Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.7657031

2D-CNN BASED DEEP LEARNING MODEL FOR MULTI LABEL LAND COVER CLASSIFICATION

B.Ramesh Naidu[1], Chinta Someswara Rao[2], K.V.L.Bhavani[3]*, M.Jayanthi Rao[4]
[1]Department of Information Technology, Aditya Institute of Technology and Management(AITAM),Tekkali, K.Kotturu - 532 201, Andhra Pradesh, India
[2]Department of Computer Science and Engineering, SRKR Engineering College, Bhimavaram, Andhra Pradesh, India-534204
[3]Department of Electronics and Communication Engineering, Aidtya Institute of Technology and Management,Tekkali, K.Kotturu - 532 201,Andhra Pradesh, India
[4]Department of CSE, Aditya Institute of Technology and Management(AITAM),Tekkali, K.Kotturu - 532 201, Andhra Pradesh, India
*Corresponding author email: kvlb2003@gmail.com

Abstract
Multi-label land cover classification is the process of classifying the land into different classes based on the type of land. A well-defined land classification is very useful, as we can find out the type of land with the satellite images of that particular area, which helps the users decide whether the land is suitable for their purposes or not. Several research efforts using machine learning techniques have been underway to accurately label the land, but there is still room for improvement. To improve the classification accuracy, in this paper we propose a 2D convolutional neural network (CNN) model with convolution and max-pooling, and that is fully connected, with dense layers. The proposed 2D-CNN model consists of twoConv2D layers, a flattened layer and two-dense layers. The proposed network comprises of 5,329,361 parameters/nodes out of which 5,329,169 and 192 are trainable and non-trainable parameters/nodes respectively. We classify the images into 17 labels such as agricultural, airplane, baseball, diamond, beach, buildings, chaparral, dense residential etc., with 2D-CNN model with 80% accuracy. We classified the land in this research using the 2D-CNN model. We examined 2100 satellite images to evaluate the model's performance. The experimental study shows that multiple labels in remote sensing images is predicted most accurately by the proposed CNN model. It distinguishes trees, pavement, water, and other labels in remote sensing images considerably well. The tabulated results show that a state-of-the-art analysis was done to learn varying land cover classification models. In the future, we want to investigate graph-based multi-label classifiers and design more effective algorithms for remote sensing image annotation.
Keywords: Classification, Land Cover, Deep Learning, CNN

Introduction
The classification of land use and land cover, often known as LULC, is an essential variable in earth monitoring since it has been demonstrated to have a direct association with both human activities and the physical environment in general. Because of the rapid expansion of human

societies, numerous sorts of activities have become more intense, which has resulted in a consistent and visible influence on LULC [1].

In the most recent few years, scientists working in remote sensing have become increasingly interested in using machine learning classification methods for LCLU mapping[2]. In general, the results of previous research show that machine learning classifiers produce more accurate and reliable products than traditional statistical classifiers (such as minimum distance, maximum likelihood, and parallelepiped classifiers), particularly if there is a large quantity of training data available [3]. This is especially true in situations where there is a large amount of available data. In spite of the fact that unsupervised classifiers are predicated on similarity measurements such as color information, the distance between neighboring pixels, and so on, these may not be accurate in practice [4].

The classification of remotely sensed images could be made more accurate and time-efficient thanks to deep learning's capabilities. Deep learning's strengths include its ability to handle data with a high dimensionality and to map classes with very complex characteristics. Additionally, deep learning is able to accept a wide variety of input predictor data and does not make assumptions about the distribution of the data (i.e., it is nonparametric) [5]. Numerous investigations have come to the conclusion that using these procedures typically results in a high level of accuracy. For the purpose of this investigation, we classified the land using a two-dimensional convolutional neural network (2D CNN). This methodology was chosen because of its utility and significance.

Literature survey

Rohit Gandikota and Deepak Mishra[6] gave an elaborate explanation aboutCNNs. The basic working of the CNN is to back-track and find the important pixels of an images and to make decisions. We aim to locate the nodes from the previous layer that activate it the most when given a node. This process is done from final output layer to get the important pixels of animage.CNN is made up of blocks,namely‘convolution’, ‘pooling’ and ‘fully connected’ blocks. The CNNbacktracked through these layers to determine regions in the form of important pixels.

SambitMahapatra[7]explains about the 2DCNN for digital recognition.The use a large number of datasets and applying the 2D CNN first they are preprocessing the data and then building the model. Here 2D CNN is designed with keras and tensor flow .

James McDormett[8], proposedan image classification model for the EUROSAT satellite images. It contains 10 different land classes with 64x64 pixels images from theSentinel-2 satellite. The model used transfer learning, which is the process of taking a model and training it on large data and applying it to asmaller data set. The model divided the data set into 70% training data and 30%testing data, then applied VGG16’s architecture and trained the model with Keras CNN. Then the train and validation data were plottedwith accuracy and epochs as labels. Finally the classification map of the lands which contains forest, river, highway, residential etc., were obtained as classes.

Praveen Kumar Mallupattu and Jayarama Reddy SreenivasulaReddy[9] conducted astudy mainly on the changes of the fields like urbanization, industrialization and other fields over the past decades in the area Tirupati. For this they used the two kinds of data i.e topographic map

and remote sensing data. By using the given techniques they have divided the area into different categories like agriculture area, built up area, plantation area and others. This study mainly focused on the LULC changes using the remote sensing and Geographical Information Systems (GIS). Based on the techniques they have allocated the area to particular extent. The LULC changes in Tirupati makes the people better understand the surroundings.

Adam Johnson[10] conducted research on the land and its classification as it relates with transportation. Here they have included different spatial patterns. In this transportation they have used remote sensing and geospatial technologies and requisite analysis. There were three goals of this project.First they have analysed the land trends andsecondly they have assessed the present land use and land cover with the previous LULC using remote sensing techniques. Finally they calculated the accuracy and robustness of the final model.

S.L.SenthilLekha [11] explains the LULC features of the Kanyakumari district and it contains the features and the classification of the land. This was conducted to help understand the population and dynamic of life. The classification of land mainly lead to the increase in the vegetation and leads to profits for the land. In this classified LULC features divide the areas into agriculture area, water area, building area and other areas. The classification of land is mainly done based on accuracy.

Oliver Sefrinet al., [12] performed the deep learning model in classifying the lands with eight labels and the data is satellite imagery data which describes fully convolutional neural network (FCN) combined with long short-term memory(LSTM).In these LSTM provides the best results with high accuracy. Manual Campos-Terbner, et.al.,[13] and the deep learning techniques mostly provide the accurate results from parameter estimation to image and video classification in the present world and usage of the techniques is also increasing rapidly. Remote has sensing derived data from spatial and spectral domain of the data. Deep learning uses remote sensing techniques mainly divided into two kinds, namely CNN and recurrent neural networks(RNN). In this CNN model they classify the land according to the land parcel identification system[13]. The land was divided and 60% is allotted to natural vegetation and 24%of the land allocated to the permanent crops and 16%of the land allocated to the annual crops.

Chiman Kwan et al., [14], explained the hyper spectral image land classification with two types of CNN models. Hyper spectral images increase the number of spectral channels which provide better information of the images which helps in improving the performance of the land classification. They used a customized CNN method. The dataset contains 15 classes and each class is mapped to different colours. These gave the best results when compared to support vector machine(SVM).

Harish Chandra Verma et.al.,[15],showedthe process of evaluating the supervised, unsupervised and hybrid classifiers. These are the common techniques used for satellite image classification. They compared supervised classifier Artificial neural network(ANN), SVM and unsupervised classifier K-Means and also knowledge based classifier decision tree. These classifiers here are used for fruit trees classification so here fruittrees satellite images are given as input to the models. Finally they concluded that the decision tree classifier performed best inoverall accuracy .

Methods

The proposed methodology has been organized into different phases. Figure 1 shows each phase in detail.

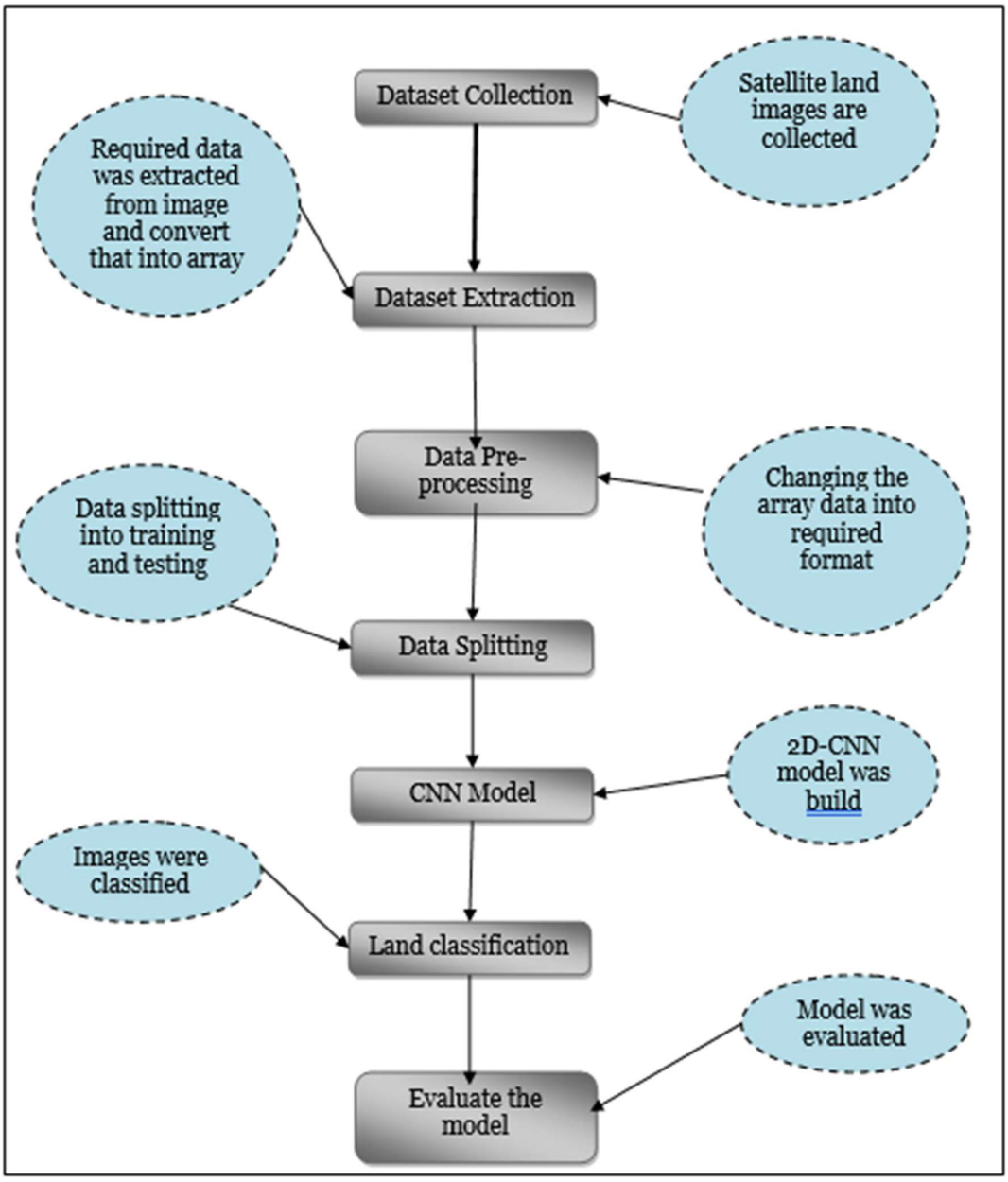

Figure 1.System architecture.CNN; convolutional neural network.

Data collection and extraction

Dataset is downloaded fromthe weegee database[16] (see Underlying data). The dataset is named UCMerced. It consists of 2100 images, the images are in RGB (red, green, blue)formatand contains 17 classes and 21 labels. The collected and extracted images are converted into array for further processing.

Data pre-processing

As the images are in RGB format, values are encoded as 8-bit integers so it ranges from 0 to 255, so we have to convert those values and make them to lie between 0 and 1 so we have to convert the image as array and divide with 255.This process is performed image by image. This process is implemented in Python

Data splitting

Dividing the data into training sets plays a vital role in model training. Generally the data is splitin ratio 80:20 or70:30 i.e, 80% of the data is for training and 20% is for testing respectively. This process is implemented in python with sklearn.model_selection.train_test_split() method.

Building the model

This is the main step in the process. In this paper we are using a 2-D CNN model designed using Keras with tensor flow backend.

Keras is a library that provides high-level building blocks for the development of deep learning models. It operates at the model level. It does not handle low-level operations such as tensor products, convolutions, and other similar operations on its own. Instead, it uses a tensor manipulation library that is both specialized and highly optimized to perform these tasks. This library is referred called as Keras' "backend engine." Instead of selecting a single tensor library and tying the implementation of Keras to that library, Keras solves the problem in a modular approach, and numerous different backend engines can be plugged into Keras without any problems. This allows Keras to accommodate a wider variety of use cases. There are now two backend implementations that are available for use with Keras. These are the TensorFlow backend and the Theano backend. In our investigation, we made use of TensorFlow, which is a framework for the open-source manipulation of symbolic tensors that was developed by Google, Inc. We utilized pre-existing modules such as tensorflow.keras, layers, and keras.sequential in order to implement Keras using TensorFlow as a backend.

Figure 2 shows the design of the convolution neural network.A complete CNN stage contains an input layer, convolution layer, batch normalization, pooling layer, flatten layer, dense layer and finally output layer. Images are given as input at input layer, later these inputs are fed to the convolution layer.

The model summary is depicted in Table 1.From the table 1 we understood that proposed model consists of input layer that accepting image of size 148x148, later has batch normalization (it will split the data into mini batches, instead of performing operation on whole data that will reduce the computation time), then has max pooling (it will consider the max value from the matrix, so that reduce the dimension), then has another convolution layer, batch normalization and max pooling layers, later has flatten (it will convert whole image data into single vector), later has dense layer and then finally output layer that consists of 17 nodes that means it will classify the image into 17 classes

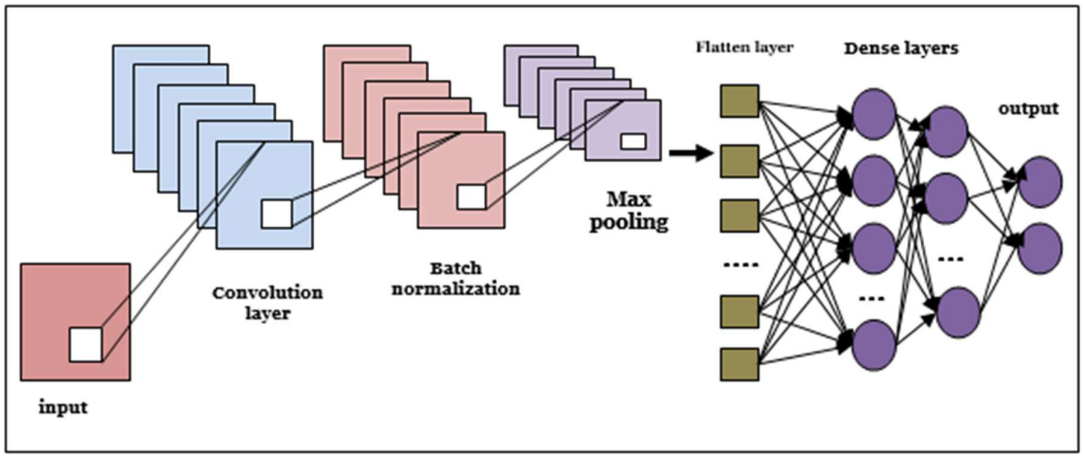

Figure 2.Convolution neural network.

Table 1.Modelsummary.

Model: "sequential"		
Layer (type)	**Output shape**	**Parameters#**
conv2d (Conv2D)	(None, 148, 148, 32)	896
batch_normalization	(BatchNo (None, 148, 148, 32)	128
max_pooling2d (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_1 (Conv2D)	(None, 72, 72, 64)	18496
batch_normalization_1	(Batch (None, 72, 72, 64)	256
max_pooling2d_1	(MaxPooling2 (None, 36, 36, 64)	0
flatten (Flatten)	(None, 82944)	0
dense (Dense)	(None, 64)	5308480
dense_1 (Dense)	(None, 17)	1105
Total params: 5,329,361		
Trainable params: 5,329,169		
Non-trainable params: 192		

In the convolution layer, the value of a neuron v_ij^x at position x of the jth feature map in the ith layer is calculated with equation 1 and 2.

$$v_{ij}^{x} = g\left(b_{ij} + \sum_{m}\sum_{p-0}^{P_i-1} w_{ijm}^{p} v_{(i+1)m}^{x+p}\right) - - - (1)$$

$$g(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} - - - (2)$$

Where the feature map in the preceding layer ((i − 1)th layer) that is related to the present feature map is indexed by m. The weight of position p related to the mth feature map is w_ijm^p. p is the kernel's breadth toward the spectral dimension, and b_ij is the bias of the ith layer's jth feature map.

By re-centering and re-scaling the inputs to the layers, batch normalization is a method that can make CNN run more quickly and with more consistency. Instead of doing it all at once with the complete data set, it is done in smaller batches. It helps to expedite training and utilize better learning rates, which in turn makes learning much simpler. This helps to stabilize the learning process and considerably decreases the number of training epochs that are required to develop deep networks.

Let us consider mini batch as B of size m for the training set. Then mean and variance of B is calculated with equation 3 and 4.

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \quad ---(3)$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_B)^2 \quad ---(4)$$

For each layer of the network, batch normalization is calculated with equation 5

$$x_i^k = \frac{x_i^k - \mu_B^k}{\sqrt{\sigma_B^{(k)2} + \epsilon}} \quad ---(5)$$

Where k∈[1,d]andi∈[1,m]

The feature maps' dimensions are reduced by using pooling layers. Thisis calculated with equation 6.

$$Pooling = \frac{n_h - f + 1}{s \times (n_w - f + 1)/s \times n_c} \quad ---(6)$$

where n_h is the feature map's height, n_w is its width, n_c is its number of channels, f is the size of the filter, and s is length of stride. Stride is a parameter of the neural network's filter that modifies the amount of movement over the image

Max pooling

It is an operation known as pooling that chooses the most significant element from the section of the feature map that has been processed by the filter. As a result, the output of the max-pooling layer would be a feature map that contained the most prominent features of the prior feature map.It has been shown in Figure 3(a). It is calculated with equation 7.

$$MaxPool = \underset{N\times1}{^{max}}\big(Pooling(i.ecalculatedineqn.6)\big) \quad ---(7)$$

Average pooling

Average poolingcomputes the average of the elements present in the region of feature map covered by the filter giving the average of features present in a patch.It has been shown in Figure 3(b). It is calculated with equation 8.Here the numbers are image pixel values.

$$AVGPool = \underset{N\times1}{^{AVG}}\big(Pooling(i.ecalculatedineqn.6)\big) \quad ---(8)$$

Figure 3(a).Max Pooling

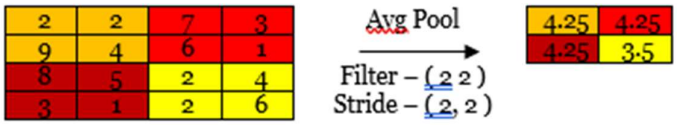

Figure 3(b).Average (AVG)Pooling

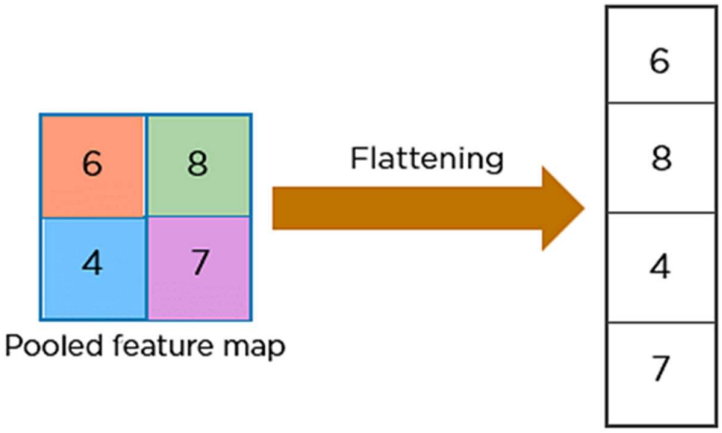

Figure 4.Flatten operation

Flatten layer

It is used to convert all of the generated two-dimensional arrays from pooled feature maps into a single extensive linear vector that is continuous throughout its entirety. Flattening the output of the convolutional layer allows for the construction of a single feature vector that is quite long. The image is classified based on the input provided by the flattened matrix, which is given to the fully connected layer. This operation is performed with tensorflow.keras.layers.Flatten() from the TesorFlow library.The operation is shown in Figure 4.

Dense layer

The term comes from the fact that each neuron in this layer receives information from all of the neurons in the layer below it, making it the most fundamental layer of neurons. It does this by determining an image's identity based on the output of the convolution layer. Output at this layer is calculated with equation 9.

$$\text{output} = \text{activation}(\text{dot}(x_i, \text{kernal}) + \theta) \quad ---(9)$$

Here we also use the activation function which can be applied at middle or at the end which helps to decide whether the neuron has to pass the information as input to the next neuron or end the process by activating it. We can use any type of activation function. Here we are using sigmoid activation function[10]. It keeps the values between 0 and 1.This means it is easy to identify whether a class of land is present or not. It is calculated using equation 10.

$$\text{sigmoid activation function} = \frac{1}{1+e^{-outp \quad at\ the\ hidden\ neuron}} \quad ---(10)$$

here e is Euler number that is 2.17 and $output\ at\ the\ hidden\ neuron = \sum_m \sum_{p-0}^{p_i-1} w_{ijm}^{p} v_{(i+1)m}^{x+p}$, and it was described in equation 1.

Compiling and evaluating the model

After the model is built,the model was trained with rain data and after building the model we compile the model by using the parameters described.

Optimizer

Optimizers are used to change the parameters of our DNN(deep neural network) which includes weights, learning rate etc. So that it reduces the loss. Optimizers define how we can change our learning rates or weights to reduce the loss and increase accuracy. Here we used gradient descent optimizer.

Loss

One of the most important aspects of deep learning neural networks is the concept of loss. It is utilized in the process of calculating the error rate that our model generates in its predictions. The loss function is the approach that is used to calculate loss. Through the process of changing the weights of the neural network, the gradient descent method contributes to a reduction in the loss. Using the gradient at the current position, the Gradient Descent Algorithm calculates the next point to be used, scales it (by a learning rate), and then iteratively subtracts the resultant value from the current position (makes a step). Because we want to make the function as little as possible, it will subtract the value (to maximise it would be adding). This can be measured with equation 11.

$$p_{n+1} = p_n + \eta \nabla f(p_n) - - - (11)$$

Here p_{n+1} *is next point and* p_n *is present point and* $\nabla f(p_n) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial n} \end{pmatrix}$

There is an important parameter thatηthat regulates the step size by scaling the gradient. It is known as learning rate in machine learning and has a significant impact on performance. The approach is shown here

Gradient Descent method's steps are:
1.select a starting point
2.At this point calculate gradient
3.make a scaled step in the opposite direction to the gradient
4.repeat steps 2 and 3 until one of the criteria is met:
4.1.maximum number of iterations reached
4.2.step size is smaller than the tolerance.

Results

In this paper total of 2100satellite images of size 150× 150 are collected and used. These data had 17 classes with 21 labels. Some of the images are depicted in Figure 5 [16]. From these images 80% (1680) are used for training and 20%(420) images are used for testing. This splitting pseudo code is shown in Table 2.

Table 2. Train and Test split pseudo code

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

Here X input i.e array of image; y is output label
```

Model training process pseudo code is depicted in Table 3.

Table 3 Training process pseudo code

```
model.fit(X_train, y_train, epochs=50, validation_data=(X_test, y_test))
```

We observed from the loss and accuracy curves (Figure 6) how the model is performing over the training and testing data. Loss describes the rate of wrong predictions that our model makes and accuracy of the measurement of the model predicting the classes correctly. Here loss is decreasing constantly, and accuracy is increasing which suggeststhe modelperformswell. Here we are using 100 epochs, at each epoch we are calculating how loss and accuracy are changing over the data. At the final epoch we observe 80% accuracy.From Figure 6, it is also observed that the proposed CNN model avoids over fitting because training and testing accuracy are almost equal.

Testing images

For testing we are taking the external test images and passing them through our model.
The testing process pseudo code is depicted in Table 4. Here we given the airplane as input for testing the model. Our model predicts the 21 labels out of which we just display the top three labels because they are more relevant. Our model predicted data will be stored in a numpy array of proba, that contains the probability scores among the training images and the present testing image. From this proba array we displayed the top three images.

Table 4 testing process pseudo code

```
img = image.load_img('../input/test-images/airplane_439.jpg',target_size=(400,400,3))
img = image.img_to_array(img)
img = img/255

classes = np.array(train.columns[2:])
proba = model.predict(img.reshape(1,400,400,3))
top_3 = np.argsort(proba[0])[:-4:-1]
for i in range(3):
   print("{}".format(classes[top_3[i]])+" ({:.3})".format(proba[0][top_3[i]]))
plt.imshow(img)
```

The tested prediction results are shown in Figure 7. From Figure 7(a), it is observed that, the proposed model predicted the labels airplane, runway, and pavement, originally, it contained an airplane, a runway, and pavement, so in this case, we said that both were equal.

In the image 7(b),which contains cars, trees, and pavement, our model also predicts those cars, trees, and pavement. In the image 7(c), which contains grass, trees, and a highway, our model also predicts that there will be grass, trees, and a highway. In the image 7(d), which contains a mobile-home, trees, and pavement, our model also predicts a mobile home, trees, and pavement. In these four images, our model prediction exactly matches actual classes.

Whereas coming to Figure 8, test image 8(a) contains water, agricultural and trees, but our model predicts that water, which is equal to the actual class, mobile homes, and sand are not equal to the actual classes.In image 8(b), there are freeway, cars, and pavement, but our model predicts that freeway, cars are equal to actual classes, while trees are not equal to an actual class.From these two cases, we find that the trained network properly predicts most of the labels of the images tested. However, there are certain instances where it fails to detect the presence of specific items.

Figure 5. Sample satellite images

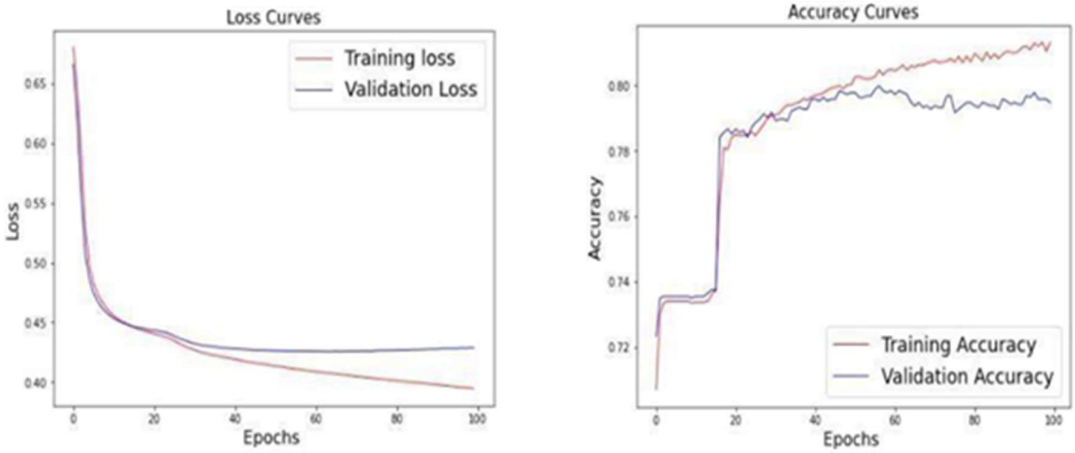

Figure 6.Loss and accuracy graphs

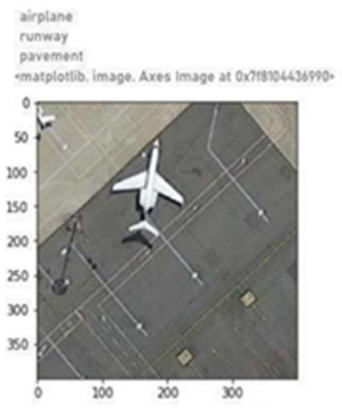

Figure 7(a).

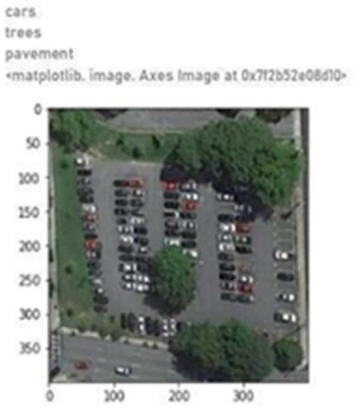

Figure 7(b).

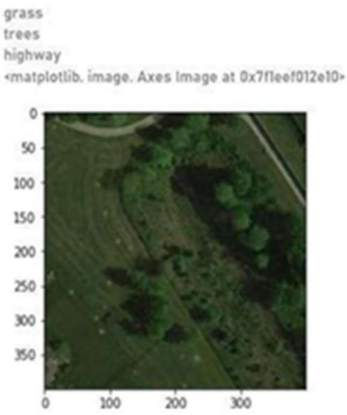

Figure 7(c).

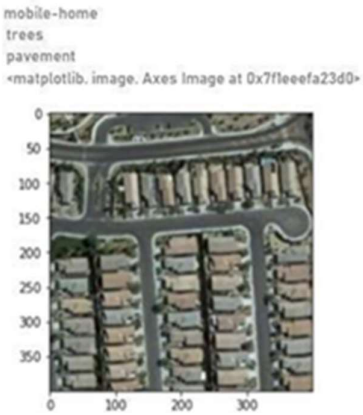

Figure 7(d).

Figure 8(a).

Figure 8(b).

Figure 8. Mismatched images

Conclusions

In this paper, land cover classification is performed using deep learning technology with the 2D-CNN method. Here we used a dataset named UCMerced, which contains 2100 images. The images are observed by the satellite Sentiel-2, and the dataset is classified into 17 different classes with 21 labels. The model was trained with the dataset and obtained an accuracy of 80%. This model can be used by any business person or other who has to identify the type of land for their construction, farming, or other land use based on their requirements. The proposed approach will reduce manpower requirements and reduce the time consumed.

Data availability

All source data within this article is taken from a weegee database [16]. This can be found at http://weegee.vision.ucmerced.edu/datasets/landuse.html. To gain access to this database you must

Software availability

Source code available from:https://github.com/someshchinta/multi_lable_classification
Archived source code at time of publication:https://doi.org/10.5281/zenodo.7299746
License:Apache 2.0

Competing interests

The authors declare that they have no conflict of interest.

Grant information

This research was not funded

References

Petropoulos, G. P., Partsinevelos, P., &Mitraka, Z. , “Change detection of surface mining activity and reclamation based on a machine learning approach of multi-temporal Landsat TM imagery”, Geocarto International, 28(4), pp.323-342, 2013, https://doi.org/10.1080/10106049.2012.706648

Shih, H., Stow, D. A., & Tsai, Y. H. , “Guidance on and comparison of machine learning classifiers for Landsat-based land cover and land use mapping. International Journal of Remote Sensing”, 40(4), pp.1248-1274, 2018. https://doi.org/10.1080/01431161.2018.1524179

Schneider, A. “Monitoring land cover change in urban and peri-urban areas using dense time stacks of Landsat satellite data and a data mining approach”, Remote Sensing of Environment, 124, pp.689-704, 2012. https://doi.org/10.1016/j.rse.2012.06.006

Sivagami, R., Krishankumar, R., & Ravichandran, K. S., “A Comparative Analysis of Supervised Learning Techniques for Pixel Classification in Remote Sensing Images”, 2018 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), pp.1-4, 2018. https://doi.org/10.1109/WiSPNET.2018.8538518

Maxwell, A. E., Warner, T. A., & Fang, F., "Implementation of machine-learning classification in remote sensing: An applied review", International Journal of Remote Sensing, 39(9), pp.2784-2817, 2018. https://doi.org/10.1080/01431161.2018.1433343

Rohit Gandikota, Deepak Mishra, "How You See Me: Understanding Convolutional Neural Networks", TENCON 2019-2019 IEEE Region 10 Conference(TENCON), pp. 2069-2073, 2019.

SambitMahapatra,"A simple 2D-CNN for MNIST digit recognition",2018.

James Mc Dormett, "Land Cover Classification Using Keras", 2020.

Praveen Kumar Mallupattu,Jayarama Reddy SrinivasuluReddy,"Analysis of Land Use/Land Cover Changes Using Remote Sensing Data and GIS at an Urban Area,Tirupati,India", The Scientific World Journal, vol. 2013, pp.1-6, 2013.

Adam Johnson,"Remote sensing, GIS, and land use and land cover mapping along the 1-10 corridor", American Society for Photogrammetry and Remote Sensing Environmental Protection Agency, 2002.

S.L.SenthilLekha, "Classification and Mapping of Land Use Land Cover change in Kanyakumari district with Remote Sensing and GIS techniques", International Journal of Applied Engineering Research, Vol.13, issue.1, pp. 158-66, 2018.

Oliver Sefrin,Felix M Riese,Sina Keller , "Deep Learning for Land Cover Change Detection", Remote Sensing,Vol.13, issue.1, pp. 1-27, 2021.

Manual Campos-Terbner, Fransisco Javier Gracia-Haro, Clement Atzberger, Maria Amparo Gilabert,"Understanding deep learning in land use classification based on Sentinel-2 time series", Scientific reports, Vol.10,issue.1,pp.1-2, 2020.

Chiman Kwan, BulentAyhan, BenceBudavari, YanLu, Daniel Perez, Sergio Bernabe 3,Antonio Plaza, "Deep Learning for Land Cover Classification Using Only a Few Bands" , Remote Sensing, vol.12, issue.12, pp.1-17, 2020.

Harish ChandraVerma,ShailendraRajan,TasneemAhmed, "A Review on Land Cover Classification Techniques for Major Fruit Crops in India-Present Scenario and Future Aspects", International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM), Amity University Rajasthan, Jaipur-India, pp.1-6, 2019.

Yi Yang and Shawn Newsam, "Bag-Of-Visual-Words and Spatial Extensions for Land-Use Classification," ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS), 2010.