Multilabel land cover aerial image classification using convolutional neural networks

by Roy Setiawan

Arabian Journal of Geosciences (2021) 14:1681
https://doi.org/10.1007/s12517-021-07791-z

ORIGINAL PAPER

Multilabel land cover aerial image classification using convolutional neural networks

Razia Sulthana Abdul Kareem[1] · Anil Gandhudi Ramanjineyulu[2] · Regin Rajan[3] · Roy Setiawan[4] · Dilip Kumar Sharma[5] · Mukesh Kumar Gupta[6] · Hitesh Joshi[7] · Ankit Kumar[6] · Haritha Harikrishnan[8] · Sudhakar Sengan[9]

Received: 21 April 2021 / Accepted: 10 June 2021

Abstract

Classifying the remote sensing images requires a deeper understanding of remote sensing imagery, machine learning classification algorithms, and a profound insight into satellite images' know-how properties. In this paper, a convolutional neural network (CNN) is designed to classify the multispectral SAT-4 images into four classes: trees, grassland, barren land, and others. SAT-4 is an airborne dataset that captures the images in 4 bands (R, G, B, infrared). The proposed CNN classifier learns the image's spectral and spatial properties from the ground truth samples provided. The contribution of this paper is three-fold. (1) A classification framework for feature extraction and normalization is built. (2) Nine different architectures of CNN models are built, and multiple experiments are conducted to classify the images. (3) A deeper understanding of the image structure and resolution is captured by varying different optimizers in CNN. The correlation between images of varying classes is identified. The experimental study shows that vegetation health is predicted most accurately by the proposed CNN models. It significantly differentiates the grassland vegetation from tree vegetation, which is better than other classical methods. The tabulated results show that a state-of-the-art analysis is done to learn varying land cover classification models.

Responsible Editor: Syed Hassan Ahmed

This article is part of the Topical Collection on *Data Science for Ocean Data Visualization and Modeling*

✉ Sudhakar Sengan
sudhasengan@gmail.com

Razia Sulthana Abdul Kareem
razia@dubai.bits-pilani.ac.in

Anil Gandhudi Ramanjineyulu
anilgrcse@gmail.com

Regin Rajan
regin12006@yahoo.co.in

Roy Setiawan
roy@petra.ac.id

Dilip Kumar Sharma
dilipsharmajiet@gmail.com

Mukesh Kumar Gupta
mukeshgupta@skit.ac.in

Hitesh Joshi
hr.hitesh@rediffmail.com

Ankit Kumar
iiita.ankit@gmail.com

Haritha Harikrishnan
hharikrishnan@ud.ac.ae

[1] Department of Computer Science, Birla Institute of Technology and Science-Pilani, Dubai Campus, UAE

[2] School of Computer and Information Sciences, University of Hyderabad, Gachibowli 500046, Telangana, India

[3] Department of Computer Science and Engineering, Adhiyamaan College of Engineering, Tamil Nadu Hosur 635109, India

[4] Department Management, Universitas Kristen Petra, Jawa Timur, Indonesia

[5] Department of Mathematics, Jaypee University of Engineering and Technology, Guna473226, Madhya Pradesh, India

[6] Department of Computer Science & Engineering, Swami Keshvanand Institute of Technology, Management & Gramothan, 302017, Jaipur, Rajasthan, India

[7] Bhagwan Arihant Institute of Technology, Bhagwan Mahavir University, 395017, Surat, Gujarat, India

[8] Department of College of Engineering and Information Technology, University of Dubai, Dubai, UAE

[9] Department of Computer Science and Engineering, PSN College of Engineering and Technology, Tirunelveli627152, Tamil Nadu, India

Published online: 05 August 2021

Keywords Convolutional neural network · Confusion matrix · Image classification · Land cover detection · Remote sensing · Stochastic gradient descent

Introduction

Meteorologists research satellite images to forecast the behavior of the earth's atmosphere or earth's crust. Remote sensing (RS) images are the images captured by satellites, and these images are captured with the property of the earth's surface, reflecting certain radio bands in the electromagnetic spectrum. Advanced baseline imager (ABI) is an instrument in the satellite that measures the reflected or emitted energy. ABI captures visible, infrared, and microwave electromagnetic portions of the spectrum. The visible and infrared spectrums are considered to have more energy and are the brightest part of the spectrum. The visible spectrum (0.4 micrometers to 0.7 micrometers) captures colors ranging from violet to red (VIBGYOR), and the wavelength of color increases from violet to red. The infrared spectrum (1 millimeter to 750 nanometers) includes near-infrared, mid-infrared, and infrared. The reflected signals reach ABI as photons are converted to electrical signals and then to digital signals. The ABI's information is transmitted as radio waves to the ground and then processed by antennas to reform the original image. Neglect of minutiae of color bands in visible spectrum may lead the ABI to lose the "RGB" of the "composite color imagery." Hence, every image captured by satellite will be measured with the set of spectrum bands it covers.

To process the composite color imageries and detect the patterns, machine learning algorithms like k-NN (k-nearest neighbor), SVM (support vector machines), SVM with kernels like RBF (radial basis function), HMM (hidden Markov models), DE (differential equations), and MRF (Markov random field) can be applied. However, deep learning algorithms like CNN (convolutional neural network) and deep CNN capture the image's high dimensional features. The deep learning algorithms understand the pattern of raw pixels in the image, the image's spectral and spatial features, and several deep learning algorithms applied to classify land cover images and vegetation images in contemporary research works. Machine learning algorithm like random forest, one of the best ensemble learners, is applied for remote sensing image classification in many articles (Pal 2005; Puissant et al. 2014; Rodriguez-Galiano et al. 2012). It is applied (Juel et al. 2015) to classify the aerial orthophoto mosaic imagery that includes red, green, blue, and near-infrared bands. The algorithm achieves high performance on applying digital elevation model (DEM) with vegetation images in Denmark's coastal regions. RF is applied to study the vegetation community in Great Britain (Bradter et al. 2011) and is applied in conjunction with feature elimination to analyze Germany and Canada's landscapes (Guan et al. 2013). Savanna ecosystem in Southern Kalahari, Africa, is classified using RF and is proven to effectively handle overfitting and reduce classification error (Mishra and Crews 2014). Though RF is powerful with large datasets and skillfully handles the unbalanced dataset, it is a black box, and the developer has little control over the model, and most of the time, it falls into an overfitting trap.

Another prime model, conditional random fields (Parikh and Batra 2008) (CRF), a discriminative classifier, uses a pattern of undirected graphical models for remote image classification (Sun et al. 2020). The context information and spatial neighborhood are applied for land cover image classification (Albert et al. 2017). A fusion model of CRF with hidden Markov models (HMM) identifies the spatial neighborhood dependencies by estimating the distribution between the images and pixels (Andrejchenko et al. 2019); CRF goes good with Markov random fields (MRF) for context analysis in images (Basu et al. 2015). CRF is quite complex and consumes more time during the phase.

Research questions and answers

Q1. What is the effect of classifying satellite images? Satellite images are highly effective as they are captured from a very long distance. It is challenging to analyze them in minimal time so that immediate measures can be taken post-analysis.

Q2. What impact is created by the proposed work over existing literature work? The current work relies on building a single model and is detailed in the tables in the literature work. However, the proposed method builds a manifold of models for different class labels.

Q3. What are the concerns of existing models? The existing models try to predict the results by fine-tuning the hyperparameters; however, it is obligatory to analyze the pattern in images and build different models with varying optimizers.

Q4. What is the most critical procedure applied in this proposed work? The essential procedure is building the layers of the CNN with varying optimizers and training procedure

Q5. In what strength the model prediction is measured? It is measured as stacked analysis using confusion matrix, treemap showing the prediction value of proposed system vs. ground truth, prediction values of proposed model vs. ground truth (trees, grassland, barren land, and others), training loss, validation loss, training accuracy, and validation accuracy of CNN models for 10 epochs.

Related works

CNN shows significant results in computer vision and has motivated researchers to apply it for image classification. Considerable research work is done in remote sensing image classification using CNN, and a few of them include Basu et al. (2016, 2017), Sherrah (2016), and Yao et al. (Yao et al. 2016). CNN is applied for scene understanding, detecting land cover, and semantic segmentation and classification.

Numerous combinations of deep learning networks are applied to classify the RS images using supervised learning and unsupervised learning methods. Stacked autoencoder (SAE) is applied to extract the high-level features (Chen et al. 2014) from hyperspectral images. Deep autoencoder extracts the spectral-spatial information (Chen et al. 2015) and spectral signatures of the image using CNN. In applying deep autoencoder in Ma et al. (Ma et al. 2016), image classification performance becomes better than SVM. In general, images classified using deep unsupervised learning as the manual identification of features from RS images is arduous. The authors (Hu et al. 2015; Romero et al. 2015) have applied unsupervised deep CNN and local discriminant function to identify pixels and subspace. Table 1 and Table 2 described the recent work done on RS image classification from different perspectives.

Researchers have applied efficient methods to classify an RS image by combining CNN with other algorithms (Basu et al. 2015; Chen et al. 2019; Sameen et al. 2018; Shakya et al. 2020). The dimensionality approach (Basu et al. 2015) combines CNN with autoencoders, extracts features from the image, and applies normalization over it but forsaken to handle sparse areas. As the earth's crust is not flat, the captured RS images should not be considered flat. The author (Shakya et al. 2020) applies polynomial regression and CNN for predicting the storm location from the RS images. The results proved its efficiency, yet flat images that captured the earth's curvature were not handled successfully. Methods like object-based image analysis (or OBIA) plays well with high-resolution images and are applied over a research paper (Sameen et al. 2018) to classify the land cover classes of images (Cheng et al. 2016; Hoberg et al. 2012; Scott et al. 2017).

A couple of image classification methods in literature uses the Siamese network, and this network differs from a conventional network in the following ways: A conventional neural network is trained with some fixed number of class labels, and the addition of any new class label is compelled to train the entire network again. However, in the Siamese network, many identical, similar networks with similar patterns are stacked and require no retraining to add new class labels.

The challenges of the existing systems include the following:

- Images are analyzed on a partition basis to handle sparse representation.
- SiamCRNN or its variants for CD in hyperspectral images fail to handle if newer class labels are added.
- Scaling the architecture for sizeable remote sensing datasets and other data sources such as satellite images and laser scanning point clouds is challenging.

The analysis of the state-of-the-art classification of RS images uses various optimization approaches, and validation measures reveal that neither all of these models can be applied over all the datasets nor a standard model be designated. Hence, it is imperative to choose a distinctive architectural model for a particular dataset to achieve a specific objective. This motivates to proposal manifold of models suitable for classifying specific class labels of images.

This research investigates applying CNN to classify the airborne image patches covering four broad land covers (barren land, grassland, trees or dense vegetation, others) and explores the behavior of distinct CNN architecture in classifying the RS images with high accuracy.

The significant contributions of the paper include the following:

Table 1 The architecture and dataset used

Reference	Method	Architecture	Dataset	Performance measure
(Hu et al. 2015)	CNN	AlexNet, VGGNet, and GoogleNet	21 class land use publicly available dataset	Accuracy
(Juel et al. 2015)	Dynamic CNN	CaffeNet, GoogleNet, ResNet	UC Merced Data	Accuracy
(Khan et al. 2020)	Fully convolutional network	FC5, FC6	ISPRS Vaihingen and Potsdam Benchmark datasets	Accuracy
(Kim et al. 2020)	Deep CNN	37 pixels with all the data sources included. The network consists of five layers (four convolutional layers and one fully connected layer)	ISPRS Benchmark	Precision, recall, accuracy

Table 2 The architecture, findings, concerns, and metrics used

Reference	Objective	Classifier	Technique/approach	Findings	Concerns	Metrics
(Ouyang et al. 2017)	A classification framework DeepSat extracts features from an input image, normalizes them, and feeds the normalized feature vectors to a deep belief network for classification	DBN CNN Stacked denoising autoencoder	DeepSat	DeepSat with DBN outperforms the traditional CNN, DBN, and SDAE DeepSat with random forest also outperforms the traditional random forest	Images can be analyzed on a partition basis to handle sparse representation	Accuracy
(Pal 2005)	A deep learning model is trained and tested with artificially densified and classified storm data for cyclone classification and locating the cyclone vortex and polynomial regression for predicting the path	CNN and polynomial regression	YOLO RetinaNet	YOLO model is suggested for detecting and locating the cyclone R-CNN model is suggested for predicting the location of storm	The flat images fail to impart effect due to curvature of the earth and need to be incorporated separately in the future	Mean square error (MSE), mean difference error (MDE), number of sites of disagreements (NSDs), percentage error (PE), peak signal-to-noise ratio (PSNR)
(Parikh and Batra 2008)	This article presents a robust and general end-to-end network architecture, SiamCRNN, for change detection (CD) inhomogeneous and heterogeneous very high resolution (VHR) image, which combine CNN and RNN to process images	CNN and RNN	SiamCRNN consists of three subnetworks: DSCNN (deep Siamese CNN), MRNN (multiple recurrent neural networks), and FC (fully connected)	Analyzed the change maps using 12 different approaches: (a) MAD, (b) IRMAD, (c) SFA, (d) ISFA, (e) CVA, (f) PCA-K-means, (g) SVM, (h) DSCN, (i) RNN-CD, (j) SiamCRNN (TR), (k) SiamCRNN (GRU), (l) SiamCRNN (LSTM)	SiamCRNN or its variants for CD in hyperspectral images	Recall, precision, OA—overall accuracy, KC—kappa coefficient
(Puissant et al. 2014)	Classify the aerial photographs into seven land cover classes: building, grassland, dense vegetation, waterbody, barren land, road, and shadow	CNN with normalization	OBIA (object-based image analysis	The proposed model could balance generalization ability and training efficiency using advanced regularization techniques such as dropout and batch normalization	Scaling the architecture for sizeable remote 4 using datasets and other data sources such as satellite images and laser scanning point clouds	OA—overall accuracy, KC—kappa coefficient, sensitivity and specificity

- The principal objective is to find the optimal CNN architecture and ideal hyperparameters for classifying airborne images in 4 different perspectives.
- A multi-architecture model of CNN that uses varying classifiers and network structures to accurately predict heterogeneous and homogeneous images irrespective of spatial-spectral features is presented.
- A stacked analysis is made on individual images in a test set using a confusion matrix for each model.
- An extensive performance analysis is done over each model to identify the number of correctly and incorrectly mapped images, thus providing insights to choose an ideal model for certain groups of images.

The chosen dataset contains four classes of images, and so the classification can be done among the four broad land covers. The number of classifications can be increased when the dataset is added with other training images.

The proposed methodology can be applied for classifying high-end remote sensing images, land cover images, spectral and unique images, water vapor imageries, medical images, product images, etc.

Methodology

This section details the dataset, the proposed CNN models, and the network training and testing procedure.

Dataset

The data is from the National Agriculture Imagery Program (NAIP) dataset. It contains 330,000 images covering the Continental United States (CONUS). It contains uncompressed Digital Ortho Quarter Quad tiles (DOQQs). DOQQ is digital aerial images that are orthorectified at a resolution of 1 m. They correspond to the United States Geological Survey (USGS). On average, the image tile width is ~6000 pixels, and height is ~height, and its size would be 200 megabytes. The size of the complete dataset for CONUS is ~65 terabytes.

The images are captured in 4 bands(RGB and near infrared), thus covering different land covers (agricultural areas, densely forested, urban areas, mountainous terrain, rural areas, small to large water bodies, etc.) of the complete state of California. Each land cover class's images are labeled by extracting 28 × 28 sliding window blocks in a nonoverlapping mode and are saved to the dataset. The dataset includes the R, G, B, and I values of the image. The image pattern in the other section has no specific pixel color pattern, and processing them would not help us know the type of image hidden in it. The structure of the dataset is shown in Table 3. Window size 28 × 28 is significant to capture the statistical properties. The conditional class distributions of the training images fit within their class labels with no interclass overlaps. Both the training and test classes are entirely disjoint.

The article (Basu et al. 2015) applies SAT-4 and SAT-6 datasets. The proposed systems have analyzed SAT-4 data in multiple dimensions. A sample tiled image of the SAT-4 dataset is given in Figure 1.

Preprocessing

The training dataset includes digital values that represent RGB format. These values are normalized to avoid abnormal gradients. The normalization (LeCun et al. 1998) in Eq. (1) improves the gradient descent reaching the global minima.

$$P' = (P/max - \mu)/\sigma \tag{1}$$

μ represents the mean, and σ refers to the standard deviation. P' is the normalized pixel value.

A global minimum is where the function takes a minimum value and vice versa for global maxima. When applied to different values of variables, a function might lead to many minima values and few maxima values. It is continuously the aim of the model to pick the correct values of the coefficients/variables, so the maximum value is attained. The maximum value is the one that brings the predicted data significantly closer to the ground truth. The dataset includes the class labels and those that represent the ground truth values of the images.

In many instances, the normalized data will not lead to an overfitting model. Normalization lets on the gradient sloping down faster to global minima. Gradient descent (GD) is an optimization algorithm and a mathematical model that plays around loss function. The mathematical model for loss function is mentioned in Eq. (2), describing the error between the predicted value and target value.

$$cost(W) = \frac{1}{2M} \sum_{i=1}^{M} (pred(x_i) - target(y_i))^2 \tag{2}$$

In order to reduce the loss value or bring it down to a minimal value, the possible option would be to work with their partial derivatives. For specific initial weight values, the partial derivatives would interpret a tangential equation to identify its gradient. The gradient value is the differential value of the loss function given in Eq. (3), (4), and (5).

$$\frac{\partial}{\partial w^i} cost(W) = \frac{\partial}{\partial w^i} \frac{1}{2M} \sum_{i=1}^{M} (pred_w(x_i) - target_w(y_i))^2 \tag{3}$$

$$\frac{\partial}{\partial w^i} cost(W) = \frac{2}{2M} \sum_{i=1}^{M} (pred_w(x_i) - target_w(y_i)) \frac{\partial}{\partial w^i} (w^i x_i - y_i) \tag{4}$$

Table 3 Dataset structure

Dataset	No. of samples	The unsigned integer of 8 bits (unit 8)	Per image pixel
X_Train	400,000	28 × 28 × 4 × 400,000	28 × 28 in 4 channels
Y_Train	400,000	400000 × 4	4 × 1 vector (class label)
X_Test	100,000	28 × 28 × 4 × 100,000	28 × 28 in 4 channels
Y_Test	100,000	100,000 × 4	4 × 1 vector (class label)

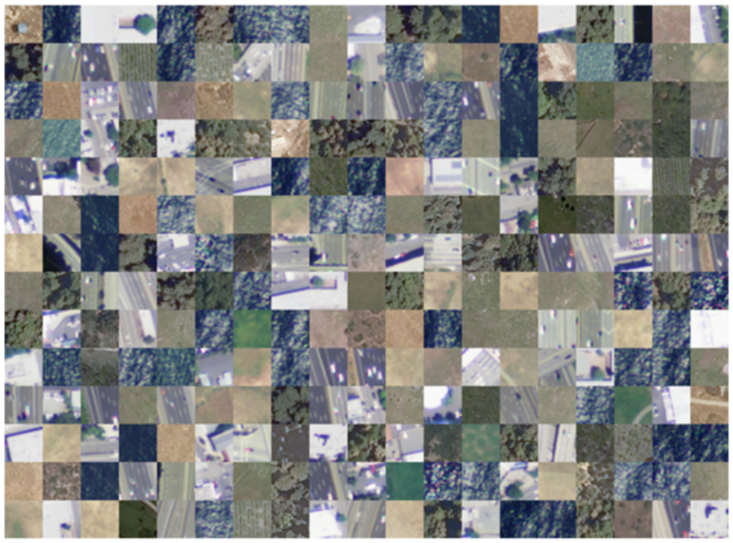
Figure 1 Images of SAT-4 dataset

$$\frac{\partial}{\partial w^i} cost(W) = \frac{1}{M} \sum_{i=1}^{M} (pred_w(x_i) - target_w(y_i))x_i \quad (5)$$

Further, the weights are adjusted as per the calculated gradient values. A configurable hyperparameter (α) is introduced to adjust the weight update. This hyperparameter decides the rate of gradient sloping down toward the global/local minima. The higher the value of α, the gradient slopes faster to minima or might jump to the minima, and the lower the value of α, the gradient takes baby steps to reach the minima following Eq. (6).

$$w^i = w^i - \alpha \frac{\partial}{\partial w^i} cost(W) \quad (6)$$

The hyperparameter (α) is tuned, and the corresponding weights (let say w0, w1, s2) get adjusted accordingly. The loss function mentioned in Eq. (2) is calculated again with the adjusted weights, and the process continues. The loss function saturates when it reaches the global minima, where the gradient neither increases nor decreases and becomes ~0.

$$w^0 = w^0 - \alpha \frac{1}{M} \sum_{i=1}^{M} (pred_w(x_i) - target_w(y_i))x_0 \quad (7)$$

$$w^1 = w^1 - \alpha \frac{1}{M} \sum_{i=1}^{M} (pred_w(x_i) - target_w(y_i))x_1 \quad (8)$$

$$w^2 = w^2 - \alpha \frac{1}{M} \sum_{i=1}^{M} (pred_w(x_i) - target_w(y_i))x_2 \quad (9)$$

The steps from Eq. (7),(8), and (9) are applied in every iteration until a minimal loss value is reached in the training phase. The model is finalized and is applied for the validation dataset.

Convolutional neural network

CNN (Fukushima et al. 1983) in recent years has become more popular as it shows its excellence in image classification (He et al. 2015; Liu et al. 2017), facial recognition (Sun et al. 2014), target object detection (Ouyang et al. 2015), lane detection, or obstacle detection or pedestrian detection (Ouyang et al. 2017). Neural network is applied in diversified fields for image categorizations and analysis (Khan et al. 2020; Kim et al. 2020; Sanchez Lasheras et al. 2020). The best part of CNN is that it performs effectively when applied to remote sensing images. It classifies hyperspectral images by shifting them in different angles, scaling them in various dimensions, and handling distortions/outliers/noise in them. CNN is made of neurons, and each of the neurons symbolizes a spatial region in the image.

Neurons in an ith layer are connected to a subset of neurons in the (i-1)th layer, and their links share the same weight and generate a feature map. As weight parameters are standard across a set of neurons, the number of parameters to train the CNN will drastically reduce. Using convolution operators, the feature map is further reduced, and this process is called subsampling. Subsampling in CNN can be otherwise called pooling. Some pooling techniques are available in the literature: average pooling, max pooling, min pooling, global pooling, etc. During the pooling operation, the size of the feature map is reduced by applying different filters of varying sizes. Max pooling is the operation that chooses the pixel with maximum value in the region of applying the filter over the feature map. Min pooling is the opposite of max pooling. Min and max pooling choose the dominating pixel and the light variant pixel from the feature map covered by the filter.

Average pooling is the operation that takes the average value of the pixels in the region of the feature map covered by the filter. The global pooling reduces the chosen region on the feature map to one value. Every feature map is reduced to one value in the last convolutional layer.

A typical architecture of CNN is shown in Figure 2.

Convolutional layer

Convolutional and pooling layers are applied in pairs to the hyperspectral image. Several of such pairs can be included to deepen the feature extraction. The initial convolutional layer extracts the lower level features, and the forthcoming layers extract the higher features, and so on. To extract such features, filters are applied over the original image. A filter includes kernels, and a notable point is the number of kernels and the size of the kernels used. Each filter holds a unique bias value and introduces translational invariance (Table 4).

Kernel, k, a matrix of numbers, pass over the image, i, from strides and transforms the image based on the values in the kernel matrix and calculates the weighted sum. The sum of local weight obtained is passed through an activation function like ReLU (rectified linear unit) and generates a feature map of size i × k. For instance, on applying a kernel of size (3 × 3) on the image of size (28 × 28), a feature map of size (26 × 26) is obtained. For different values in the kernel matrix, different feature maps are obtained. Whensoever a feature map is created, the image size is reduced, critical to a loss in some of the images. On applying multiple filters (say 32) over the same image, 32 feature maps are obtained. The 32 feature maps (26 × 26) are stacked one above the other and combined. The final tensor is a 3D matrix.

Pooling layer

This layer is applied to speed up the operation and reduce the tensor's size in the convolutional layer. It reduces the dimension of the feature map. It introduces slight invariance in the image, like rotation or translation, that guarantees appropriate image classification. The feature maps are divided into smaller regions, and particular operations are performed on them. A commonly used pooling operation is "max pooling" that calculates the smaller regions' maximum value in the feature map. When applying to pool over 3D tensor, the convolutional layer's output, 3D pooled feature matrix, is obtained. A 3 × 3 max pool, when applied over 32 (26 × 26) feature maps, generates 32 (13 × 13) sized images.

Fully connected layers

This layer summarizes the information of the lower layers and aggregates them. For aggregating the results, ReLU or softmax is applied in the majority of the CNN models. They flatten the results before classification. The last layer of CNN might contain many neurons, and the value produced by each neuron can take values in any range. If the objective of the developed model is a two-class classification model, then for a given input image, the model must predict it to the suitable class. The softmax activation function will process the results of the entire neurons in the output layer into binary values [0,1], thus classifying them into a specific class. However, for a given input, ReLU gives a linear output for values greater than 0 and zero output for values lesser than 0, thus classifying them into one of the two classes. The increase in filters in the CNN layer learns the intricacies of the image in a better way. Edges and corners are detected in initial layers, and the parts of the image are identified in the middle layers, and the last layer has high-end representations that recognize the entire objects, their shape, and texture.

Framework

A mathematical description of the proposed framework is explained in this section. In the given samples and the class

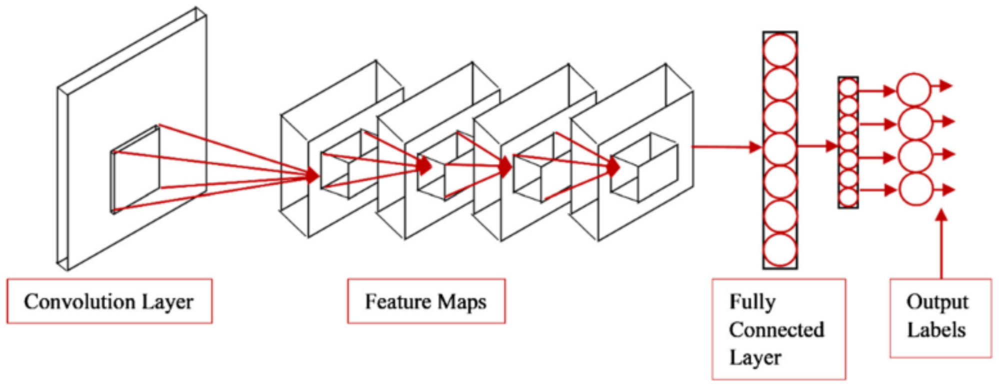

Figure 2 Architecture of CNN

Table 4 CNN filter size

Input image	Filter		The output of the convolutional layer
	Kernel size	Number of kernels	
28 × 28	(3 × 3)	32	32 × 26 × 26

labels, how the model processes the training image and classifies them is elucidated.

The dataset contains N samples that are labeled as $X = \{x\}_{i=1}^{N}$ in a space and the class labels Y={1,2…. C_{ncl}}, where the ncl represents the number of class labels. In the SAT-4 dataset, the number of classes is ncl=4, hence Y={1,2,3,4}. Let say there are n_j samples in each jth class and $\sum_{j=1}^{ncl} n_j = N$. The training set (X) includes the images, and the test set (Y) includes one hot encoded value. The TrX, $TrX = \{trx\}_{i=1}^{40000}$ is the training collection of images. The training images' corresponding class labels are represented in TrY,$TrY = \{(y_b, y_b, y_b, y_b)^1 \| (y_b, y_b, y_b, y_b)^2 \| (y_b, y_b, y_b, y_b)^3 \| (y_b, y_b, y_b, y_b)^4\}$.

TrY takes either one of the values given as $\{(y_b, y_b, y_b, y_b)^1 = (1, 0, 0, 0),\ (y_b, y_b, y_b, y_b)^2 = (0, 1, 0, 0),\ (y_b, y_b, y_b, y_b)^{3-} = (0, 0, 1, 0),\ (y_b, y_b, y_b, y_b)^4 = (0, 0, 0, 1)\}$. The above set represents that a specific image in the training set will belong to either one of the four class/class labels. This is called to be one hot encoding of class labels. In like manner, TeX includes test images, $TeX = \{tex\}_{i=1}^{10000}$ and its corresponding target class labels are given as $TeY = \{(y_b, y_b, y_b, y_b)^1 | |(y_b, y_b, y_b, y_b)^2 \| (y_b, y_b, y_b, y_b)^3 \| (y_b, y_b, y_b, y_b)^4\}$. The proposed system develops and analyzes various CNN models to predict the target class label, PrY. The predicted class label is given in Figure 3.

Feature extraction

The CNN architecture identifies the significant features of applying a distinct blend of convolutional layers (the kernel, activation function), max pooling layers, and fully connected layers (activation functions) to identify the spatial correlation pixels by exploiting the relationship between neurons in the layers as mentioned above. The classifiers that are applied in the proposed approach for classifying images are mentioned below:

SGD (stochastic gradient descent)

Keras provides the SGD optimizer that makes use of both learning rate and momentum. SGD updates the gradient by working over one training example pair (x^i, y^i) and is much faster than batch gradient update. A brief idea of gradient descent is explained in the "Preprocessing" section. Momentum adds weightage to the SGD by accelerating the gradient to move in the right direction and penalizing redundant oscillations. SGD might overshoot the local minima, yet varying the learning rate might allow SGD to reach global minima for convex and non-convex cost functions. The weight update formula in SGD, when momentum is 0, is given in Eq. (10).

$$W = W - \alpha \frac{\partial}{\partial W} cost\left(W; x^i, y^i\right) \tag{10}$$

When momentum exceeds 0.

$$V_t = \beta V_t + \alpha \frac{\partial}{\partial W} cost(W; X, Y) \tag{11}$$

$$W = W - V_t \tag{12}$$

In Eq. (11) and (12), β, beta, represents the hyperparameter that takes values from 0 to 1. In the majority of the cases, the β value is set above 0.9 to smoothen the curve. SGD speeds up the iterations, calculates the expected loss, and converges faster to more essential data.

Adagrad

Adagrad adjusts the learning rate based on the frequency of the features. The learning rate is updated with smaller steps for frequently occurring features and vice versa and so handles sparse data. That is the reason why Adagrad optimizer shows promising results in predicting barren lands accurately. At each iteration of execution, Adagrad uses different learning rates for various parameters. For reference, the SGD parameter update Eq. (13) is taken

$$w^{t+1,\ i} = w^{t,i} - \alpha \frac{\partial}{\partial w^i} cost\left(W^{t,i}\right) \tag{13}$$

The Adagrad modifies the cost value at each step t and for every parameter of w^i based on the gradient's past value, Eq. (14).

$$w^{t+1,\ i} = w^{t,i} - \frac{\alpha}{\sqrt{G^{t,ii} + \epsilon}} \frac{\partial}{\partial w^i} cost\left(W^{t,i}\right) \tag{14}$$

G^t is a d × d diagonal matrix. Each value in the diagonal element i holds a value = (sum of all gradient squares, w^i till t). Epsilon is the smoothing term. Another advantage of Adagrad is that it tunes the learning rate automatically as it sums the gradients and adds in the denominator.

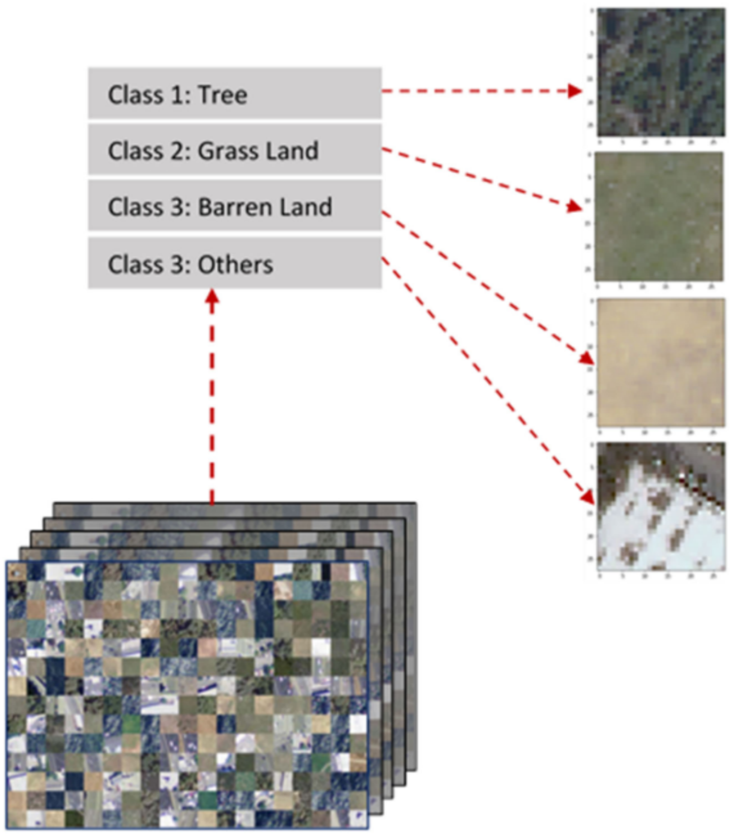

Figure 3 Process of organizing the pixel pair model and classifying them into multiple classes

RMSprop

The most significant concern of Adagrad is that the learning rate diminishes and shows no enormous variation. RMSprop is another version of Adagrad that resolves the diminishing gradient problem. It replaces the denominator by exponentially decaying the average of gradient square, Eq. (15).

$$w^{t+1,\ i} = w^{t,i} - \frac{\alpha}{\sqrt{\left[0.9\ e[g^2]_{t-1} + 0.1[g^2]_t\right] + \epsilon}} \frac{\partial}{\partial w^i} cost\left(W^{t,i}\right) \tag{15}$$

A reasonable learning rate will hold a value of 0.001.

Adam

Adaptive Moment Estimation (Adam) keeps a record of exponentially decaying average of past gradients for momentum. It is a rolling ball with friction. Adam measures both the first and second momentum. The update rule is measured as Eq. (16).

$$w^{t+1,\ i} = w^{t,i} - \frac{\alpha}{\sqrt{\frac{var^t}{1-\beta_2^t} + \epsilon}} \frac{mean^t}{1-\beta_1^t} \tag{16}$$

$$mean^t = \beta_1 mean_{t-1} + (1-\beta_1)\frac{\partial}{\partial w^i} cost\left(W^{t,i}\right) \tag{17}$$

$$var^t = \beta_2 var_{t-1} + (1-\beta_2)\left[\frac{\partial}{\partial w^i} cost\left(W^{t,i}\right)\right]^2 \tag{18}$$

The first momentum (mean) and second momentum (variance) are calculated, decaying averages of past gradients and past squared gradients. Since the mean and variance are considered, the optimizer works well in handling the learning curve. Among the number of optimizers applied for training the deep CNN networks, the proposed model applies the four optimizers mentioned earlier in light of their excellence with satellite imagery. The experimental models of CNN architecture run over these optimizers.

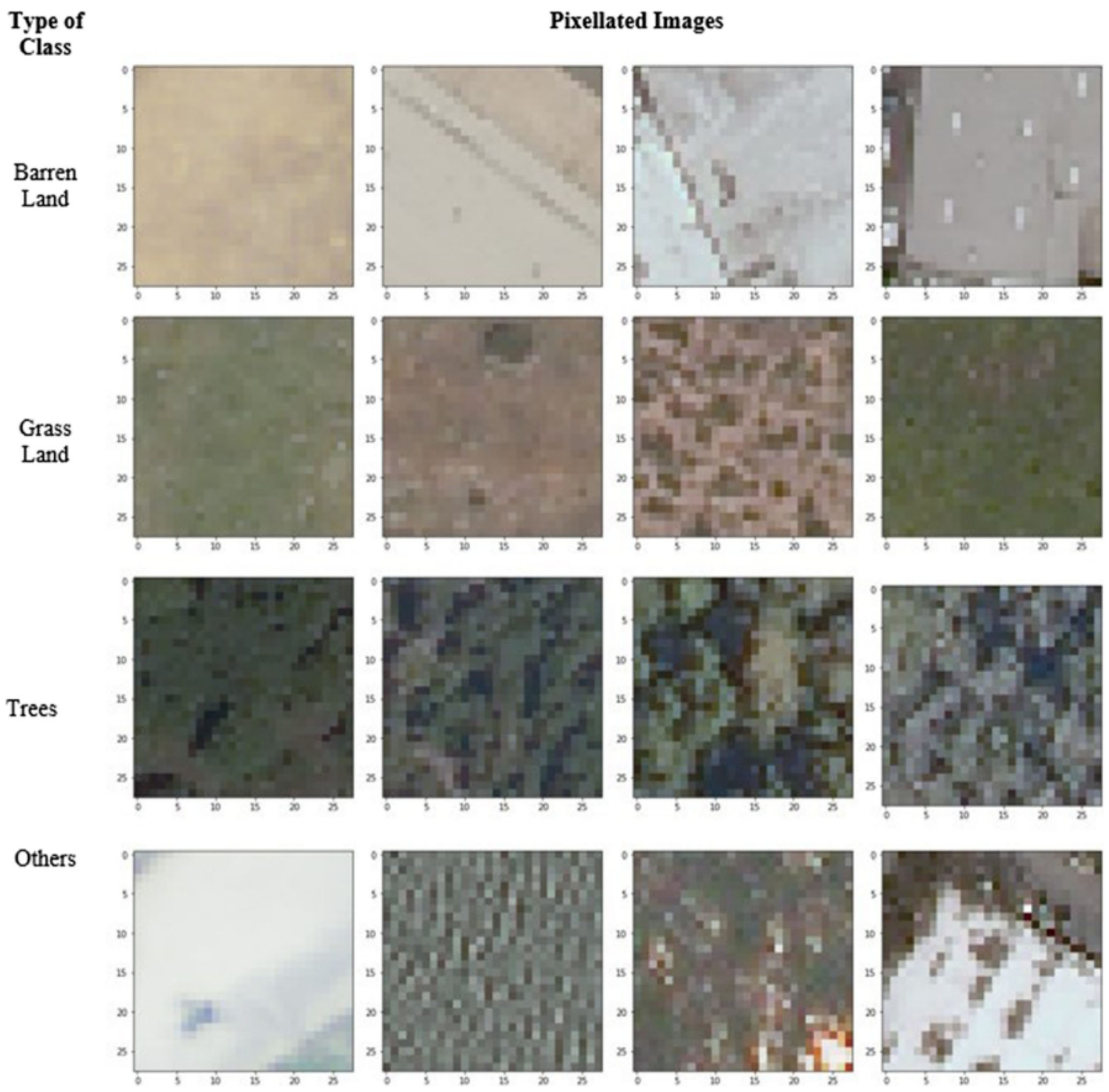

Figure 4 Training images

Experimental models

Around 9 CNN models were implemented with varying architecture, and each of them is analyzed with different batch sizes and optimizers. Around 228 images of barren land, 215 images of trees, 170 images of grassland, and 387 other images were taken for prediction models. We group all the 9 models into two categories based on the number of layers they use: category 1 includes models 1, 2, 4, 6, and 8 and category 2 includes models 3, 5, 7, and 9. The models in category 1 are designed with one-layered architecture, and models in category 2 are designed with 5-layered architecture. The model that accelerates the training of the samples and speeds up the convergence is identified. The application of the loss function is categorical cross-entropy. A sample of four images from each of the training classes is put in Figure 4.

The analysis over the pattern of pixels in the sample images is mentioned below:

Table 5 Training loss, validation loss, training accuracy, validation accuracy of nine CNN models

Model	Layer/no. of neurons					Batch size	Optimizer	Training loss	Validation loss	Training accuracy	Validation accuracy	Correctly mapped	Incorrectly mapped
	L1/8N	L2/12N	L3/16N	L4/12N	L5/4N								
1	X	X	X	X	Y	32	Adam	0.65742	0.65766	0.72395	0.7297	667	333
2	X	X	X	X	Y	64	Adam	0.65024	0.62649	0.72822	0.7382	769	231
3	Y	Y	Y	Y	Y	64	Adam	1.3506	1.3404	0.35594	0.387	690	310
4	X	X	X	X	Y	64	Adagrad	0.82971	0.81032	0.70851	0.7175	756	244
5	Y	Y	Y	Y	Y	64	Adagrad	0.88348	0.85618	0.63261	0.6481	651	349
6	X	X	X	X	Y	64	RMSprop	0.78857	0.77826	0.6749	0.679	694	306
7	Y	Y	Y	Y	Y	64	RMSprop	0.77706	0.79946	0.66143	0.6412	387	613
8	X	X	X	X	Y	64	SGD	0.68099	0.65312	0.7206	0.7333	725	275
9	Y	Y	Y	Y	Y	64	SGD	0.60644	0.57469	0.75671	0.7661	683	317

i. The pixels in barren land have higher values of RGB. The pixels will have the same RGB value because barren land in any region follows the same hue. The possibility of finding pixels with lower values of RGB is significantly less in the barren land. We may not find clouds of pixels with varying RGB.
ii. The pixel pattern in grassland might have lower values of RGB than barren land. We might find clouds of dusky pixels that represent the density of the grassland. The depth of vegetation covering the grassland matters the most. If it is too less, then grassland might resemble barren land representing high values of RGB.
iii. Dense and deep vegetation is called tree land. The color of the pixels in these images is intense and carries lower values of RGB. The hue pattern might be dark enough to represent the thickness of the forest. A blend of the dark region and dusky region can be seen.
iv. Images that do not bear a typical pattern fall into the last category.

Training the models

The training samples are run with nine models classified into two categories that accelerate the weight update cycle with more minor backpropagation errors to speed up the whole model's convergence. The optimization was run to reduce the loss function (J) (i.e., categorical cross-entropy) of CNN expressed in Eq. (19).

$$L(IF, W, b, PCL) = -\frac{1}{N}\left[\sum_{i=1}^{N}\sum_{j=1}^{K}\{y_i = target\}\, y_i\right] \tag{19}$$

L	is the loss function.
IF	represents the normalized features.
W	represents the weight vector, and b represents the bias.
y_i	

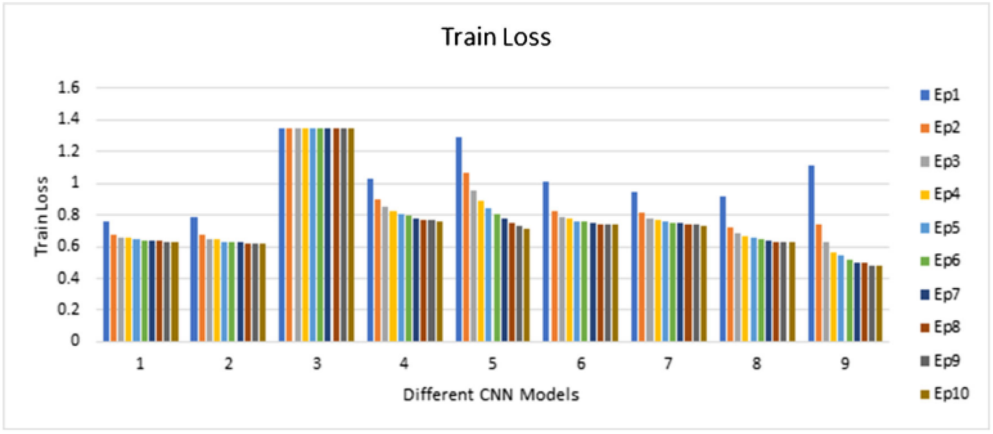

Figure 5 Training loss of CNN models for 10 epochs

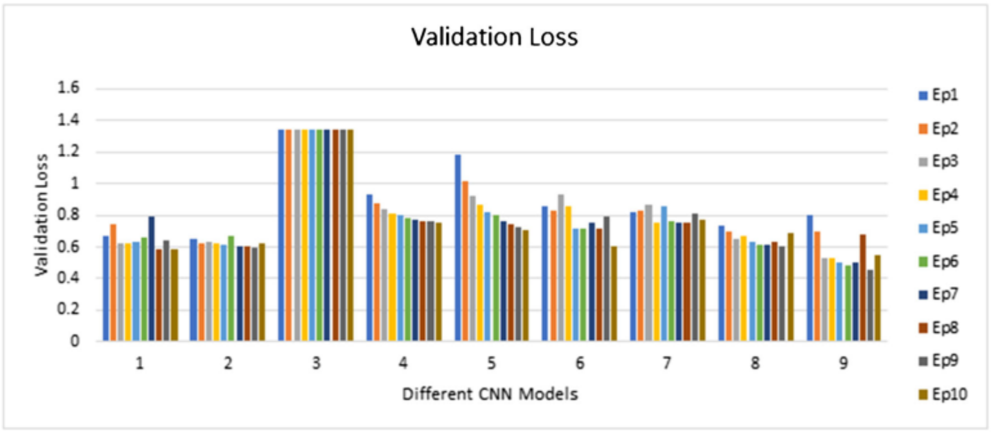

Figure 6 Validation loss of CNN models for 10 epochs

represents the probability of sample I falling into a target class t, and that is derived by softmax.

$\{y_i = target\}$ is the prediction vector mentioned as:

$$\left\{y_i = \left(barren\ land, grass\ land, trees, others\right\}\right.$$

K represents a value of 4, indicating each class.
N is the number of training examples.

Category 1

Model 1 This model is designed with one layer having four nodes using the softmax activation function. The softmax function takes a vector of numbers as input and normalizes them to a probability function. The probability values generated are proportional to the exponential of the input vector. Specific input vectors can take a negative sign or positive sign, and their sum of probabilities might not equal 1. However, on applying softmax Eq. (20), the input vectors will be reduced to a probability factor between {0,1}.

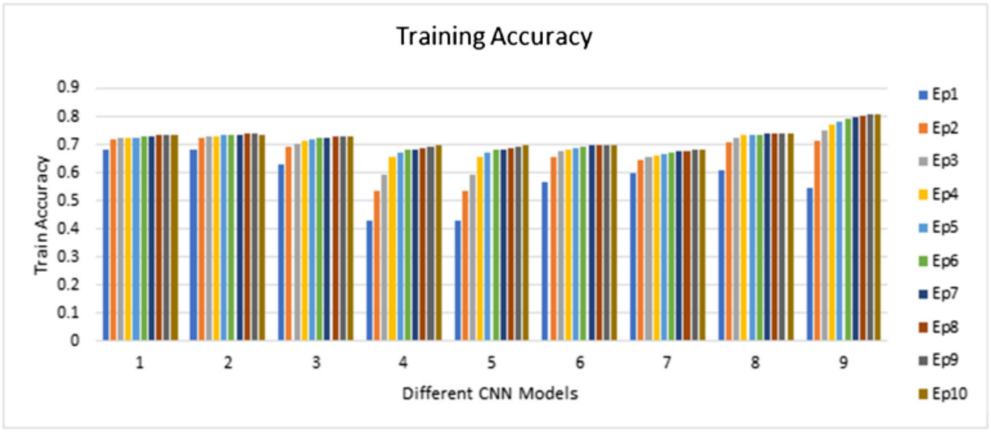

Figure 7 Training accuracy of CNN models for 10 epochs

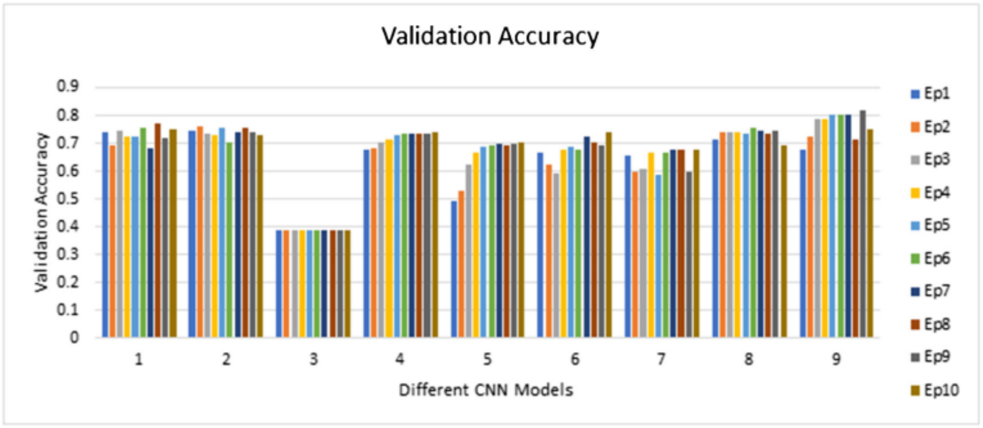

Figure 8 Validation accuracy of CNN models for 10 epochs

$$y_i, softmax(PCL_i) = \frac{e^{PCL_i}}{\sum_{j=1}^{K} e^{PCL_j}} \tag{20}$$

PCL represents the input vector. PCL_i is the input vector's value for a class label, i., i.e., the input image's value being classified into any of the four class labels. It may take a positive or negative sign. Raising this value to the power of the exponent always generates a positive value. There would be four such values generated and each of them corresponds to one class label: $\{PCL_1\}$, $\{PCL_2\}$, $\{PCL_3\}$, and $\{PCL_4\}$. These values are further normalized by dividing all the PCL_i, as mentioned in the denominator. It defines the sum of softmax of all the PCL_i to be 1, and each of its values is a probability factor.

In the context of the proposed system, the input vector contains RGBI values of each image. A tuple of RGBI values is passed as input to Model 1. The model gives outputs of four values: $\{\{PCL_1\}, \{PCL_2\}, \{PCL_3\}$, and $\{PCL_4\}\}$. On applying the softmax function, the probability values, each for each class label (barren land, trees, grassland, others) are derived. The sum of all the probability values adds up to one. The tuple will be assigned to the class with the highest probability value. Model 1 uses Adam optimizer and applies a batch size of 32. Its outputs are 72.395 for training accuracy and 72.97 for validation accuracy.

Model 2 Model 2 is designed similar to Model 1 but with a slight variation in the batch size. Model 1's batch size is 32, and for model 2, it is 64. Change in batch size does not cause a high difference in the training and validation accuracy between the models, but model 2 shows a slight increase in training and validation accuracy by 0.00427 and 0.00998. It is seen that batch size 64 performs better than 32.

Model 4 Model 4 is designed with one layer encompassing four nodes, and the Adagrad optimizer is applied with an input batch size of 64. The training accuracy of 0.70851 and the validation accuracy of 0.7175 are obtained.

Model 6 All the models in category 1 use one layer. In the same way, model 6 applies one layer with four nodes but uses an RMSprop classifier. Input batch size is 64. This model shows minor training and validation accuracy among all the 5 models in category 1; this model shows lesser training and validation accuracy (0.6749 and 0.679) since it is the RMSprop classifier's characteristic. RMSprop will try converging to local minima or take a shorter path when a new data point or a new image says an outlier is encountered. It is shaky to outliers.

Model 8 Model 8 applies an SGD classifier and uses one layer of four nodes. The training accuracy and validation accuracy are 0.7206 and 0.7333, respectively. However, this model shows promising results similar to model 2, and it is because of the stochastic gradient descent optimizer's sustenance capacity.

On varying the number of layers and number of nodes in CNN architecture, it was found that the SAT-4 dataset shows no significant difference with two layers or three layers. For a four-layered architecture, the accuracy was slightly increased, and a couple of models were built with four layers and are grouped in category 2.

Category 2

Model 3 Model 3 is formed in four layers with 8, 12, 16, and 12 nodes in each layer, respectively. Adam optimizer applies a batch size of 12 and shows trivial performance with training error as 0.35594 and validation split as 0.387. Adam is essential in its way because it shows superior performance by taking

Table 6 Confusion matrix enclosing the true positive (TP), false positive (FP), true negative (TN), and false negative (FN) of all the models for 1000 samples

MODEL 1		Actual Class			
		Barren Land	Trees	Grassland	Other
Predicated Class	Barren Land	188	0	9	3
	Trees	0	47	2	10
	Grassland	27	82	129	71
	Other	13	86	30	303

MODEL 2		Actual Class			
		Barren Land	Trees	Grassland	Other
Predicated Class	Barren Land	215	0	20	19
	Trees	1	168	19	45
	Grassland	1	18	89	26
	Other	11	29	42	297

MODEL 3		Actual Class			
		Barren Land	Trees	Grassland	Other
Predicated Class	Barren Land	222	0	19	46
	Trees	0	105	23	16
	Grassland	5	1	81	43
	Other	1	109	47	282

MODEL 4		Actual Class			
		Barren Land	Trees	Grassland	Other
Predicated Class	Barren Land	215	0	16	15
	Trees	0	134	17	27
	Grassland	1	30	95	33
	Other	12	51	42	312

MODEL 5		Actual Class			
		Barren Land	Trees	Grassland	Other
Predicated Class	Barren Land	212	0	30	77
	Trees	0	134	43	23
	Grassland	0	6	20	2
	Other	16	75	77	285

MODEL 6		Actual Class			
		Barren Land	Trees	Grassland	Other
Predicated Class	Barren Land	206	0	8	3
	Trees	0	42	2	10
	Grassland	3	48	100	28
	Other	19	125	60	346

MODEL 7		Actual Class			
		Barren Land	Trees	Grassland	Other
Predicated Class	Barren Land	0	0	0	0
	Trees	0	0	0	0
	Grassland	0	0	0	0
	Other	228	215	170	387

MODEL 8		Actual Class			
		Barren Land	Trees	Grassland	Other
Predicated Class	Barren Land	212	0	16	10
	Trees	0	92	13	15
	Grassland	1	14	64	5
	Other	15	109	77	357

MODEL 9		Actual Class			
		Barren Land	Trees	Grassland	Other
Predicated Class	Barren Land	214	0	6	8
	Trees	0	214	62	202
	Grassland	5	0	85	7
	Other	9	1	17	170

Table 7 Prediction values of proposed model's vs. ground truth: tree image

	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8	Model 9
Barren land	0.004	0.012	0.259	0.066	0.017	0.003	0.012	0.009	0.001
Trees	0.132	0.126	0.207	0.332	0.241	0.109	0.096	0.093	0.724
Grassland	0.764	0.77	0.178	0.364	0.339	0.776	0.361	0.838	0.186
Other	0.1	0.093	0.356	0.238	0.403	0.113	0.531	0.06	0.089

minimal training time. Yet, for some datasets and architecture, it does not converge to an optimal solution.

Model 5 and Model 7 This model 5 is designed with the same model 3 but applies Adagrad optimizer and gives a training error of 0.63261 and validation error of 0.6481. As the iterations of gradient calculation keep growing, the learning rate takes a negligible value, thereby reducing the model's performance. This is one of the significant drawbacks of the Adagrad optimizer. Model 7 applies an RMSprop optimizer and gives a training error of 0.66143 and a validation error of 0.6412. Both model 5 and model 7 provide almost similar training accuracy.

Model 9 It applies an SGD optimizer and gives a training error of 0.75671 and a validation error of 0.7661. This model provides an accuracy that is the highest of all the models. The reason behind the high accuracy of SGD is that the SGD optimizer works with a small subset that randomly picks a sample and narrows down the gradient.

The architecture of the models, as mentioned above, provides good performance for predicting specific class labels. However, with a particular architectural style, the CNN performance might be high and would saturate. Two such models are model 2 and model 9, with a validation accuracy of 0.7382 and 0.7661. The measured batch size, training loss, validation loss, training accuracy, and validation accuracy of the nine CNN models are tabulated in Table 5.

Figure 5, 6, 7, and 8 show the training loss, validation loss, training accuracy, and validation accuracy of all the CNN models with normalization for 10 epochs on both training and validation datasets. In terms of training loss, all the models indicate a slight decrease in error with increasing epochs. This shows that the model has learned to extract useful features from the pixels in the image and has self-understood the class labels that support the classification of the images. Though the validation dataset shows meager fluctuations while calculating the loss, the last epoch's loss value is minimal in most models. Model 9 shows a minimal validation error of 0.574. The proposed models do not seem to overfit, and hence dropout is not required. Dropout is applied to release or drop some neurons in the CNN layers, and dropout prevents the model from getting overfitted.

The performance of the models in classifying the images depends on the hyperparameters and architecture. A sensitivity analysis serves as an essential measure to find the specific set of parameters to learn the model prediction. Table 5 shows the impact of hyperparameters on the loss and accuracy of CNN. The hyperparameters applied in the proposed work include varying the number of CNN hidden layers, varying the neurons in the layer, applying varying optimizers, varying the number of epochs, and varying the batch size.

The sensitivity analysis in CNN presumes that the larger the number of layers and filters, the higher the performance. Variation is seen in 32 and 64 batch size, and 64 batch size shows the higher optimal solution. With 64 batch size configurations, the CNN model achieves higher accuracy. CNN's performance with different optimizers is investigated, and the results show that SGD and Adam show good performance. The poor performance is shown by RMSprop and Adagrad optimizers. The spatial quality of the features from images is chosen only by the specific combination of filters, batch size, and epoch.

Table 8 Prediction values of proposed model's vs. ground truth: grassland image

	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8	Model 9
Barren land	0.037	0.024	0.098	0.169	0.171	0.022	0.2	0.068	0.05
Trees	0.033	0.014	0.029	0.129	0.095	0.066	0.001	0.111	0.002
Grassland	0.732	0.526	0.58	0.423	0.241	0.766	0.228	0.364	0.781
Other	0.198	0.436	0.293	0.278	0.493	0.146	0.572	0.457	0.167

Table 9 Prediction values of proposed model's vs. ground truth: barren land image

	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8	Model 9
Barren land	1	1	1	0.974	0.945	1	0.454	0.999	1
Trees	0	0	0	0	0	0	0	0	0
Grassland	0	0	0	0.003	0.001	0	0	0.001	0
Other	0	0	0	0.023	0.054	0	0.546	0	0

In parallel, the model accuracy for the training and validation dataset for 10 epochs is determined. The results imply that the accuracy of all the models shows a linear increase with epochs. The maximum accuracy obtained while validating model 9 is 0.7661. The validation accuracy of the model shows a slight off and on change, and the consolidated accuracy of all the CNN models shows that the model has learned well to classify images. The CNN models are trained with no dropout of nodes, and the accuracy of the classification map is seen from the results obtained.

Analysis of confusion matrix of the proposed CNN models

The entire set of images is passed through each of the models, and the classification characteristics of the model are recorded in the form of a matrix. The matrix tabulates the predicted class labels to the ground truth of images. Each of the models' classification is tabulated as a contingency table (4 × 4 matrix). The confusion matrix (Table 6) displays the true positive (TP), false positive (FP), true negative (TN), and false negative (FN) values of all the models with 1000 input samples. On analyzing the matrix, the results indicate the models that rightly classify the vegetation. Model 1 sounds good in predicting the grassland. Model 9 shows a high true positive count for trees and barren land. Several samples were misclassified by model 7, and most of the misclassifications were amid trees and grasslands. Either way, trees were assumed as grasslands or vice versa.

The confusion matrix represents the stacked analysis of images in the dataset by nine models. Every model runs through the dataset and constructs the matrix as given in Table 6. We get to know the suitable model for a specific class of image and the prediction characteristics of each of the developed models. In general, such a stacked analysis is built for image classification algorithms to know the strength of the classification.

Prediction of proposed system vs. ground truth

We chose four random images whose ground truth entitles it as a tree, grassland, barren land, and others to evaluate the model's prediction behavior. The percentage of prediction by the proposed models for sample images is reviewed in Table 7, 8, 9 and 10.

All the 9 models evaluate this image, and the result of the prediction is put together in Figure 9.

The treemap image of the predicted results is plotted here. A puzzling case is to rightly predict a tree as a tree and grassland as grassland. Model 9 identifies a tree image as a tree. All the proposed models rightly predict the ground truth image with few showing slight deviations. The computing performance of the proposed nine models is efficient for SAT-4 dataset.

The proposed nine models are trained in a way specific to the satellite images in the training set. Following any other dataset, the models can be slightly modified in their layers and can be applied. As such, the proposed model can be applied to other satellite-based datasets for classifying the images into four broad classes. However, other platforms can also be proposed using different packages, and the performance can be compared. But the proposed model focus on building the suitable CNN architecture for the specific class label classification.

Conclusion

This paper builds CNN classification models to determine the spectral-spatial features from SAT-4 images and group them under 4 different classes. In the paper's initial sections, the proposed CNN framework, the various optimizers, and the corresponding gradient optimization methods applied are

Table 10 Prediction values of proposed model's vs. ground truth: others

	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8	Model 9
Barren land	0.000	0.002	0.264	0.043	0.001	0.000	0.000	0.003	0.000
Trees	0.090	0.211	0.202	0.161	0.252	0.018	0.430	0.430	0.157
Grassland	0.026	0.070	0.176	0.143	0.116	0.007	0.126	0.115	0.021
Other	0.884	0.717	0.358	0.653	0.631	0.974	0.444	0.444	0.822

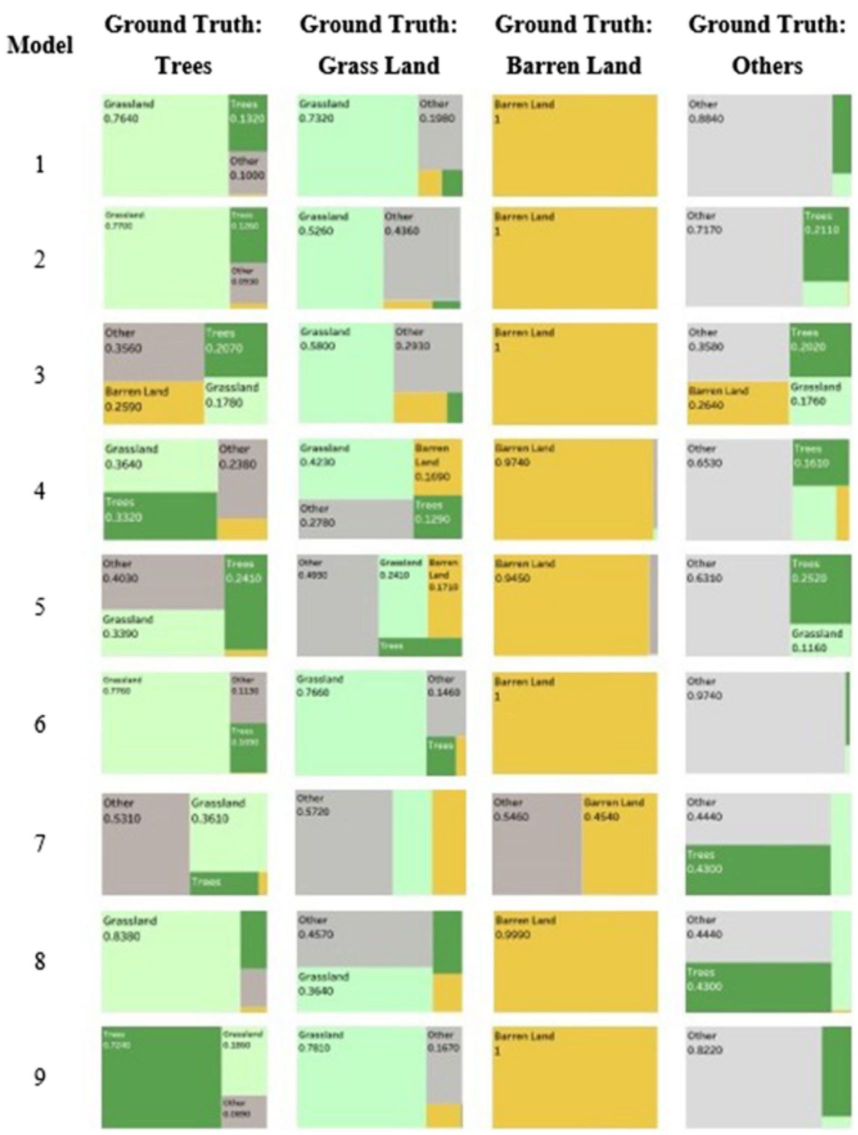

Figure 9 Treemap showing the prediction value of proposed system vs. ground truth

briefly described. Around 9 CNN models are constructed with varying degrees of regularization, varying number of layers, tuned hyperparameters, altered batch size, etc. The proposed model signifies the different ways of classifying the images into four classes: trees, grassland, barrel land, and others using all the CNN models. The sensitivity analysis of the proposed CNN models was studied, and the results show that they are robust and demonstrate sound reasoning in classifying the images. For these 9 models, the categorical loss entropy and accuracy are determined. The CNN classifier is trained end-to-end in various dimensions by passing around 40,000 images. The training and validation accuracy of the models in each of 10 epochs are measured. A true positive, true negative, false positive, and false negative are analyzed for each model. Similarly, a test image is passed to all the models, and then the prediction percentage of all of them is calculated. An extended version of model 9 can be built to classify the images with minimal error and high accuracy and can be considered to scale for large remote sensing datasets.

Declarations

Conflict of interest The authors declare that they have no competing interests.

References

Albert L, Rottensteiner F, Heipke C (2017) A higher-order conditional random field model for simultaneous classification of land cover and land use. *ISPRS J Photogramm Remote Sens 130*:63–80 *11*(1), 3

Andrejchenko V, Liao W, Philips W, Scheunders P (2019) Decision fusion framework for hyperspectral image classification based on Markov and conditional random fields. *Remote Sens 11*(6):624

Basu S et al (2016) A theoretical analysis of Deep Neural Networks for texture classification. 2016 International Joint Conference on Neural Networks (IJCNN), pp 992–999. https://doi.org/10.1109/IJCNN.2016.7727306

Basu S, Ganguly S, Mukhopadhyay S, DiBiano R, Karki M, & Nemani R (2015). Deepsat: a learning framework for satellite imagery. In *Proceedings of the 23rd SIGSPATIAL international conference on advances in geographic information systems* (pp. 1-10).

Basu S, Karki M, Ganguly S et al (2017) Learning sparse feature representations using probabilistic quadtrees and deep belief nets. Neural Process Lett 45:855–867. https://doi.org/10.1007/s11063-016-9556-4

Bradter U, Thom TJ, Altringham JD, Kunin WE, Benton TG (2011) Prediction of National Vegetation Classification communities in the British uplands using environmental data at multiple spatial scales, aerial images and the classifier random forest. *J Appl Ecol 48*(4):1057–1065

Chen Y, Lin Z, Zhao X, Wang G, Gu Y (2014) Deep learning-based classification of hyperspectral data. *IEEE J Select Topics Appli Earth Observ Remote Sens 7*(6):2094–2107

Chen Y, Zhao X, Jia X (2015) Spectral–spatial classification of hyperspectral data based on deep belief network. *IEEE J Select Topics Appl Earth Observ Remote Sens 8*(6):2381–2392

Chen H, Wu C, Du B, Zhang L, Wang L (2019) Change detection in multisource VHR images via deep Siamese convolutional multiple-layers recurrent neural network. *IEEE Trans Geosci Remote Sens 58*(4):2848–2864

Cheng G, Ma C, Zhou P, Yao X, & Han J (2016). Scene classification of high-resolution remote sensing images using convolutional neural networks. In *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)* (pp. 767-770). IEEE..

Fukushima K, Miyake S, Ito T (1983) Neocognitron: a neural network model for a mechanism of visual pattern recognition. *IEEE Trans Syst Man Cybern* 5:826–834

Guan H, Li J, Chapman M, Deng F, Ji Z, Yang X (2013) Integration of ortho imagery and lidar data for object-based urban thematic mapping using random forests. *Int J Remote Sens 34*(14):5166–5186

He K, Zhang X, Ren S, Sun J (2015) Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Trans Pattern Anal Mach Intell 37*(9):1904–1916

Hoberg T, Rottensteiner F, Heipke C (2012) Context models for CRF-based classification of multitemporal remote sensing data. *ISPRS Ann Photogramm Remote Sens Spatial Inform Sci I-7 (2012), Nr. 1* 1(1):129–134

Hu W, Huang Y, Wei L, Zhang F, Li H (2015) Deep convolutional neural networks for hyperspectral image classification. *J Sensors 2015:1–12*

Juel A, Groom GB, Svenning JC, Ejrnaes R (2015) Spatial application of random forest models for fine-scale coastal vegetation classification using object-based analysis of aerial orthophoto and DEM data. *Int J Appl Earth Obs Geoinf 42*:106–114

Khan SA, Kazmi KR, Yambangwai D, Cholamjiak W (2020) A hybrid projective method for solving system of equilibrium problems with demicontractive mappings applicable in image restoration problems. *Math Methods Appl Sci 43*(6):3413–3431

Kim G, Han M, Shim H, Baek J (2020) A convolutional neural network-based model observer for breast CT images. *Med Phys 47*(4):1619–1632

LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. *Proc IEEE 86*(11):2278–2324

Liu Q, Hang R, Song H, Li Z (2017) Learning multiscale deep features for high-resolution satellite image scene classification. *IEEE Trans Geosci Remote Sens 56*(1):117–126

Ma X, Wang H, Geng J (2016) Spectral–spatial classification of hyperspectral image based on deep auto-encoder. *IEEE J Select Topics Appl Earth Observ Remote Sens 9*(9):4073–4085

Mishra NB, Crews KA (2014) Mapping vegetation morphology types in a dry savanna ecosystem: Integrating hierarchical object-based image analysis with Random Forest. *Int J Remote Sens 35*(3):1175–1198

Ouyang W, Wang X, Zeng X, Qiu S, Luo P, Tian Y, ... & Tang, X. (2015). DeepID-net: deformable deep convolutional neural networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2403-2412).

Ouyang W, Zhou H, Li H, Li Q, Yan J, Wang X (2017) Jointly learning deep features, deformable parts, occlusion and classification for pedestrian detection. *IEEE Trans Pattern Anal Mach Intell 40*(8): 1874–1887

Pal M (2005) Random forest classifier for remote sensing classification. *Int J Remote Sens 26*(1):217–222

Parikh D, & Batra D (2008). *CRFs for image classification*. Technical report, Carnegie Mellon University.

Puissant A, Rougier S, Stumpf A (2014) Object-oriented mapping of urban trees using random forest classifiers. *Int J Appl Earth Obs Geoinf 26*:235–245

Rodriguez-Galiano VF, Ghimire B, Rogan J, Chica-Olmo M, Rigol-Sanchez JP (2012) An assessment of the effectiveness of a random forest classifier for land-cover classification. *ISPRS J Photogramm Remote Sens 67*:93–104

Romero A, Gatta C, Camps-Valls G (2015) Unsupervised deep feature extraction for remote sensing image classification. *IEEE Trans Geosci Remote Sens 54*(3):1349–1362

Sameen MI, Pradhan B, Aziz OS (2018) Classification of very high-resolution aerial photos using spectral-spatial convolutional neural networks. *J Sensors 2018*:1–12

Sanchez Lasheras F, Ordóñez C, Roca-Pardiñas J, de Cos Juez FJ (2020) Real-time tomographic reconstructor based on convolutional neural networks for solar observation. *Math Methods Appl Sci 43*(14): 8032–8041

Scott GJ, England MR, Starms WA, Marcum RA, Davis CH (2017) Training deep convolutional neural networks for land–cover classification of high-resolution imagery. *IEEE Geosci Remote Sens Lett 14*(4):549–553

Shakya S, Kumar S, Goswami M (2020) Deep learning algorithm for satellite imaging-based cyclone detection. *IEEE J Select Topics Appl Earth Observ Remote Sens 13*:827–839

Sherrah J (2016) Fully convolutional networks for dense semantic labelling of high-resolution aerial imagery. arXiv. Available online: https://arxiv.org/abs/1606.02585. Accessed 22 Aug 2018

Sun Y, Wang X, & Tang X (2014). Deep learning face representation by joint identification-verification. arXiv preprint arXiv:1406.4773.

Sun Y, Tian Y, Xu Y (2020, March) Conditional random fields based on weighted feature difference potential for remote sensing image classification. In: In *Future of Information and Communication Conference*. Springer, Cham, pp 590–603

Yao W, Poleswki P, Krzystek P (2016) Classification of urban aerial data based on pixel labelling with deep convolutional neural networks and logistic regression. *Int Arch Photogramm Remote Sens Spat Inf Sci 41*:B7

Multilabel land cover aerial image classification using convolutional neural networks

ORIGINALITY REPORT

4%	3%	3%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to SAMRAT ASHOK TECHNOLOGICAL INSTITUTE VIDISHA M.P Student Paper	1%
2	centaur.reading.ac.uk Internet Source	1%
3	wopwa.tele-lunch.pl Internet Source	1%
4	www.hindawi.com Internet Source	1%

Exclude quotes	Off	Exclude matches	< 1%
Exclude bibliography	On		