Journal of Computer Languages 54 (2019) 100908

Contents lists available at ScienceDirect

Journal of Computer Languages

journal homepage: www.editorialmanager.com/cola/default.aspx

ML-LUM: A system for land use mapping by machine learning algorithms

Liao Xiaojin[a], Huang Xiaodi[a,*], Huang Weidong[b]

[a] Charles Sturt University, Albury, NSW 2640, Australia
[b] University of Technology Sydney, Ultimo, NSW 2007, Australia

ARTICLE INFO

Keywords:
Image data mining
Classification
Capsule network
CNN
Land use mapping
Visualization

ABSTRACT

The land use mapping refers to mapping and assessing changes and patterns of land use. The use of agricultural land maps becomes increasingly important. The governments, private sectors, research agencies, and community groups rely on land use mapping data for natural resource assessment, monitoring, and planning. Finding an effective mapping approach is thereby crucial for natural resource condition monitoring and investment, agricultural productivity, sustainability and planning, biodiversity conservation, natural disaster management, and bio-security.

In this paper, four machine learning algorithms, i.e., the classic k-Nearest Neighbour (kNN), Support Vector Machines (SVMs), Convolutional Neural Network (CNN), and newly developed Capsule Network (CapsNet), are applied to classify satellite images for land use. For comprehensively comparing the performance of different algorithms for land use mapping, the experiments have been conducted on real-world datasets. Based on the experiment results, several improvements on the algorithms are proposed in order to fulfil the requirement of a large-scale land mapping. In addition, we design and implement these algorithms for land use mapping in a Machine Learning Land Use Mapping (ML-LUM) system. The system is able to train the models, predict classifications of satellite images, map the land use, display the land use statistic data, and predict production yields. With a friendly graphic user interface for farmers, the system is implemented by using the cloud computing technique for processing large land use data. Furthermore, we present a case study. For the case study, a banana plantation area from a given satellite image is correctly marked and the area size is then calculated, together with predicting banana production.

1. Introduction

Land use mapping becomes increasingly important. For example, the remote sensing centre in Queensland relies on satellite remote sensing data to map land use and land use changes, under the Australian Land Use Mapping Program (ALUMP) [1]. The program aims to build nationally consistent land use datasets so as to meet a wide range of user needs and to make the best use of existing data and available resources [1]. In particular, ALUMP provides nationally consistent spatial information in order to assist in planning, natural resource condition monitoring and investment, agricultural productivity and sustainability, biodiversity conservation, as well as natural disaster management and bio-security. As such, the Australian government, private sectors, research agencies and community groups can use the ALUMP datasets for natural resource assessment, monitoring, and planning [1].

The majority current methodologies of mapping land use still require skilled teams to manually interpret satellite imagery data and manually classify land use features [2]. It is obvious that this process is time and resource intensive. Finding an effective mapping approach will significantly reduce such costs.

With the rapid advances in technologies, automatic land use mapping becomes possible. Advances in machine learning (ML) enable to train a machine learning model on ALUMP's datasets for automatically mapping land use features [2]. In other words, ML algorithms make automatic land use mapping possible, by training models on all of the available data. As such, machine learning algorithms can be used to identify the land use from satellite images, to reduce the intensive manual effort required and to produce results in a shorter period of time [3]. Furthermore, the use of machine learning for land use mapping can make predictions. This can guide better decisions and smart actions for land use in real time. In addition, it can help seasonal crop mapping and monitoring [3] and control, preventing Banana Bunchy Top Virus (BBTV) outbreak [4].

This study will investigate the use of machine learning to identify land use from satellite imagery so as to reduce intensive manual efforts

* Corresponding author.
E-mail addresses: xhuang@csu.edu.au (X. Huang), weidong.huang@uts.edu.au (W. Huang).

https://doi.org/10.1016/j.cola.2019.100908
Received 27 February 2019; Received in revised form 9 June 2019; Accepted 1 August 2019
Available online 14 August 2019
2590-1184/

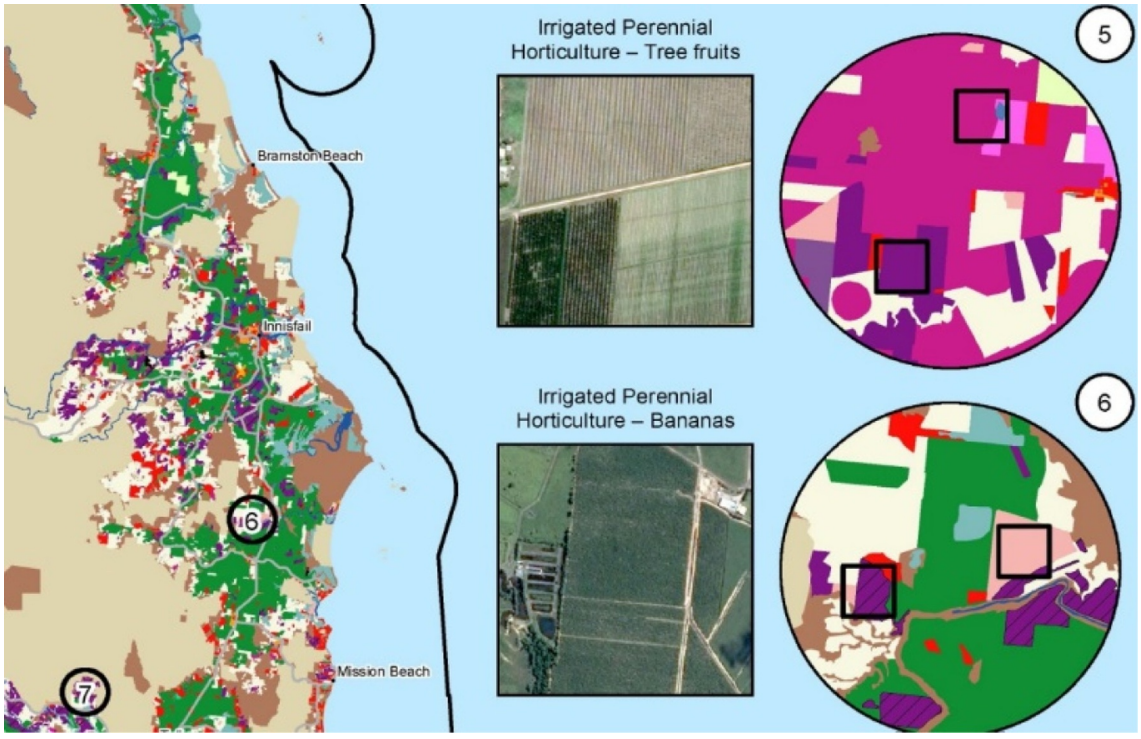

Fig. 1. An example of QLD natural resource management map 2015 [2].

required and obtain results in shorter timeframes. The question is how to apply appropriate machine learning algorithms to automatically identify and classify land use.

In order to answer this question, we start by identifying the requirements of land use mapping as follows:

1. Accuracy: a high accuracy is expected for the land use mapping; otherwise manual corrections by experienced staff are still needed.
2. Processing speed: the speed of data processing is essential. This is particularly true when the outcome results are in an urgent need, such as disaster management.

These requirements for land use mapping pose challenges to automatic mapping systems. Further, the computing capacity is essential for the huge size of the data. For example, Queensland Australia has approximately 173 million hectares in area. A baseline dataset of land use mapping in Queensland contains approximately 160,000 features [3]. An example is shown in Fig. 1. It is a challenge for a program and algorithm to process such a large dataset.

In this paper, we have made the following contributions:

- we conduct extensive experiments on comparing several important ML algorithms for land use mapping.

In particular, the deep learning algorithms of CNN and the newly developed CapsNet are used. To the best of our knowledge, this is the first work on applying deep learning algorithms, particular the newly developed CapsNet algorithm to land use mapping. Some improvements on the algorithms for land mapping use have been proposed, particularly for the settings of parameters in the algorithms.

- We make recommendations on how to choose algorithms for land use mapping under different cases are made. The recommendation is based on comparing and analysing the experimental results on both public datasets and satellite imagery datasets.
- We design and develop a system that implemented these machine learning algorithms in land use mapping (ML-LUM). By giving a set of satellite images, the ML-LUM system uses different machine learning algorithms to produce classification outputs. The system is implemented in a cloud computing environment that uses supercomputer resources for processing huge data quickly.

The organization of this paper is as follows. Section 2 reviews related work. Section 3 presents the compared approaches, followed by Section 4 that presents the cloud implementation of the system. Section 5 describes the experimental methodologies. Section 6 provides an illustrative case study. The paper concludes in Section 7.

2. Related work

2.1. Algorithms in land mapping systems

A number of methods for classifying satellite images are available. These methods can be classified into four categories [5]: template matching-based methods, knowledge-based methods, OBIA-based methods, and machine learning-based methods.

The template matching-based methods, knowledge-based methods, and OBIA-based methods are built on traditional techniques such as segmentation, edge detection, feature extraction, and classification. These methods have been used in remote sensing image analysis for decades.

In the Geographic Object-Based Image Analysis (GEOBIA), pixels are grouped into uniform objects, which are then classified or marked and modelled [4]. The main strength of the developed object-based image analysis approach is the high detection rate. The GEOBIA is effective in remote sensing image analysis. However, it still faces challenges in order to achieve the goal of geospatial source automatic geographic intelligence [6].

Random Trees are an ensemble of Decision Tree (DT) classifiers. DT uses a chain of simple decisions based on the results of sequential tests

for the class label assignment [7].

However, the biggest drawback with these traditional algorithms is the accuracy rate. This makes the template match methods hard to achieve a satisfactory accuracy rate. The processing speeds are also not well enough for computing a large amount of data in a short time.

2.2. Systems

The ESRI ArcGIS ArcMap software is a popular Geographic Information System (GIS), which supports the supervised classification process. These algorithms make use of minimum distance, maximum likelihood, and Mahalanobis' Distances [8]. The ArcMap has also been used in the Integrated Seafloor Classification Scheme (G-ISCS) method to analyse geo-morphological attributes [9]. By using imagery classification through geo-morphological attribute analysis, spatial distribution, the quantity of occurrence, an approximate area of each interpreted landform feature are determined. Using the G-ISCS method with IKONOS-2 imagery and ArcMap classification helps where the distribution of specific landform occurs along the region. As commercialized software tool suite, the eCognition is capable of classifying geomorphological satellite imagery. It makes use of the nearest neighbour and the integrated fuzzy rule-based classification method [10]. Building on fractal net evolution, the eCognition system follows a completely programmable workflow [11].

These most currently used methods in satellite image classification are still not good enough in terms of accuracy rate or processing speed. For example, by using the eCognition client version, Kasper Johansen identified banana plants from satellite images with an accuracy of 79% ~ 88%. The main limitations of the mapping approach are a large number of false positives, which require the final visual assessment step to reduce the errors [4]. This means that the result might still need human correction.

Moreover, in Kasper Johansen's experiments, the average processing time is 13 min per 1 km^2 tile. This is much faster than 73 min per square kilometre of manual interpretation speed by skilled staff. The eCognition server has some capacity in processing large data. The server runs with a total of 16 computer clusters. The processing speed achieves about 500–600 tiles per 24 h [4]. However, the accuracy and the processing speed of current methods are barely enough to run automatic geographic intelligence. It still needs human correction, which is a heavy workload. Some procedures take more than 30 steps to obtain the final results.

In order to overcome the drawbacks of the currently existing systems on land use mapping, we design and develop our system called ML-LUM.

3. Implemented algorithms in the ML-LUM system

In our ML-LUM system, the following algorithms are implemented: kNN, SVM, CNN, and CapsNet. The former two are traditional ones while the latter two are recently advanced ones. We briefly review the four algorithms in the following.

k-Nearest Neighbour (kNN): As one of the simplest classification algorithms available for supervised learning [12,13], kNN searchs for the closest match of the test data in feature space [12].

Input: a training sample $S = (\mathbf{x}_1,y_1),\ldots,(\mathbf{x}_m,y_m)$

Output: for every point $\mathbf{x} \in X$, return the majority label among $\{y_{\pi i(\mathbf{x})}: i \le k\}$

A weighted average of the targets can be taken according to the distance from $\mathbf{x}$ [14]:

$$h_S(x) = \sum_{i=1}^{k} \frac{\rho(x, x_{\pi_i(x)})}{\sum_{j=1}^{k} \rho(x, x_{\pi_j(x)})} y_{\pi_i}(x) \tag{1}$$

Support Vector Machines (SVMs): SVMs have been extensively applied in the classification of images [15]. The goal of the SVM is to separate the classes by maximizing the distance between the hyperplane and observations. Maximization of the margin is to minimize $(1/2)||w||^2$ subject to $y_i\,(W^{\mathrm{T}}\mathrm{X}i + b)\text{-}1 \ge 0$ for all i [16]. The final SVM can be written as Eq. (2).

$$L = \sum_i di - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j (\bar{X}_i \bar{X}_j) \tag{2}$$

Convolutional Neural Network (CNN): CNN employs an operation called convolution, which is a specialized kind of linear operation. CNN relies on weight sharing, without maintaining the spatial position of learning characteristics [17]. The trained CNN can be treated as a feature extractor that has learned a nonlinear mapping from each input image [18,19].

Eq. (3) is for using a two-dimensional satellite image I as the input and a two-dimensional kernel K:

$$S(i,j) = (I^*K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n) \tag{3}$$

In the experiments, a set of different settings is compared. With setting $m, n = 5$, the system obtains the best accuracy result.

*Capsule networks (CapsNet):*introduced by Hinton and his colleagues in 2017, it is the-state-of-art machine learning algorithm. The approach has reduced both error rates on MNIST by 45% and training set sizes. Experiment results have shown that CapsNet performs considerably better than CNN on highly overlapped digits [20].

The output vector of a capsule represents the probability of an entity represented by a capsule in a current input. The nonlinear activation of Eq. (4) V_j is the vector output of capsule j, and S_j is its total input.

$$V_j = \frac{||S_j||^2}{1 + ||S_j||^2} \frac{S_j}{||S_j||} \tag{4}$$

Except for the first layer of capsules, the total input to a capsule S_j is a weighted sum over $\hat{u}_{j|i}$ from the capsules in the layer below.

$$S_j = \sum_i c_{ij} \hat{u}_{j|i} \tag{5}$$

The "prediction vectors" $\hat{u}_{j|i}$ from the capsules is an affine transform by multiplying the output u_i of a capsule in the layer below by a weight matrix W_{ij} . [20]

$$\hat{u}_{j|i} = W_{ij}\, u_i \tag{6}$$

Using a non-linearity, capsules convert a whole set of activation probabilities and their poses in one layer into the activation probabilities and poses of capsules in the next layer [21]. Capsule networks perform much better than the SVM when being trained on the large amounts of pictures. For identifying overlapping pictures, the performance of capsules is superior to that of CNN [22].

In our experiments, k-nearest neighbour (kNN) is also tested. kNN has the fastest training speed compared with other ML algorithms. But the accuracy rate drops significantly when it processes more complex datasets.

4. Overview of ML-LUM

For comparing the algorithms, the ML-LUM system is designed and developed to implement the ML algorithms in land use mapping. The modular design is applied to make the system extendable.

4.1. Modules

The ML-LUM system consists of the following modules: GUI, Dataset, Algorithms, Post-processing and Draw imagery, as shown in Fig. 2. A cloud training module is also built for using HPC or the supercomputer.

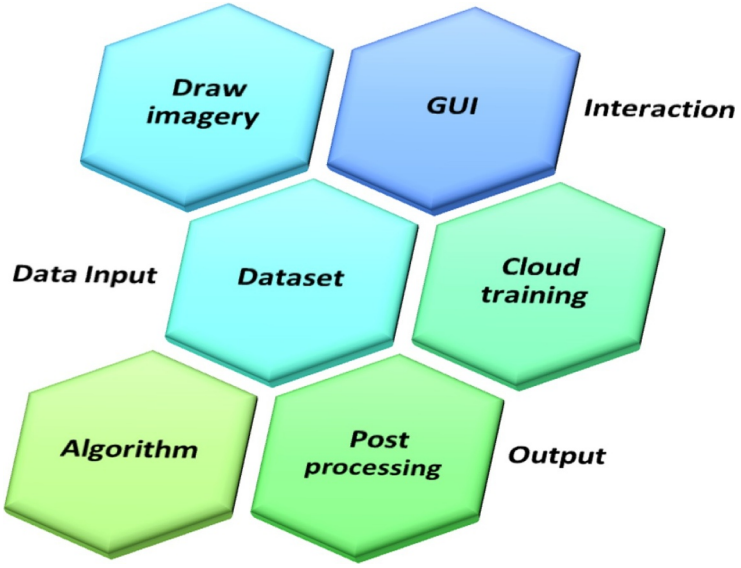

Fig. 2. The structure of the image classification software modules in ML-LUM.

4.2. Graphic user interface (GUI)

The ML-LUM system has a graphic user interface. Tkinter is used to build its GUI interface. The Tkinter is chosen because it is compatible with Windows, Linux and Mac OS. Therefore, the system can be running under all the operating systems.

As shown in Fig. 3, the interface of ML-LUM contains five components: menu, toolbar, setting list, result display panel, and an information output panel.

1. Top Menu bar – the top menu bar consists of the six sets of drop-down menu list (File, Dataset, Algorithm, Classification, Tools, and

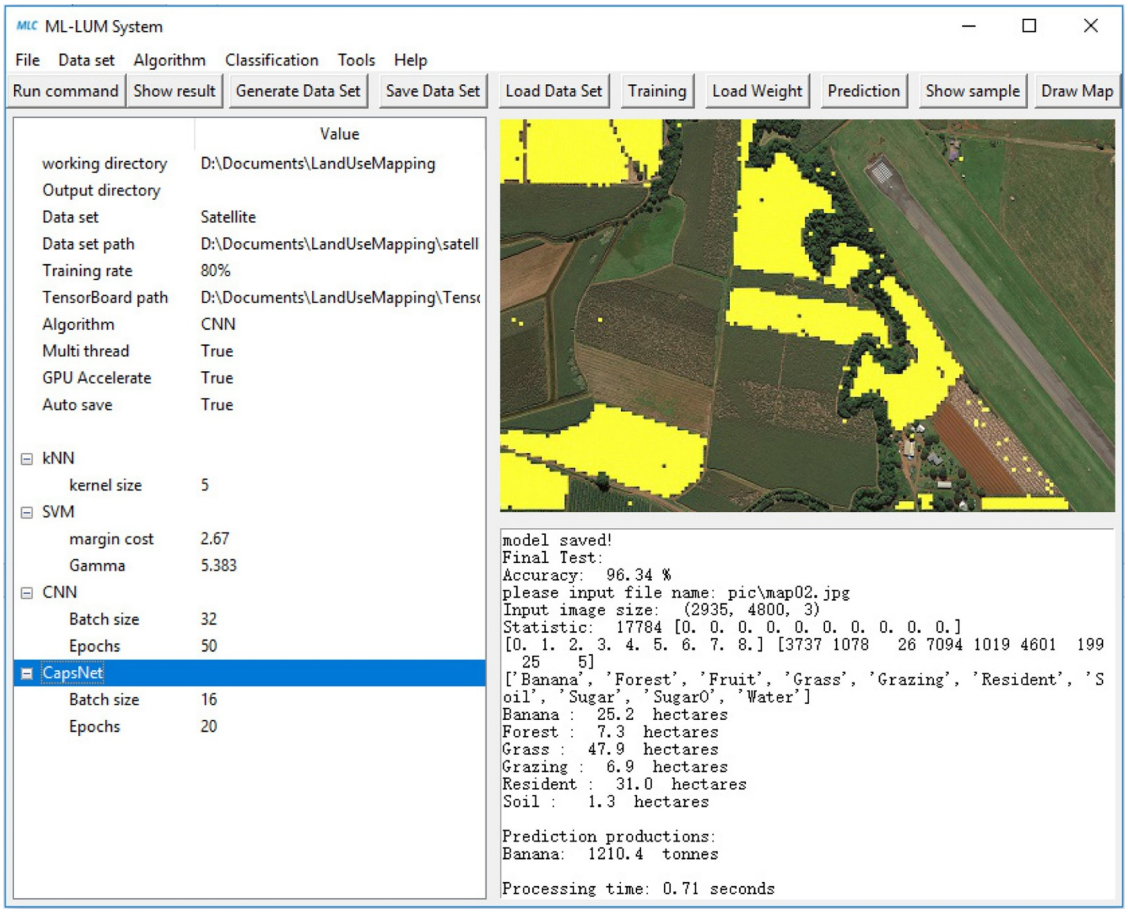

Fig. 3. The user interface of the ML classification.

Help). It includes all the settings, functions, and tools.
2. Toolbar – the toolbar includes frequently used commands and functions from dataset to training and to prediction tasks.
3. Side Parameter Panel – the options panel displays global and algorithm-specific options which a user can customize their settings.
4. Results Display Panel – the results pane shows the results of the training, prediction or mapping process.
5. Information Output Panel– the information panel displays all the results from an ML classification session e.g. training time, accuracy. It also shows the statistics of land using data and the production prediction.

By default, the outputs of the system are labelled images and their statistic information. For satellite images, a user can select the 'draw map' function to output a marked map and area information as well as production prediction.

4.3. Datasets processing module

The dataset module is designed to pre-process a dataset. Its functions include produce a dataset, reformat data, and shuffle the dataset. This module supports uploading and downloading a dataset, transforming images and arrays, saving and loading ML models, saving and loading trained weights, and making predictions. By navigating the interface of ML-LUM Classification, users are able to create a customised dataset, which can be downloaded, by selecting different available images in the system.

In the 'create' mode, a user can select the target folder and load images from all subfolders. In the following step, the system resizes or segments all of these images into the same specified size. These images are then converted into arrays and joined together. If needed, the array can be rearranged to suit the requirement of the algorithms.

4.4. Algorithm module

In the algorithm module, the system implements the four machine learning algorithms: kNN, SVM, CNN, and CapsNet. As shown in Fig. 4, each of these algorithms has three sub-modules of pre-processing, training, and prediction.

After the pre-processing step, labelled images are stored into an array with a specific format to meet the input requirement of an algorithm. The data can be converted into other data types, with their reshaped or adjusted dimensions. For a particular algorithm, for example, a data point that is inputted to SVM is converted into a vector by using the histogram of the oriented gradients (HOG function). The HOG function is an expensive processing, particularly for training and predicting on a large number of images. Multi-thread computing is applied for this function. The part of the code is as follows:

```
…
def hog(img):
        bin_n = 16 # Number of bins
        HALF_W = img_W // 2
        gx = cv2.Sobel(img, cv2.CV_32F, 1, 0)
        gy = cv2.Sobel(img, cv2.CV_32F, 0, 1)
        mag, ang = cv2.cartToPolar(gx, gy)
        bins = np.int32(bin_n * ang / (2 * np.pi))
        bin_cells = bins[:HALF_W,:HALF_W], bins[HALF_
W:,:HALF_W], bins[:HALF_W,

       HALF_W:], bins[HALF_W:, HALF_W:]
        mag_cells = mag[:HALF_W,:HALF_W],    mag[HALF
_W:,:HALF_W], mag[:HALF_W,
        HALF_W:], mag[HALF_W:, HALF_W:]
        hists = [np.bincount(b.ravel(),    m.ravel(),
bin_n) for b,
        m in zip(bin_cells, mag_cells)]
        hist = np.hstack(hists)
return hist
…
if __name__ == '__main__':
        …
        pool = Pool() # Create a multiprocessing Pool
        hog_train = np.float32(pool.map(hog, x_train))
# process data with pool
        pool.close()
        # Training
        svm.train(hog_train,      cv2.ml.ROW_SAMPLE,
y_train.astype(int))
        # Testing
        e2 = cv2.getTickCount()  # Measuring  perfor-
mance Training
        pool = Pool() # Create a multiprocessing Pool
        hog_test = np.float32(pool.map(hog, x_test))
        pool.close()
        pool.join()
        ret, result = svm.predict(hog_test)
        mask = result == y_test
        correct = np.count_nonzero(mask)
        print('Accuracy: ', correct * 100.0 / re-
sult.size, '%')
        …
```

Fig. 4. Algorithm Module and its sub-modules.

4.4.1. Training

The training module sets some parameters to achieve the best accuracy and efficiency. By setting training rate parameters, a user can partition data for training, and the rest of the data for verification. For the best accuracy, the training for kNN and CNN should be set to 80 for satellite images, 50 for SVM, and 70–80 for the CapsNet algorithm. The parameters also include the kernel size, margin cost, gamma, the batch size, and the number of epochs. All of these parameters need to be configured according to different algorithms.

After setting all the parameters, a training process starts. The training time varies according to the selected algorithms and datasets. The progress report is generated in the result display area. The report includes current steps, accuracy, and lost. When a training process finishes, summary information is displayed, including the overall accuracy, validation accuracy, lost, and time used. The trained model and weights can be saved at this step. After training the model, it is not necessary to re-start the training every time for making predictions.

4.4.2. Prediction

After the machine learning model has been trained or loaded from previously trained data, the model is ready to predict the labels of images. Given the full path of images for prediction, the system reshapes these images and uses the trained model to classify them. The outputs of kNN and SVM are the matched labels and the distances. While the outputs of CNN and CapsNet are a vector, respectively, these data can be converted into a classification label by using the one-hot encoding method. The probability of a prediction can also be given.

4.5. Post-processing module

The classified result is marked on an image according to its identified label. We implement a filter function by applying morphological transformations. An erosion is first applied, followed by dilation. By setting with a proper kernel size, the small noise can be filtered out. The final output on the map is clear and accurate, compared to the original one.

The statistics and analysis of the output data are also provided in the ML-LUM system. From a prediction result, the land size of each class can be calculated. The production of a given area can be then predicted by using these statistic results. Because it is a fully automatic process, the whole process can be done in one second. This meets the requirement identified before.

4.6. Cloud computing module

A further optimization has been achieved, which allows the ML-LUM system to run in a cloud environment with cloud hardware acceleration. This enables the application to use high-performance computing resources or even a supercomputer for processing big data. Some of the libraries and functions need to be changed or modified before they can run in the cloud environment with cloud hardware acceleration.

The NeCTAR cloud platform provides Ubuntu Linux 18.04 LTS Bionic with Jupyter Notebook. By uploading a training dataset into Google Drive, the system is able to run in the cloud environment, as shown in Fig. 5.

The ML-LUM system is also tested in the Google Colaboratory platform, which is a cloud-based tool for machine learning and research. It provides a Jupyter Notebook environment with GPU hardware acceleration. An instance is created in Clouderizer and used to connect the Google Colaboratory. The running speed of CNN and CapsNet cloud instances is several times faster than that of those running in a local machine.

5. Experiments

Four ML algorithms have been implemented to process various satellite images and to produce classification results by using Python 3. The selection of Python 3 as the programming language is based on its great range of new libraries for machine learning applications. Python 3 supports TensorFlow which was developed by Google Brain Team. It is also able to build applications with multi-thread and GPU accelerate support.

The main libraries include TensorFlow-GPU, Keras, OpenCV and Matplotlib. The TensorFlow and Keras are used for developing CNN and the Capsule Network programs. The OpenCV is used for image processing. The Matplotlib is used to visualise the processes and results.

5.1. Datasets

Three different datasets are used in the experiments. These datasets are chosen with the aim to cover from the basic to advanced, and from simple to difficult levels. By doing so, the characteristics of each ML algorithm with data of different types can be compared and analysed.

The first dataset is the MNIST (Modified National Institute of Standards and Technology) handwriting database [23]. It has widely been used in machine learning research communities. The use of this dataset can help quickly tune the parameters of the algorithms.

The second dataset is the CIFAR-10 (Canadian Institute for Advanced Research) dataset. It is a very complex dataset for supervised ML algorithms. It contains 60,000 32*32 colour images. These images have 10 classes, such as aeroplane, car, bird, and cat [24]. Fig. 6 shows some samples from the MNIST and the CIFAR-10 datasets.

The third dataset is a customised satellite image dataset. It is the core dataset in our experiments. We attempt to find the best classification algorithm for this satellite image dataset. The Wet Tropics Natural Resource Management (NRM) Region Map 2015 is used to build this customised satellite image data. It contains banana, fruit plantations, and other land use classes. As shown in Fig. 7, ten typical classes have been chosen. High definition satellite images are downloaded. These images have been converted into greyscale and segmented into 28*28 pixels images with labels. These images are then shuffled to build the dataset. The total size of the satellite image dataset is 102,222 images. It is then divided into 3 groups: 73,778 images for training, 18,444 images for test, and 10,000 images for the final verification.

5.2. Experiments results and discussions

kNN: training a kNN model is very fast because it basically only inserts the data into the right positions of the model. The comparison of the experiment results on the MNIST dataset and satellite image dataset shows that, in both datasets, the k value does not significantly affect the accuracy, and the processing speed neither. But the accuracy rates with different datasets are significantly different, as given in Table 1.

SVM: in a test, the Histogram of the Oriented Gradients (HOG) function is used to extract the feature vectors from an image. The vectors are then output to the SVM. In the experiments, it is noticeable that when the size of a dataset becomes bigger, the HOG function could take the extensive calculation time. For the satellite image dataset test with more than 100,000 images, the HOG function takes 3 min for its calculation. After we optimize the parameters and use multi-processing (8 threads), the processing time is, however, reduced to 5.22 s.

Table 2 reports the training time, testing time, and accuracy of SVM with respect to different datasets and different parameters of C (the soft margin cost) and Gamma. Large numbers of parameters have been tested, but here only those with best results are listed. The parameters C and G do affect the results slightly. The SVM algorithm can correctly predict categories of most images. The SVM has achieved much better results in the satellite image dataset, compared to the kNN algorithm

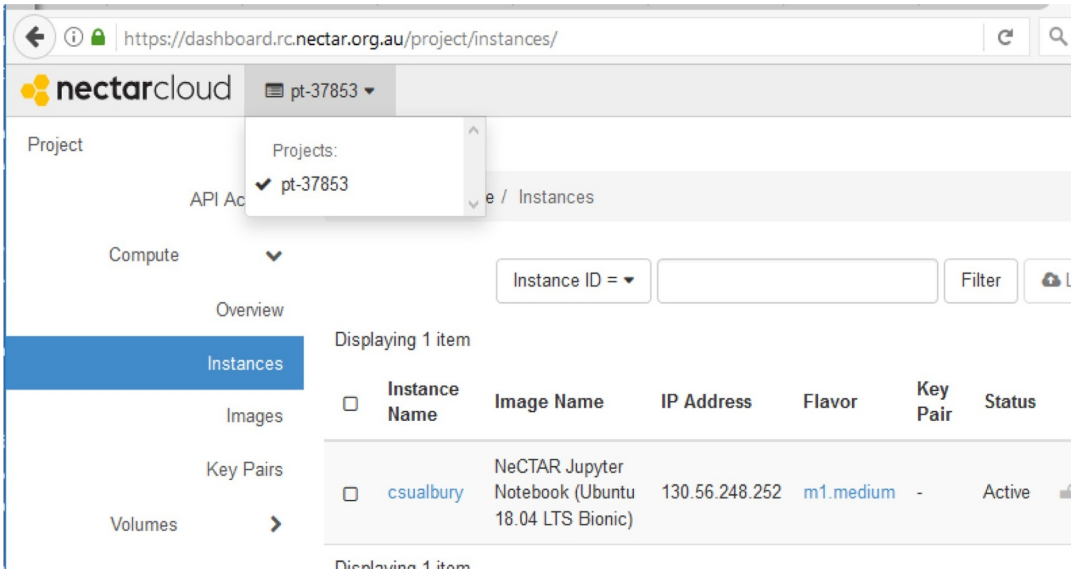

Fig. 5. A cloud implementation with the NeCTAR cloud.

Fig. 6. MNIST dataset and CIFAR-10 dataset.

with only 54% accuracy on satellite image datasets. When using SVM to process the CIFAR-10 dataset, the accuracy, however, achieves only around 40% (Table 3).

From the experiments, it is also found that the more training data do not always obtain a better result. Using too much training data may cause overfitting with the SVM algorithm, lowering the accuracy.

CNN: It uses much more computing resources in training than the SVM algorithm. In the experiments, a few CNN models are built with different depths and sizes in order to find a better one for the satellite image dataset. The best model that is chosen for the experiments is the 6 layers convolutional neural network. The detailed model structure is shown below.

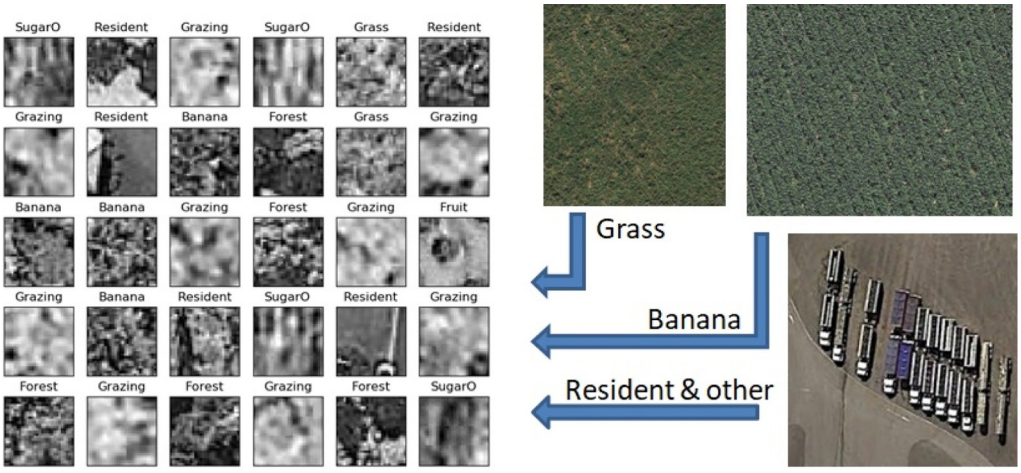

Fig. 7. The dataset of satellite images.

Table 1
The kNN test results with different k values.

Dataset						
k	MNIST dataset			Satellite image dataset		
	Accuracy	Train	Test	Accuracy	Train	Test
$k = 1$	96.82	0.06	70.8	54.06	0.09	169
$k = 3$	97.04	0.06	70.7	53.44	0.09	173
$k = 5$	96.88	0.06	69.3	52.59%	0.09	171
$k = 9$	96.82	0.06	70.3	51.3	0.09	167

*The training and testing process has not been optimized with multi-process.

Table 2
SVM accuracy and training time on satellite images.

Dataset	Parameter*	Accuracy	Training time	Test time
Satellite Image	$C = 50$ $G = 100$	74.75%	2.88s	5.04s
Satellite Image	$C = 2.67$ $G = 5.383$	75.38%	2.82s	5.19s
MNIST	$C = 2.67$ $G = 5.383$	94%	2.83s	4.39s

* C and G are parameters for the SVM algorithm the soft margin cost and gamma.

Table 3
Compare four ML algorithms.

Time (s)	kNN	SVM	CNN	CapsNet
Accuracy (MNIST)	97.04%	94%	98.7%	99.67%
Accuracy (Satellite Dataset)	54%	75%	98.09%	96%
Accuracy (Cifar-10)	24%	34%	81.59%	66.95%
Training Time(s)	0.09	2.82	572	46,530
Testing Time(s)	169	5.19	1.5	18
Prediction Time(s)	395	1.14	0.72	52

```
   convnet = input_data(shape=[None, img_W, img_H, 1], name='input')

   convnet = conv_2d(convnet, 32, 5, activation='relu')
   convnet = max_pool_2d(convnet, 5) #

   convnet = conv_2d(convnet, 64, 5, activation='relu')
   convnet = max_pool_2d(convnet, 5)

   convnet = conv_2d(convnet, 128, 5, activation='relu')
   convnet = max_pool_2d(convnet, 5)

   convnet = conv_2d(convnet, 64, 5, activation='relu')
   convnet = max_pool_2d(convnet, 5)

   convnet = conv_2d(convnet, 32, 5, activation='relu')
   convnet = max_pool_2d(convnet, 5)

   convnet = fully_connected(convnet, 1024, activation='relu')
   convnet = dropout(convnet, 0.8) #

   convnet = fully_connected(convnet, 10, activation='softmax')
   convnet = regression(convnet, optimizer='adam', learning_rate=LR, loss='categorical_crossentropy', name='targets')
model = DNN(convnet, tensorboard_dir='tensorboard')
```

It takes about 10 min to run 50 epochs in the training for this 6-layer CNN on the satellite image dataset. The training accuracy is 95%. As shown on the left side of Fig. 8, the final validation accuracy is 98.09%. The right side of Fig. 8 displays the sample outcomes of CNN predictions.

To obtain a better accuracy on the CIFAR-10 dataset, another CNN model is built with an 8-layer Conv2d neural network. The best validation accuracy on the CIFAR-10 in the experiments is 81.59%.

CapsNet: the Capsule Network model used in the satellite image experiments is shown in Fig. 9.

The CapsNet algorithm consumes more computing resources. It spends 13 h on running 50 epochs, with its accuracy of 96.82%. The CapsNet works well with the lower training rate, and it can use the less amount of training data to achieve a good accuracy. The left hand of Fig. 10 shows the accuracy, while its right side shows a reconstructed image from this model.

By using the 8-layer Conv2d model, 50 epochs of CapsNet training are run on the CIFAR-10 dataset. The final results show that the CapsNet has achieved 71.03% training accuracy and 66.95% validation accuracy. According to Jaiswal et al., the accuracy can be further improved by adjusting and optimising the model [25].

5.3. Comparisons

In this section, the performances of the four algorithms are compared against three datasets in terms of their accuracy and running time.

5.3.1. Accuracy

For the MNIST dataset, all of kNN, SVM, CNN, and CapsNet have obtained good results in the experiments. Even the simplest kNN algorithm achieves the 97% accuracy. For the CIFAR-10 dataset with the complex images, SVM achieves only 34% accuracy, and kNN gets even less. But CNN and CapsNet achieve around 70–80% accuracy. Fig. 11 plots that CNN performs best with 98.09% accuracy, CapsNet with nearly 96%, and SVM with 75% for the satellite image dataset.

The same algorithm trained on different datasets can result in quite different accuracy rates. Table 4 lists performance measures of CNN and CapsNet algorithms. These two neural network models are capable of modelling very complex relationships in the data and producing good results. In contrast, SVM and kNN are not good at handling such complex images.

5.3.2. Running time

Table 5 lists all running times for the training, testing and prediction of the four algorithms against the satellite image dataset. SVM takes the least running time with the minimum computer resource. Training SVM takes few seconds and one square kilometre tile is mapped in only 1.14 s.

Training CNN takes 10 min with running 50 epochs. It takes 13 h for training CapsNet with 50 epochs on the test platform. After training, the prediction of a new image is, however, much faster. CNN takes less a second to mark a square kilometre map. CapsNet takes 52 s, which are longer than the time CNN and SVM do.

From these experiments, the CNN algorithm is found to be the best performer in mapping the satellite images. The SVM algorithm is very fast in training and testing, however, the HOG function for SVM that extracts the feature vectors takes more processing time. The CapsNet has achieved the highest accuracy, but it takes much more training time and prediction time than the other two algorithms do. Training the CNN also takes a couple of minutes, but its prediction is faster than the other two algorithms in the experiments.

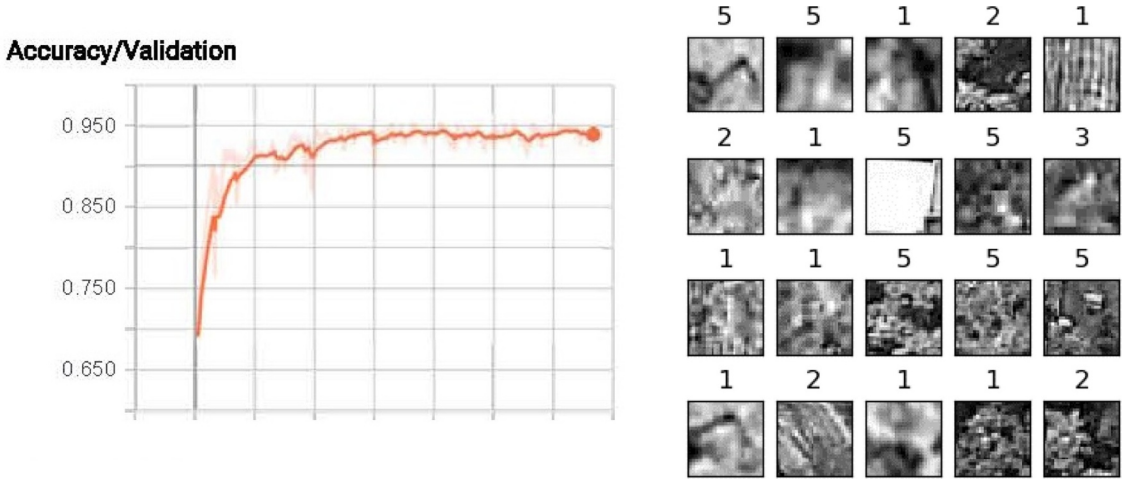

Fig. 8. CNN validation accuracy and output samples.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 28, 28, 1)	0	
conv1 (Conv2D)	(None, 20, 20, 256)	20992	input_1[0][0]
primarycap_conv2d (Conv2D)	(None, 6, 6, 256)	5308672	conv1[0][0]
primarycap_reshape (Reshape)	(None, 1152, 8)	0	primarycap_conv2d[0][0]
primarycap_squash (Lambda)	(None, 1152, 8)	0	primarycap_reshape[0][0]
digitcaps (CapsuleLayer)	(None, 10, 16)	1474560	primarycap_squash[0][0]
input_2 (InputLayer)	(None, 10)	0	
mask_1 (Mask)	(None, 160)	0	digitcaps[0][0] input_2[0][0]
capsnet (Length)	(None, 10)	0	digitcaps[0][0]
decoder (Sequential)	(None, 28, 28, 1)	1411344	mask_1[0][0]

Total params: 8,215,568
Trainable params: 8,215,568
Non-trainable params: 0

Fig. 9. The summary of experiment capsule network model.

5.3.3. Recommendations

For an image classification in a simple application, kNN and SVM algorithms are recommended as a simple and easy to use the classifier. From the experiments, all these four ML algorithms are good at processing MNIST dataset. Although the results are different, all of them have achieved more than 97% accuracy. However, the kNN and SVM can be easily trained, the code is simple, without GPU acceleration. These are the advantages of the two ML algorithms.

For real-time land mapping on a large scale, it is recommended that CNN for mapping satellite images should be used instead. The CNN verification accuracy is slightly higher than the CapsNet. According to the real-world mapping results, CNN performs better than CapsNet. This could be caused by the over-fitting of the CapsNet. CNN also uses much less training and prediction time, which is another reason why CNN is recommended for the satellite imagery classification.

6. A case study: estimation of banana plantation areas using ML-LUM

In order to further validate the experiment conclusions, we compare these algorithms by applying them to a real-world example. Given high definition satellite images as inputs, the trained model will identify which area is a banana plantation. After this, the ML-LUM system calculates the size of the identified areas, and predicts the productions of

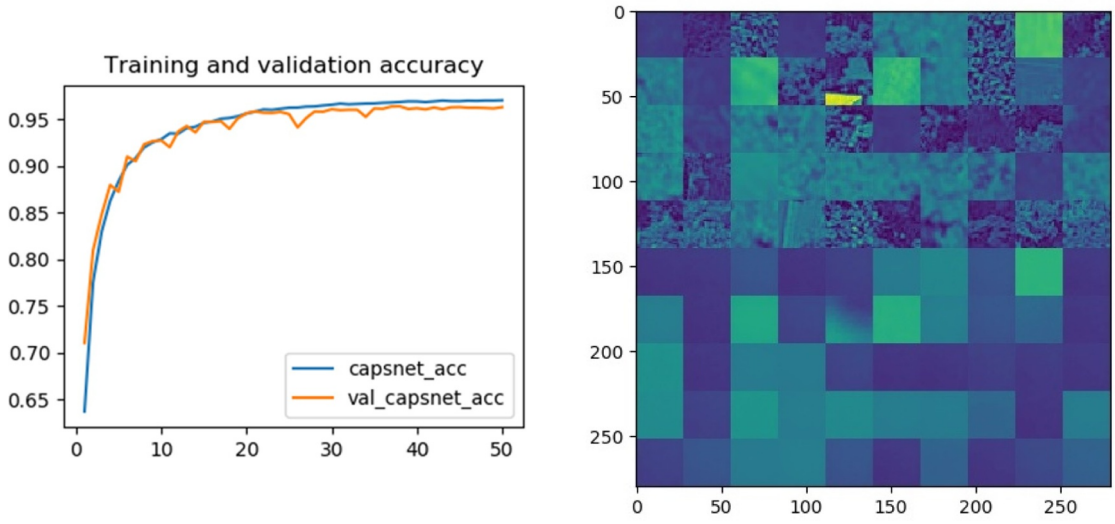

Fig. 10. CapsNet accuracy and reconstructed images from the satellite image dataset.

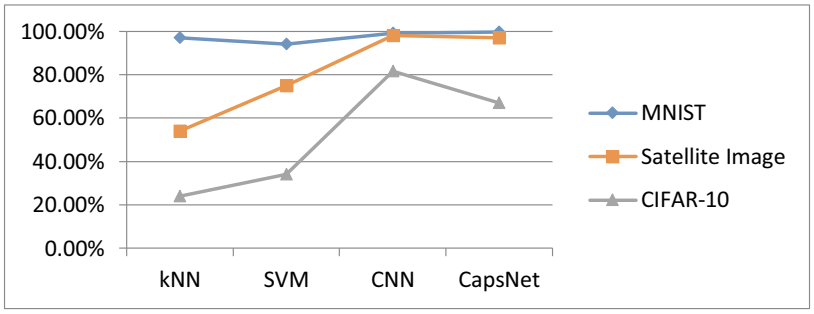

Fig. 11. Comparisons of accuracies of four algorithms on different datasets.

Table 4
Performance measures of CNN and CapsNet algorithm.

Algorithm	Errors	Accuracy	Precision	Recall	F1 Score
CNN	1.91%	98.09%	91.87%	97.49%	94.60%
CapsNet	3%	97.00%	87.38%	96.45%	91.69%

Table 5
Training and test times on the satellite image dataset.

	kNN	SVM	CNN	CapsNet
Training Time (s)	0.09	2.82	572	46,530
Test Time (s)	49	5.19	1.5	18
Prediction Time(s)/km^2	95	1.14	0.72	52

Table 6
Performance comparisons of the algorithms on a real-world example.

	kNN	SVM	CNN	CapsNet
Accuracy (%)	52.3	75.38	98.09	96.82
Predict time (s)/km^2	104	1.14	0.64	52

the banana plantation.

The ML-LUM system makes use of the trained model to classify land use. It also marks the banana plantation (Banana Farms) in the given satellite images. These trained models are those that are able to achieve the best accuracy by each of the compared algorithms. The prediction time of each algorithm mapping over one square kilometre land is recorded. Table 6 reports the training accuracy and average prediction times of each algorithm for one square kilometre tile.

From Fig. 12, it can be seen that CNN performs best in identifying and marking banana plantation areas. It is remarkable that CNN takes only 0.66 s for mapping per square kilometre land use. CapsNet has the highest verification accuracy. But it takes much longer when applied to the satellite image. The output image also shows more noise spots from the result of CapsNet algorithm. This may result from the overfitting with the training dataset.

Fig. 13 is a mapping output for another class: grass vegetation. It demonstrates that the ML-LUM system is not only capable of a single use of banana classification, but it can also easily classify other types of the plantation.

Furthermore, a simple process of opening and closing filtering has been applied to remove noises from the output of CNN. The areas of the banana plantation are perfectly marked from a given image. The ML-LUM system produces the statistics of the detailed land use, and predicts the productions of the land use, as given in Fig. 14.

7. Conclusion

Land use mapping becomes increasingly important. In this paper, we have compared several ML algorithms ranging from classic to state-of-the-art against the three datasets for the mapping task. In order to achieve the best performance, we have improved the algorithms for this particular task. Through the experiments, we conclude that CNN and CapsNet are good candidates for land use mapping in terms of their prediction accuracy. In particular, CapsNet uses much more computing resource and time than other compared algorithms. With the 98.09% validation accuracy, CNN is able to map land per square kilometre in less than one second. In the case study, CNN further demonstrates its

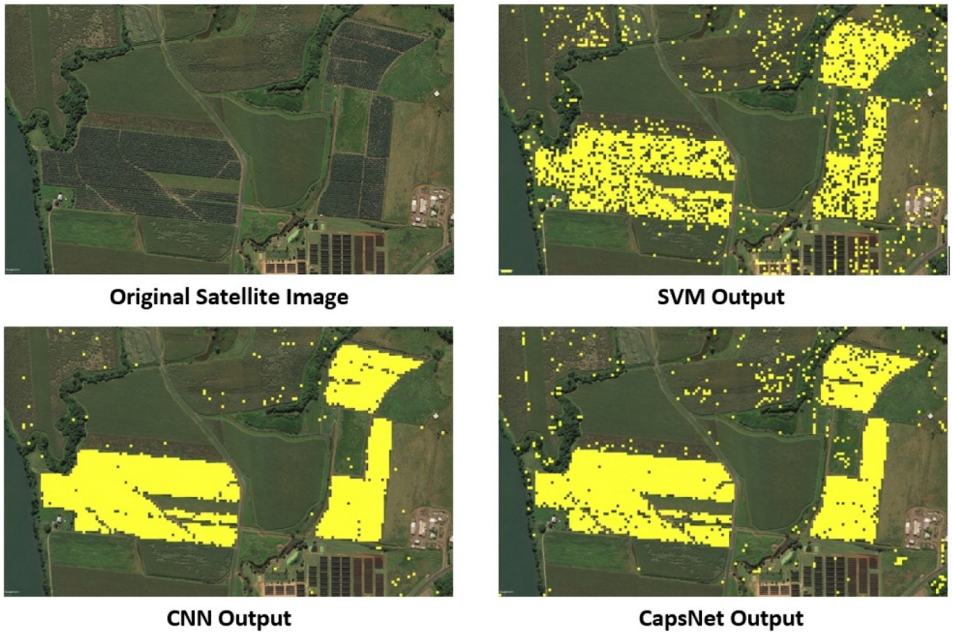

Fig. 12. An original image and its classified images by the three algorithms.

Fig. 13. Apply the classification to marking grass area.

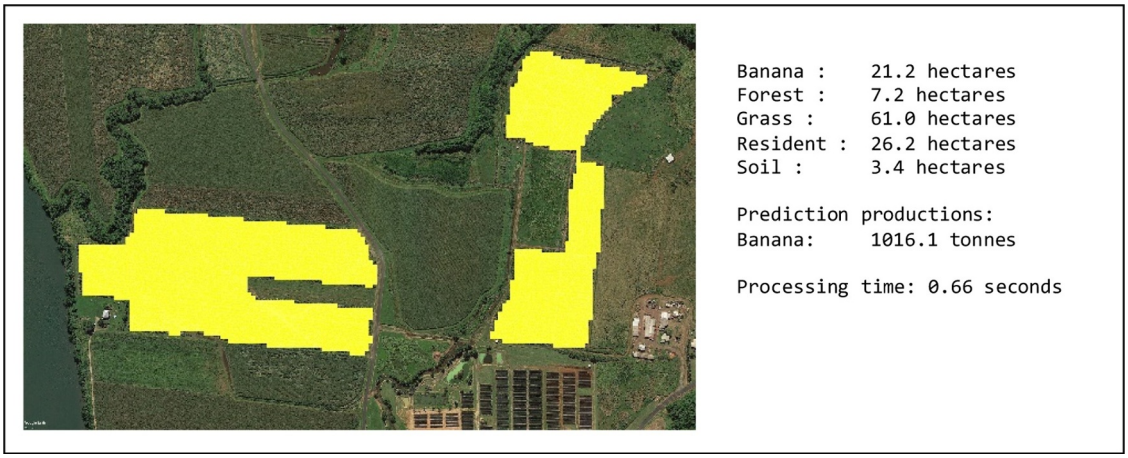

Fig. 14. An output marked image and the statistics of its land use.

capacity of mapping satellite images into different land uses.

In this paper, we have also presented a modular, extensible system called ML-LUM based on our experiment results. In the system, several ML algorithms have been implemented to classify satellite images or other types of images. The algorithms can be applied to other types of images or data classification. This is because each of these modules meets the requirements of the ease-of-use and functionality of the overall application. It can be an effective tool for land use mapping. The design of the modular structure enables to be flexible enough to accommodate different requirements. The future work will focus on further improving the system of online farmland use mapping.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (Nos. 61877020). We acknowledge CSU's Spatial Data Analysis Network (SPAN) Team and Mr Simon McDonald for providing valuable support including data, tools and software.

Cloud implementation of this research was supported by use of the NeCTAR Research Cloud, a collaborative Australian research platform supported by the National Collaborative Research Infrastructure Strategy.

References

[1] The Australian Land Use and Management Classification Version 8; A.B.o.A.a.R.E.a.S. (ABARES), Editor. 2016.
[2] Department of Science, Information Technology and Innovation (DSITI), DSITI89267 Land use mapping using machine learning – specification. 2017.
[3] Department of Science, Information Technology and Innovation (DSITI), DSITI89267 Building machine learning capability: land use mapping request for quote. 2017.
[4] K. Johansen, et al., Mapping banana plants from high spatial resolution orthophotos to facilitate plant health assessment, Remote Sens. (Basel) 6 (12) (2014) 8261–8286.
[5] G. Cheng, J. Han, A survey on object detection in optical remote sensing images, ISPRS J. Photogramm. Remote Sens. 117 (2016) 11–28.
[6] G. Chen, G.J. Hay, B. St-Onge, A Geobia framework to estimate forest parameters from lidar transects, quickbird imagery and machine learning: a case study in Quebec, Canada, Int. J. Appl. Earth Observ. Geoinf. 15 (2012) 28–37.
[7] M. Wieland, M. Pittore, Performance evaluation of machine learning algorithms for urban pattern recognition from multi-spectral satellite images. 2014.
[8] T.E. Parece, et al., Remote sensing in an arcmap environment, Classif. Landsat Image (Supervised) (2014).
[9] C. Makowski, C.W. Finkl, H.M. Vollmer, Geoform and landform classification of continental shelves using geospatially integrated ikonos satellite imagery, J. Coast. Res. 33 (1) (2016) 1–22.
[10] L. Ma, et al., A review of supervised object-based land-cover image classification, ISPRS J. Photogramm. Remote Sens. 130 (2017) 277–293.
[11] T. Blaschke, Object based image analysis for remote sensing, ISPRS J. Photogramm. Remote Sens. 65 (2010) 2–16.
[12] A. Mordvintsev, K. Abid, Opencv-python tutorials documentation. 2014; Available from: https://media.readthedocs.org/pdf/opencv-python-tutroals/latest/opencv-python-tutroals.pdf.
[13] I.H. Witten, et al., Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann, 2016.
[14] S. Shalev-Shwartz, S. Ben-David, Understanding Machine learning: From theory to Algorithms, Cambridge university press, 2014, p. 259.
[15] A. Appice, et al., Sampling training data for accurate hyperspectral image classification via tree-based spatial clustering, Proceedings of the Conference of the Italian Association for Artificial Intelligence, Springer, 2017.
[16] P.D.A.U. Python, M. Swamynathan, Mastering machine learning with python in six steps.
[17] F. Medhat, D. Chesmore, J. Robinson, Masked conditional neural networks for environmental sound classification, Proceedings of the International Conference on Innovative Techniques and Applications of Artificial Intelligence, Springer, 2017.
[18] N. Jean, et al., Combining satellite imagery and machine learning to predict poverty, Science 353 (6301) (2016) 790–794.
[19] I. Goodfellow, et al., Deep Learning 1 MIT press, Cambridge, 2016.
[20] S. Sabour, N. Frosst, G.E. Hinton, Dynamic routing between capsules, Adv. Neural Inf. Process. Syst. (2017) 3859–3869.
[21] G. Hinton, N. Frosst, S. Sabour, Matrix capsules with EM routing. 2018.
[22] Y. Li, et al., The recognition of rice images by UAV based on capsule network, Cluster Comput. (2018) 1–10.
[23] Y. LeCun, C. Cortes, C. Burges, MNIST handwritten digit database. AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist, 2010.**2**.
[24] A. Krizhevsky, V. Nair, G. Hinton, The CIFAR-10 dataset. online: http://www. cs. toronto. edu/kriz/cifar. html, 2014.
[25] A. Jaiswal, W. AbdAlmageed, P. Natarajan, CapsuleGAN: generative adversarial capsule network. arXiv preprint arXiv: 1802.06167, 2018.