pixels with relative polar coordinates $(d, \theta)$ have the intensity values $(i, j)$. For an image with $p$ gray levels, the dimensions of the GLCM are $p \times p$. Although digital images typically have 256 gray levels, they usually are quantized into a reduced number of levels in order to decrease the processing time of the algorithm without altering the fundamental texture structures.

Haralick texture features consists of a set of 14 descriptors computed from the co-occurrence matrix and are used to obtain information about the texture of a digital image. In practice only the first 13 features are used, which are: angular second momentum, contrast, correlation, variance, inverse difference moment, sum average, sum variance, sum entropy, entropy, difference variance, difference entropy, information measure of correlation 1 and information measure of correlation 2. Details about how to compute each feature can be found on Haralick et al. [1973].

### 3.3 Classification from color and texture features

After extracting the color and texture features from the sub-images, a classification model is needed to map the feature vectors into classification values. In this paper, we trained four different classification models using cross-validation with the same training and test sets. The first model consists of ten one-vs-one SVMs, where each one of them specializes in the binary classification of two terrain classes. The best set of hyperparameters that were found for the SVMs were a regularization parameter $C = 10$ and a RBF kernel with coefficient $\gamma = 0.0233$. The second model is a MLP, composed of one hidden layer with 256 neurons. The neurons in the hidden layer used the ReLU activation function, while the output layer uses the sigmoid function. The network was trained for 500 epochs, using the Adam optimizer and a regularization parameter $\alpha = 0.0001$. The third model is a random forest classifier with 100 trees that uses the Gini impurity to measure the quality of the splits. Lastly, the fourth model is a naive Bayes classifier, which doesn't need hyperparameter tuning.

### 3.4 Classification using convolutional neural network

Unlike the algorithms mentioned previously in sections 3.1, 3.2, and 3.3, CNNs don't require a manual feature extraction stage. Instead, CNNs use convolutional layers, which are made up of filter banks with trainable coefficients. This means that the network is able to find the optimal set of features given a certain dataset and network architecture. Convolutional layers provide three main advantages over MLPs: sparse interactions, parameter sharing, and equivariant representations [Goodfellow et al., 2016], all of which are important properties that may improve the system. The second type of layers found on a CNN are pooling layers, which perform windowed statistical operations on the output of the convolutional layers. The most common type of pooling is max-pooling, which simply takes the maximum value of the window. Pooling is often used in CNNs because it makes the representation become approximately invariant to small translations of the input, allows the CNN to handle of inputs of vaying sizes, and improve the computation efficiency of the network [Goodfellow et al., 2016]. Lastly, the output of the convolutional and pooling layers is flattened into a feature vector, which then is used as input to a MLP. The MLP learns to map the feature vector into one or more classification or

Table 2: Classification performance of the five models using sub-images extracted from the same distribution as the train set (test set 1). The models are evaluated under using the $F_1$-score per terrain class, the weighted average (WA) $F_1$-score, and the accuracy.

| | $F_1$-score | | | | | | |
|---|---|---|---|---|---|---|---|
| Model | City | Sand | Vegetation | Mountain | Water | WA $F_1$-score | Accuracy |
| SVM | 0.97 | 0.99 | 0.98 | 0.98 | 0.99 | 0.98 | 0.98 |
| MLP | 0.97 | 0.99 | 0.97 | 0.97 | 0.99 | 0.98 | 0.98 |
| RF | 0.96 | 0.99 | 0.97 | 0.96 | 0.99 | 0.98 | 0.98 |
| NB | 0.84 | 0.91 | 0.77 | 0.64 | 0.80 | 0.79 | 0.80 |
| CNN | 0.97 | 0.98 | 0.96 | 0.94 | 0.99 | 0.97 | 0.97 |

prediction values, depending on the amount of neurons in the output layer and their activation function. The architecture of the CNN used in this paper is shown on figure 3.

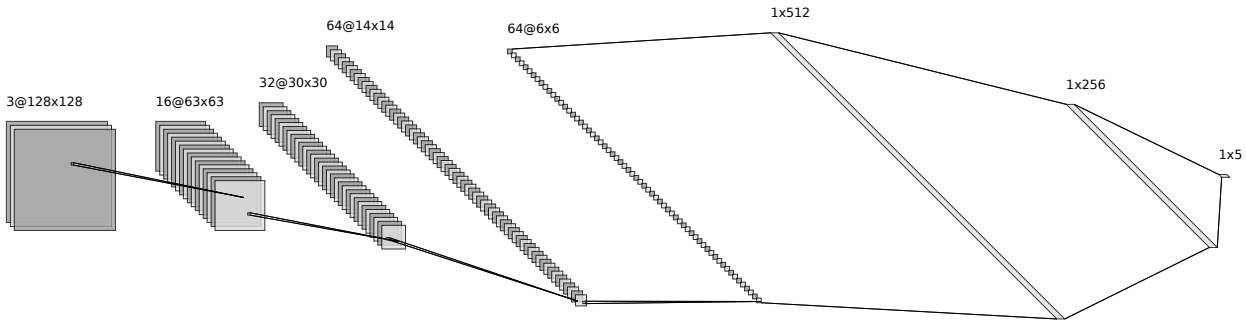


Figure 3: CNN architecture. The original image has a size of $128 \times 128 \times 3$. Between each feature map, convolution and max pooling operations are applied. All filters have a kernel size of $3 \times 3$ and all max pool operators have a kernel size of $2 \times 2$. The last three layers are fully connected: the first one has 512 neurons and the second one 256. The output layer has 5 neurons with the softmax activation function.

## 4 Results

The complete dataset used for training and testing consists of 105,804 sub-images. Form this dataset, 84,643 sub-images (80%) were used for training the model and 21,160 (20%) were used for testing. Additionally, all models were trained using 10-fold cross-validation in order to find the best set of hyperparameters before testing. Then, after the best hyperparameters are found, the model is trained again using the 105,804 sub-images. The performance of the models was evaluated using the F1-score for each class of terrain, the weighted average F1-score for all classes, and the accuracy, which are shown on table 2. It is worth mentioning that the train and test set sub-images are extracted from the same set of complete images. This means that, although the train and test set sub-images are different, they are both sampled from the same distribution.

The best model for each class of terrain can be found by analyzing the individual $F_1$-scores. For sub-images containing cities, the best models are the SVM, NN, and CNN (0.97). Then, for sand sub-images, the best models are the SVM, NN and RF (0.99). In the case of vegetation and mountain sub-images, the SVM outperforms all other models, with a $F_1$-score of 0.98. Lastly, in the case of water sub-images, all models except for naive Bayes performed equally, with a $F_1$-score of 0.99. From these results, it is clear that the SVM is the most consistent model across all classes. The weighted average $F_1$-score and accuracy values also show that the SVM is the best model, tied with the MLP and RF classifiers (0.98). Interestingly, the CNN occupies fourth place, with a $F_1$-score and accuracy of 0.97. The worst model was the naive Bayes classifier, with a weighted average $F_1$-score of 0.79 and accuracy of 0.8.

After evaluating the performance of the models using sub-images that come from the same distribution as the training set, five new images were used to perform additional tests. As mentioned in section 2, these new images contain the same terrain classes as the ones from the training set. However, because they are completely new images, the distribution of the terrain is slightly different than the one from the training set. All of the new test images contain multiple terrain classes. Their sub-images were manually classified using a segmentation app developed specifically for

Table 3: Classification performance of the five models using sub-images extracted from different images as the train set (test set 2). The models are evaluated under using the $F_1$-score per terrain class, the weighted average (WA) $F_1$-score, and the accuracy.

| | $F_1$-score | | | | | | |
|---|---|---|---|---|---|---|---|
| Model | City | Sand | Vegetation | Mountain | Water | WA $F_1$-score | Accuracy |
| SVM | 0.92 | 0.64 | 0.84 | 0.84 | 0.93 | 0.90 | 0.90 |
| MLP | 0.90 | 0.62 | 0.80 | 0.85 | 0.92 | 0.88 | 0.88 |
| RF | 0.92 | 0.66 | 0.79 | 0.79 | 0.92 | 0.88 | 0.89 |
| NB | 0.91 | 0.43 | 0.69 | 0.40 | 0.93 | 0.83 | 0.83 |
| CNN | 0.94 | 0.76 | 0.87 | 0.80 | 0.96 | 0.92 | 0.92 |

this purpose. The app classifies a sub-image if at least 50% of its pixels belong to one specific class. If the sub-image contains multiple classes, then it takes the class with the most amount of pixels. This way, we reduced the bias of assigning classes to the sub-images. Table 3 shows the classification performance obtained for each classifier for the sub-images on the new test set.

Unlike the results using sub-images from the same distribution as the training set, the CNN obtained the best overall performance of all classifiers, with an accuracy and weighted average $F_1$-score of 0.92. The CNN was followed by the SVM (0.9), random forest (accuracy of 0.89 and weighted average $F_1$-score of 0.88), MLP (0.88), and naive Bayes (0.83). In terms of performance per class, the CNN achieved the best results for the city (0.92), sand (0.76), vegetation (0.87), and water (0.96) sub-images, while the SVM achieved the best performance for the mountain sub-images. The results of this test are interesting because the CNN appears to have the best generalization capabilities, since the performance between both tests decreases the least of all models.

Additionally, we analyzed the parts of the test images that were being classified incorrectly. The most obvious errors, as shown on figure 4, occur on sub-images that share two classes of terrain. This is specially noticeable on beaches, where a large portion of sub-images are being classified as either city or mountain. This explains the decrease in the $F_1$-score that is present in all classifiers. The two worst cases are the naive Bayes and MLP, with dips in performance of 0.48 and 0.37 points respectively. The CNN, which is the best overall classifier, also has similar problems in some images. For example, on the third image, some mountain sub-images were incorrectly classified as vegetation and on the fourth image, a large portion of the sand sub-images were classified as city. A possible solution to these kinds of problems would be to create sub-image categories with multiple classes. For example, some classes could be city-sand, city-water, vegetation-mountain, etc. The problem with this approach is that the number of possible classes would increase from $n$ to $n + \binom{n}{2}$ classes. However, the number of classes could be reduced by only creating new classes for the most common types of terrain intersections.

## 5 Conclusions

In this paper, we presented a satellite image classification analysis algorithm based on sub-image classification. The algorithm consists of three main stages: (i) sub-image partition, (ii) feature extraction, and (iii) sub-image classification. Sub-image classification allows the algorithm to estimate the proportion of each class of terrain in the image, taking advantage of local color and texture. In the feature extraction stage, we used manual and automatic feature extraction methods. Manual feature extraction was performed using image analysis algorithms, such as channel statistics, principal component analysis, and the gray-level co-occurrence matrix, while automatic feature extraction was performed using convolutional neural networks. Lastly, in the sub-image classification stage, we compared the performance of four classic machine learning models (support vector machines, multi-layer perceptrons, random forest, and naive Bayes) and one deep learning model (convolutional neural networks).

The models were trained using 84,643 sub-images from five terrain classes: city, desert, vegetation, mountain, and water. Then, the models were tested with 21,161 sub-images coming from the same images as the training set (test set 1) and 4,070 sub-images coming from different images (test set 2) under the weighted average $F_1$-score metric. The support vector machine model achieved the best overall performance of all models when evaluated using test set 1 (0.98), while the convolutional neural network achieved the best performance on test set 2 (0.92). This means that the convolutional neural networks have better generalization capabilities than the other models, and are an preferred choice for the satellite image analysis problem.

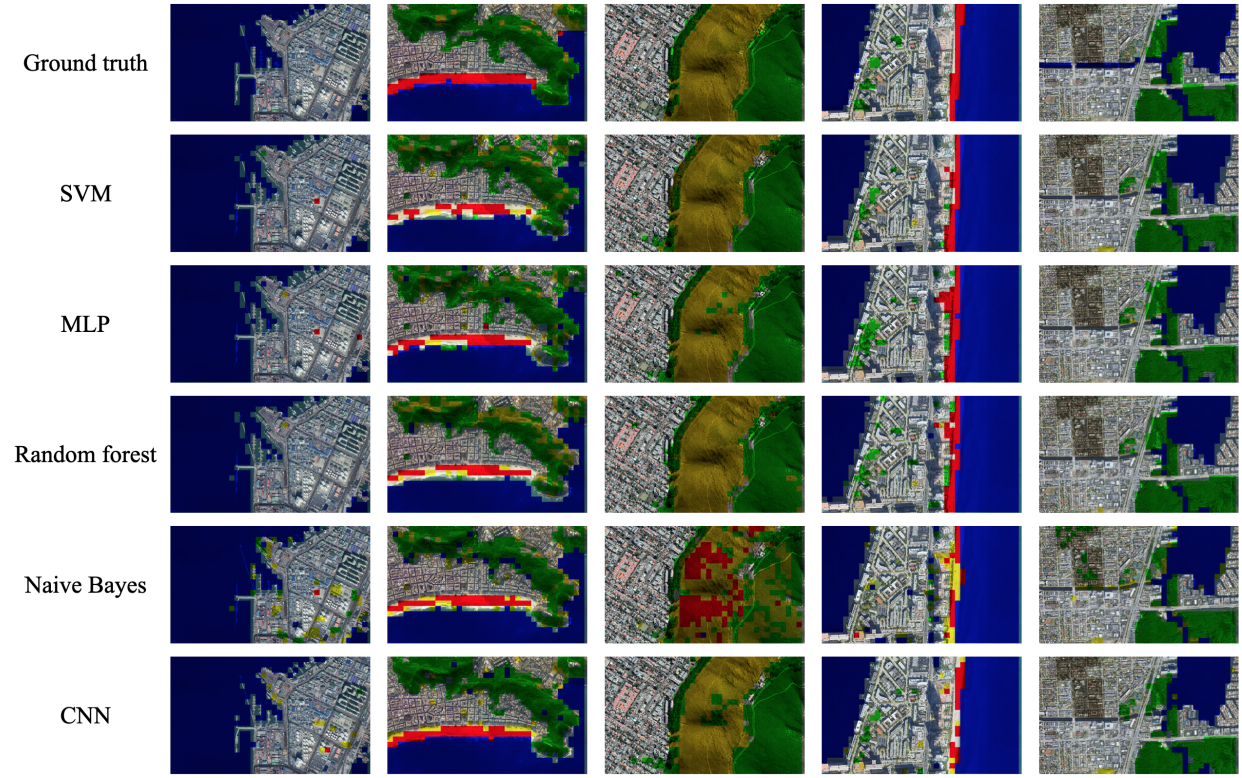


Figure 4: Classification maps of the images of test set 2, whose sub-images are not used for training the models. The first row shows the ground-truth classification maps, which are created by segmenting the images manually. Each subsequent row shows the classification maps obtained using a specific classifier. The following color coding is used to represent the class of a sub-image: city-transparent, sand-red, mountain-yellow, vegetation-green, and water-blue.

## References

Gerald C Nelson and Daniel Hellerstein. Do roads cause deforestation? using satellite images in econometric analysis of land use. *American Journal of Agricultural Economics*, 79(1):80–88, 1997.

E Menaka, S Suresh Kumar, and M Bharathi. Change detection in deforestation using high resolution satellite image with haar wavelet transforms. In *2013 International Conference on Green High Performance Computing (ICGHPC)*, pages 1–7. IEEE, 2013.

Meng Lu, Eliakim Hamunyela, Jan Verbesselt, and Edzer Pebesma. Dimension reduction of multi-spectral satellite image time series to improve deforestation monitoring. *Remote Sensing*, 9(10):1025, 2017.

Lior Bragilevsky and Ivan V Bajić. Deep learning for amazon satellite image analysis. In *2017 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, pages 1–5. IEEE, 2017.

Jeremy Irvin, Hao Sheng, Neel Ramachandran, Sonja Johnson-Yu, Sharon Zhou, Kyle Story, Rose Rustowicz, Cooper Elsworth, Kemen Austin, and Andrew Y Ng. Forestnet: Classifying drivers of deforestation in indonesia using deep learning on satellite imagery. *arXiv preprint arXiv:2011.05479*, 2020.

Seong-Hyeok Lee, Kuk-Jin Han, Kwon Lee, Kwang-Jae Lee, Kwan-Young Oh, and Moung-Jin Lee. Classification of landscape affected by deforestation using high-resolution remote sensing data and deep-learning techniques. *Remote Sensing*, 12(20):3372, 2020.

Theodore A Scambos, Melanie J Dutkiewicz, Jeremy C Wilson, and Robert A Bindschadler. Application of image cross-correlation to the measurement of glacier velocity using satellite image data. *Remote sensing of environment*, 42(3):177–186, 1992.

Sabine Baumann, Brian Anderson, Trevor Chinn, Andrew Mackintosh, Catherine Collier, Andrew M Lorrey, Wolfgang Rack, Heather Purdie, and Shaun Eaves. Updated inventory of glacier ice in new zealand based on 2016 satellite imagery. *Journal of Glaciology*, 67(261):13–26, 2021.

Fenjie Long, Longfei Zheng, and Zhida Song. High-speed rail and urban expansion: An empirical study using a time series of nighttime light satellite data in china. *Journal of Transport Geography*, 72:106–118, 2018.

Wadii Boulila, Hamza Ghandorh, Mehshan Ahmed Khan, Fawad Ahmed, and Jawad Ahmad. A novel cnn-lstm-based approach to predict urban expansion. *Ecological Informatics*, 64:101325, 2021.

Paul Nomikos and John F MacGregor. Monitoring batch processes using multiway principal component analysis. *AIChE Journal*, 40(8):1361–1375, 1994.

Robert M Haralick, Karthikeyan Shanmugam, and Its' Hak Dinstein. Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, (6):610–621, 1973.

Mark R Turner. Texture discrimination by gabor functions. *Biological cybernetics*, 55(2):71–82, 1986.

Tianhorng Chang and C-CJ Kuo. Texture analysis and classification with tree-structured wavelet transform. *IEEE Transactions on image processing*, 2(4):429–441, 1993.

Christos P Loizou, Víctor Murray, Marios S Pattichis, Ioannis Seimenis, Marios Pantziaris, and Constantinos S Pattichis. Multiscale amplitude-modulation frequency-modulation (am–fm) texture analysis of multiple sclerosis in brain mri images. *IEEE Transactions on Information Technology in Biomedicine*, 15(1):119–129, 2010.

Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

Jayson Tessier, Carl Duchesne, and Gianni Bartolacci. A machine vision approach to on-line estimation of run-of-mine ore composition on conveyor belts. *Minerals Engineering*, 20(12):1129–1144, 2007.

Paul Geladi and Hans F Grahn. Multivariate image analysis. *Encyclopedia of Analytical Chemistry: Applications, Theory and Instrumentation*, 2006.

MH Bharati and John F MacGregor. Multivariate image analysis for real-time process monitoring and control. *Industrial & Engineering Chemistry Research*, 37(12):4715–4724, 1998.

Honglu Yu and John F MacGregor. Multivariate image analysis and regression for prediction of coating content and distribution in the production of snack foods. *Chemometrics and Intelligent Laboratory Systems*, 67(2):125–144, 2003.

Majid Mirmehdi. *Handbook of texture analysis*. Imperial College Press, 2008.

Azriel Rosenfeld and Eleanor B Troy. Visual texture analysis. Technical report, Maryland Univ., College Park (USA). Computer Science Center, 1970.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.