Traitement du Signal
Vol. 41, No. 5, October, 2024, pp. 2529-2538
Journal homepage: http://iieta.org/journals/ts

Classification of Satellite Images Using a Deep Learning-Inspired Hybrid Novel Approach

Bihari Nandan Pandey*, Mahima Shanker Pandey

Department of Computer Science & Engineering, Sunrise University, Alwar 301028, India

Corresponding Author Email: bnpanday@gmail.com

Copyright: ©2024 The authors. This article is published by IIETA and is licensed under the CC BY 4.0 license (http://creativecommons.org/licenses/by/4.0/).

https://doi.org/10.18280/ts.410526

Received: 2 October 2023
Revised: 5 March 2024
Accepted: 20 August 2024
Available online: 31 October 2024

Keywords:
satellite images, hybrid techniques, artificial intelligence, machine learning, CNN, SVM

ABSTRACT

Satellite imagery is crucial for disaster assistance, law enforcement, and environmental monitoring. Some users need to identify facilities and items in photographs manually. Automation becomes essential when there are large areas to search and few available analysts. However, the problem can only be fixed by increasing the precision of existing object identification and categorization methods. The "deep learning" subfield of machine learning has demonstrated promising results for automating specific tasks. Using convolutional neural networks, it was able to understand images successfully. In this work, we use high-resolution, multi-spectral satellite photos to solve the problem of identifying objects and infrastructure. In this paper, we describe a deep-learning system for labeling objects. In this research, we make use of the Satellite Image Classification Dataset-RSI-CB256. This dataset uses Google Maps images and sensors to create four distinct categories. In this study, a hybrid model is proposed, which achieves an accuracy of 98.96%.

1. INTRODUCTION

In machine learning, deep learning refers to models that use numerous processing layers to create progressively more abstract input representations. Combining massive neural network models, convolutional neural networks (CNNs), with robust graphics processing units has shown astounding success in object identification and categorization (GPUs). The annual ImageNet Large Scale Visual Recognition Competition [1] for object detection and classification in pictures has been dominated by CNN-based algorithms since 2012. As a result of this breakthrough, several big IT firms have already implemented CNN-based products and services [2], including industry heavyweights like Google, Microsoft, and Facebook.

Layers of processing code make up a convolutional neural network. The picture characteristics are detected via convolution filters, one for each layer. Feature detectors in the first layers resemble Gabor-like and color-blob filters, while those in the latter layers take the shape of convolutional neural networks. In contrast to earlier techniques such as SIFT [3] and HOG [4], the algorithm designer is not required to develop feature detectors when using CNNs. Over time, the network trains itself to recognize specific traits and improves its ability.

Successful CNNs from the beginning had less than ten layers and were intended for tasks like reading handwritten postal codes. There were five levels in LeNet [5], whereas AlexNet had eight [6]. Since then, complexity has gradually increased. VGG emerged in 2015 with 16 layers [7]; in 2016, Google released Inception with 22 layers [8]. Newer iterations of Inception, such as ResNet (152 levels) and DenseNet (161 layers), add even more layers.

CNNs need tiny, fixed-size pictures to maintain a tolerable processing time. In contrast to Inception [8, 9], ResNet [10] and DenseNet [11], can handle photographs as large as 299×299. Advanced GPUs offer the processing power needed for such massive CNNs. Further progress in deep learning has been fueled in part by open-source deep learning software frameworks like TensorFlow [12] and Keras [13], as well as powerful GPUs.

Deep learning often involves cropping and warping pictures to fit [14]. These processes preserve important visual details for typical images. On the other hand, objects and facilities in satellite photos may be considerably bigger than they seem in regular photographs. Places like airports and dockyards may span tens of thousands of pixels. More details are needed when these vast photos are downsized to 224×224 or 299×299 pixels. Aeroplanes on a runway or container cranes at a shipyard are two examples of such distinctive elements. Even if you tried to crop the image down to size to preserve detail, you'd lose too much of the picture.

2. LITERATURE REVIEW

Several picture datasets containing annotations, as well as related detection and classification efforts, have recently emerged. Land cover categorization and structure recognition have dominated deep learning's applications to remotely sensed images. For instance, the UC Merced Land Use Dataset has 2100 photographs of the United States taken from the air. Cartography of the Earth [15, 16]. The ground sample distance per pixel in these 256×256 photos is 0.3 metres. Among the 21 categories are storage tanks, tennis courts, and more typical land uses like agriculture, roads, and water. One study found a 98.5 percent accuracy rate in classifying UC Merced photos into land cover categories using the VGG, ResNet, and Inception CNNs [17-19]. Nevertheless, this dataset has severe limitations in terms of scope, classification depth, and geographic coverage.

High-resolution Digital Globe satellite photos of five cities and building footprints make up the SpaceNet dataset [20]. Convolutional neural networks (CNNs) have been taught to extract building footprints from photos [21]. In terms of training a classifier, this dataset has several severe limitations.

Further remote sensing data sets are included in the previous study [22]. They need the global corpus of hundreds of thousands of photos to train a robust image classification algorithm.

Satellite image classification organises pictures by an object or semantic meaning into three primary categories: techniques based on standard features, methods based on high-scene features, and hybrid approaches [23]. Mid-features-based methods work well for complex images [24]. Plans with many qualities best handle complex visuals. CNN is a popular deep-learning image-processing method [25].

"Deep Belief Network for classification" utilizing Convolutional Neural Networks achieves 97.946 SAT4 and 93.916 SAT6 accuracy [26]. Ju et al. [27] investigated image classification and identification ensemble techniques using deep convolutional neural networks. Saikat Basu, Sangram Ganguly, and others developed "DeepSat," a satellite image learning system. The super learner, majority voting, and Bayes Optimal Classifier are examples. Albert et al. [28] use deep CNN-based computer vision and large-scale satellite imagery to examine urban land use. A deep neural network does this with data. To find ground truth land, they carefully use open-source survey class designations. The Urban Atlas land categorization dataset comprises 300 European cities and 20 land use types. They also show that deep representations using satellite pictures of urban landscapes can compare cities' communities. Metropolitan satellite photographs proved this. Robinson et al. [29] created high-resolution population estimates using satellite data and a deep-learning convolutional neural network model. CNN algorithm trained on one year of composite Landsat pictures predicts the US population on a 0.01 by 0.01 grid. CNN model validation used quantitative and qualitative methodologies. The quantitative validation compared the proposed model's grid cell estimates to many US Census county-level population estimations. Qualitative validation directly evaluated model predictions for satellite image inputs. The model illustrates how machine learning algorithms can tackle social issues using unstructured and remotely sensed data. The literature review is described in Table 1.

Table 1. Summarized results of literature review

Reference	Model	Limitations	Results
[30]	Adaboost, XGBoost, GBDT, LR, DT, RF, SVM, NB, LR	The dataset is restricted	Recall=0.9699 and F1-score =0.9582, algorithm shows the highest performance.
[31]	Neural Networks, NB, k-NN, DT, and SVM	Control and sample coming close to discovery cannot be told apart	Both kNN and SVM algorithms worked 95.56% of the time.
[32]	Boost XG	XGBoost cannot find antibiotic-resistant k-mers	The model has a 95% accuracy rate.
[33]	SVM, ANN, and kNN	influencing how well algorithms perform	k-NN performed better, achieving an accuracy of 77.15%.
[34]	ANN, NB, SVM, RF, and k-NN	issue with overfitting	RF fared better, with 97.57% accuracy.
[35]	GBDT, RF, DT, and adaptive boosting methods	There wasn't much data gathered from the surveillance system	The GBDT model's accuracy was 69%.
[36]	NN, LB, RF, SVM, and GBM	Dose-response mechanisms limit risk assessment methods	SVM fared better, achieving 89% accuracy.

3. DATASET USED AND PERFORMANCE MEASURE

The Satellite Image Classification (RSI-CB256) dataset (https://www.kaggle.com/datasets/mahmoudreda55/satellite-image-classification) combines Sensor data with a Google Maps picture to form four distinct categories as: Cloudy; Desert; Green; Water.

A collection of 5631 JPEG photos is used as training data. Figure 1 shows the sample images of all four types of images.

The exam uses 80:20 images. Model performance is measured by accuracy, precision, recall, and F1-score.

$$\text{Model's Accuracy } (A_{CC}) = \frac{T_p+T_f}{T_p+T_f+F_p+F_n} \quad (1)$$

$$\text{Model's Precision } (Pre) = \frac{\mathrm{T_p}}{\mathrm{T_p+F_p}} \quad (2)$$

$$\text{Model's Recall } (R_e) = \frac{\mathrm{T_p}}{\mathrm{T_p}+\mathrm{F}_n} \quad (3)$$

$$\text{Model's } F1-Score = \frac{Pre \times R_e}{Pre + R_e} \quad (4)$$

where, true positive, true negative, false positive, and false negative, are represented by T_p, T_f, F_p and F_n respectively.

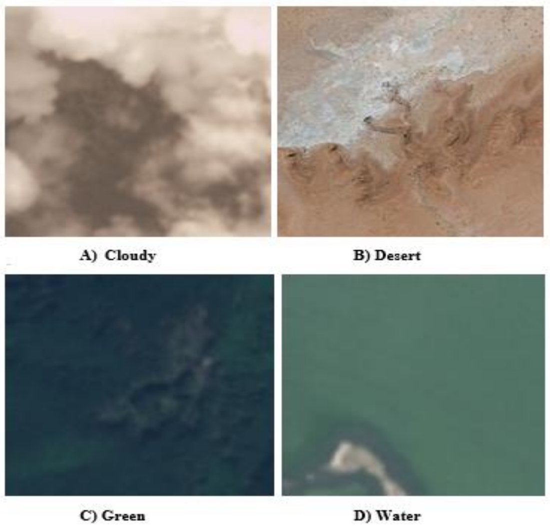


Figure 1. Sample images

4. ENSEMBLE DEEP NEURAL NETWORK FOR SATELLITE IMAGE CLASSIFICATION

The suggested ensemble model combines layered Convolutional Neural Network and SVM algorithm [30-34]. Figure 2 depicts the whole model.

Convolutional Neural Network (CNN) is used for classification, one of the most popular technologies, as shown in Figure 3. Each layer of CNN contains the following sub-layers:

(1). Convolutional Layer (2). Completely Networked Layer (3). The Pooling Layer (4). The Drop-out Layer (5). Linked Dataset Classification Layer.

• **Convolutional Layer**

The convolution procedure is crucial to the convolutional layer, which maps the input picture with a filter of size mm to produce feature maps for the output. In equation form, the result of the convolutional layer may be represented as

$$A_n^m = f\left(\sum_{k\in L_n} A_{kn}^{m-1} * M_{kn}^m + C_n^m\right) \tag{5}$$

where,

Results-based feature maps;

C_n: Bias term;

L_n: Input maps;

M_{kn}: Convolution kernel.

The final feature map's sophistication may be described as:

$$N = \frac{(\mathrm{X} - \mathrm{M} - 2\mathrm{Y})}{\mathrm{T}} \tag{6}$$

Calculating Output Height/Length (N).

Height/Length Input X.

Filter size (M), padding (Y), and stride length (T).

The data may be saved via padding in this case. Eq. (7) describes the padding:

$$Y = \frac{(M-1)}{2} \tag{7}$$

where, M is the size of the filter.

• **The ReLU Layer**

The convolutional process becomes more linear due to this layer's contributions. Hence, a ReLU layer is connected to each convolutional layer in the network. The most important thing that needs to be done in this layer is to ensure that all negative activations are set to zero and that the thresholding is set to maximum (0, p).

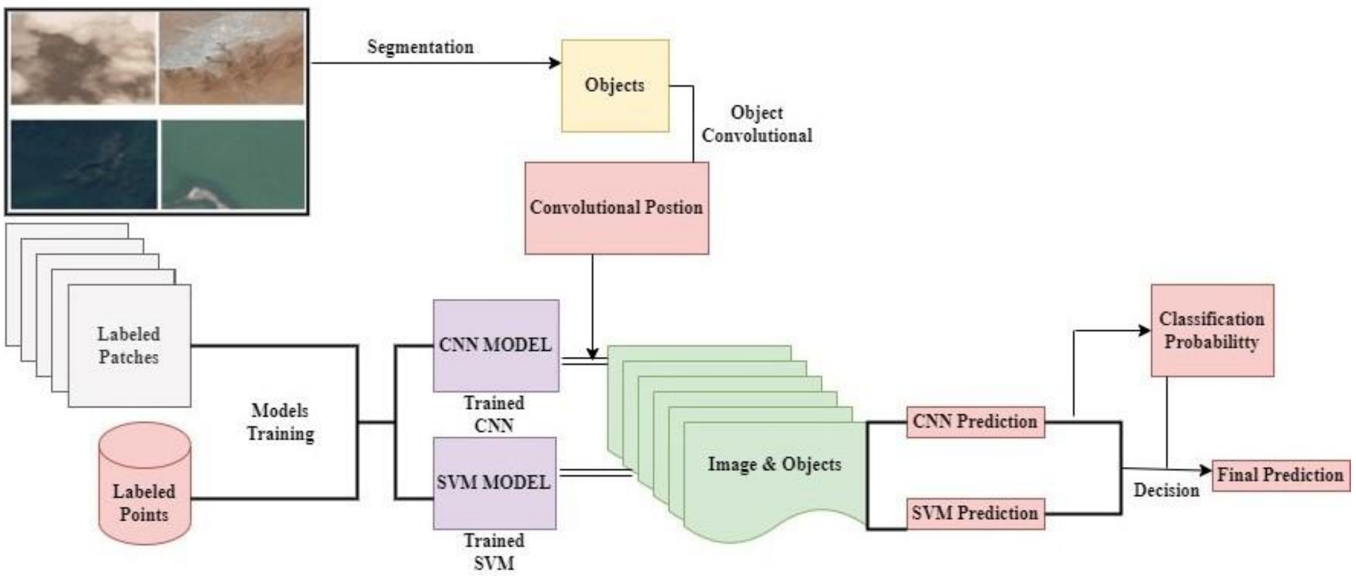


Figure 2. Ensemble deep neural network

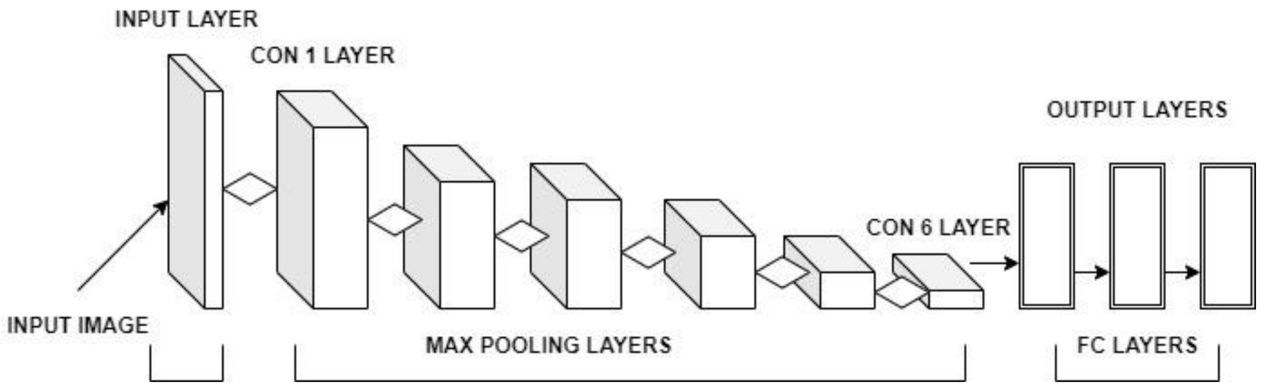


Figure 3. CNN architecture

• **The Max-Pooling Layer**

This layer is responsible for producing the output in a smaller size after the components of each block have been maximized.

• **Dropout Layer**

During the training phase, this layer is used to remove input components whose probability is less than a predetermined threshold.

• **The Batch Normalization Layer**

To standardize the value of the activation layer, this layer does a variety of mathematical operations, including subtraction, division, shifting, and scaling. The Eqs. (8)-(11) may be used to represent the batch normalized result, also known as B_k:

$$\begin{aligned} B_k &= \mathrm{D}O_{\theta_\alpha} \times (A_k) \\ &\equiv \theta\widehat{A_k} + D \end{aligned} \tag{8}$$

where, $\hat{A}_k$ is the settling down of activation A_k.

$$\hat{A}_k = \frac{A_k + U_D}{(\sigma_D^2 + \varepsilon)^{1/2}} \tag{9}$$

where,

ε: constant in nature;

U_D: Mini-batch average;

$\sigma_D{}^2$: Minimal batch variance given by:

$$U_D = \frac{1}{d}\sum_{k=1}^{d} A_k \tag{10}$$

$$\sigma_D^2 = \frac{1}{\mathrm{d}}\sum_{\mathrm{k}=1}^{\mathrm{d}} (\mathrm{A_k} - \mathrm{U_D})^2 \tag{11}$$

• **Completely Connected Layer**

This layer links the neurons of the next layer to those of the layer below it, creating a vector. The vector's dimensions indicate class numbers.

• **The Output Layer**

At this layer, the softmax algorithm is used. The equation that defines the softmax is as follows:

$$P(v_r|A,\theta) = \frac{P(A,\theta|v_r)\, P(v_r)}{\sum_{n=1}^{M} P(A,\theta|v_r) P(v_r)} \tag{12}$$

where, $0 \leq P(v_r|\mathrm{A}\ \theta) \leq 1$ and $\sum_{n=1}^{M} P(v_r|A,\theta) = P(A,\theta|v_r)$ are the conditional and class prior probabilities. Eq. (13) may also be:

$$P(v_r|A,\theta) = \frac{\exp\,[\mathrm{d_r(A,\theta)}]}{\sum_{n=1}^{M} \exp\,[\,d_n(A,\theta)]} \tag{13}$$

written as follows:

$$d_r = \ln\big(P(A,\theta|v_r)\, P(v_r)\big) \tag{14}$$

The output of the layered CNN is used as inputs for the regression model in the following way: The logistic regression model is described as follows: In this section of the model, the feature vector is represented by the letter x, and the outputs are probabilities:

$$\hat{y} = P(y = 1|x) \tag{15}$$

The feature vector denotes each instance of an item that belongs to the class, and each model is represented by one of the RGB channels outlined in Eq. (16):

$$n_x = n_h + n_w + 3 \tag{16}$$

$$\hat{y} = \sigma(z) \tag{17}$$

where,

$$\sigma(z) = \frac{1}{1 + e^z} \tag{18}$$

The logistic function is expressed as z:

$$z = w^T x + b \tag{19}$$

Eqs. (20) and (21) express the loss and cost functions, respectively:

$$\begin{aligned} L(\hat{y}(i), y(i))) = -[y(i)log\hat{y}(i) + (1 \\ - y(i))log(1 - \hat{y}(i))] \end{aligned} \tag{20}$$

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} (\, L(\hat{y}(i), y(i))) \tag{21}$$

where, m represents training examples.

The classification is provided by:

$$\frac{\delta L}{\delta w} = \big(\hat{y}(i) - y(i)\big)x_j(i) and \frac{\delta L}{\delta b} = \hat{y}(i) - y(i) \tag{22}$$

The feature vector is represented by j =1, 2, ..., nx.

The size of each picture is first normalized here in Figure 4 preprocessing so that it conforms to the criterion of 256 pixels by 256 pixels. Python libraries are used to carry out identical operations with the highest possible degree of precision. When the data have been preprocessed, the appropriate acronyms are affixed. Then the data are separated into the various classes that will be utilized for testing afterwards.

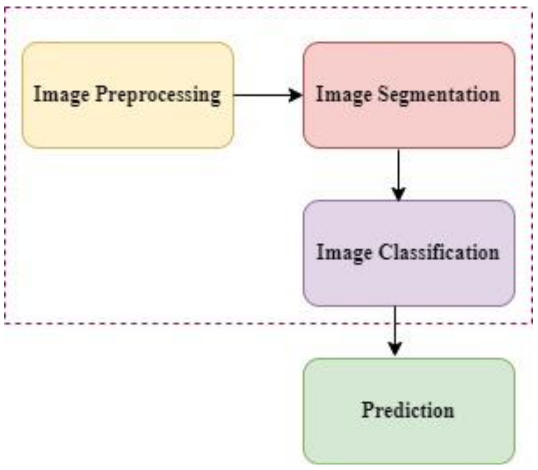


Figure 4. Segmentation and classification

The input layer depicts the picture fed into the CNN at the beginning of the process. The image loaded into the computer is denoted by the formula [height * width * many color channels]. The value of the color channel indicates the kind of picture; for example, the value channel=3 indicates an RGB image. The same input is then run through a data argumentation before being sent to the CNN for processing. The argumentation is carried out using various procedures, including cropping, rotation, and so on. Since the CNN model requires a substantial quantity of data to provide accurate results, the input data are augmented using some process that generates more data to meet its requirements.

5. PROPOSED ENSEMBLE ALGORITHM

The algorithm is summarised in several steps:

Step 1. Define a function named def generate_classification(data).

Step 2. Take Convolutional Size as cnn_svm = Conv3D (4, 32, 5, 1).

Step 3. Set parameter as cnn_svm.set_parameters.

Step 4. Take Input Size as input_shape = (batch_size, data_length, input_dim).

Step 5. Output Shape as output_shape = (batch_size, classification_dim).

Step 6. Set Output Type as output_type = 'linear').

Step 7. Set CNN_SVM to fit data cnn_svm.fit(data, True).

Step 8. Set Classification as classification = cnn_svm.output.

Step 9. Return classification.

Table 2. SVM and CNN tuning parameter

SVM	Kernel	RBF
	Regularization (C)	1
	Gamma (for RBF)	0.1
	Class Weights	'balanced'
CNN	Architecture	Convolutional layers: 3, Dense layers: 2
	Activation	ReLU
	Dropout Rate	0.25
	Batch Size	32
	Learning Rate	0.001
	Optimizer	Adam
	Input Image Size	(256, 256, 3)
Ensemble	Combining Method	Weighting Averaging
	SVM Weight	0.4
	CNN Weight	0.6
Training	Training/Validation Split	80% training, 20% validation
	Training Duration	100 Epoch
	Validation Strategy	K-Fold (5,10,20)
Evaluation	Metrics	Accuracy, precision, recall, F1-score
	Hyperparameter Tuning	Grid search for SVM parameters, random search for CNN parameters

The ensemble of CNN and SVM models may be motivated by the desire to combine the strengths of both approaches. CNNs are proficient in feature extraction from images, while SVMs are known for their strong classification capabilities. Combining these two models in an ensemble could potentially leverage the feature extraction power of CNNs and the discriminative capabilities of SVMs, leading to improved overall performance in tasks like image classification or object recognition.

Tunning parameter used in this proposed model is shown in Table 2.

6. RESULT ANALYSIS AND DISCUSSION

The investigation is done with GPU-based Google Co-Lab and karas libraries written in Python. Experiments are carried out within the scope of this study, with the batch size, epoch, and learning rate all being subjected to change. The investigation uses two different epoch sizes, 50 and 100, and three different learning rates, namely 0.1, 0.001, and 0.0001.

Figure 5 shows the training sample images. The comparison of the training loss to the validation loss, as shown in Figures 6 (a) and 6 (b), and the comparison of the training accuracy to the validation accuracy, using 50 epochs and 0.0001 as the learning rate, are shown. Figure 6 (a) shows the error rate inversely proportional to the amount of model learning. When there is an increase in model learning, there is a corresponding drop in the error rate. As shown in Figure 6 (b), the model's accuracy and the amount of learned information improve. The accuracy of the model is around 98.97% of the time.

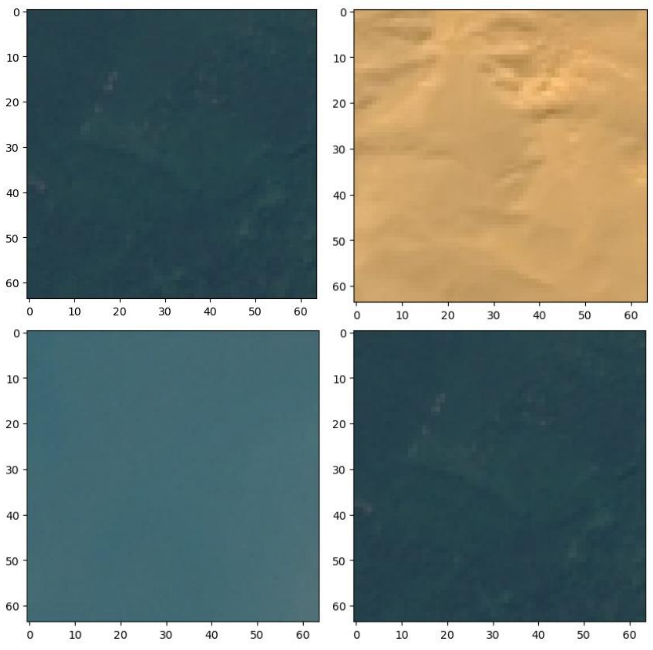
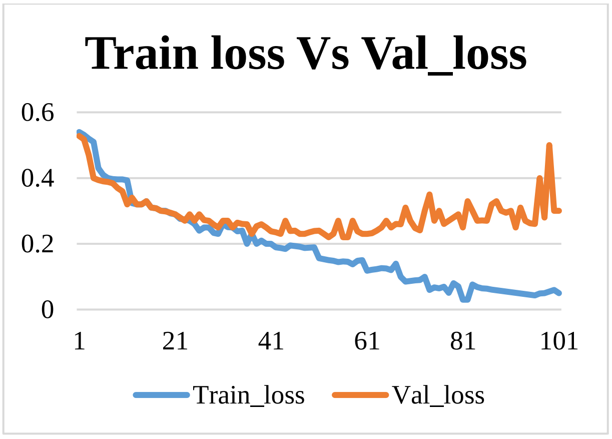
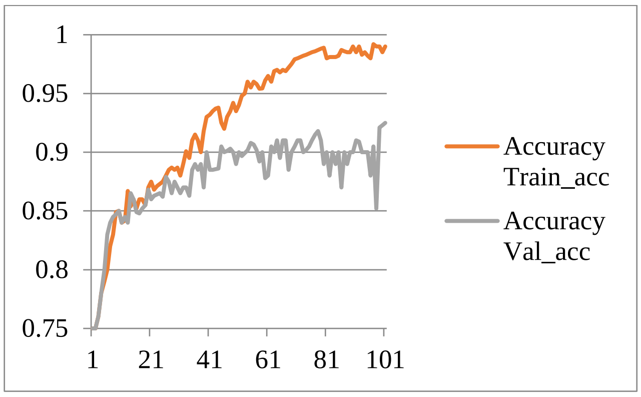


Figure 5. Sample training images

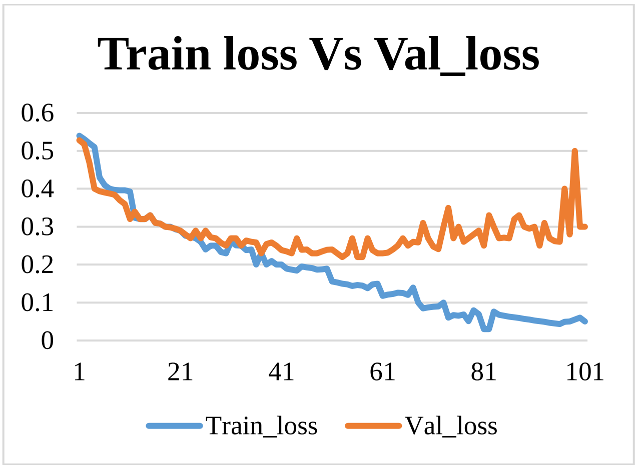


(a)

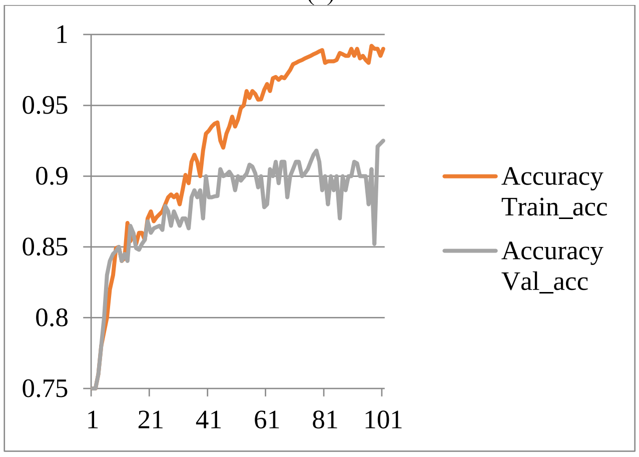


(b)

Figure 6. (a) Loss associated with a learning rate of 0.0001 and 50 epochs; (b) Accuracy with learning rate 0.0001, 50 epochs



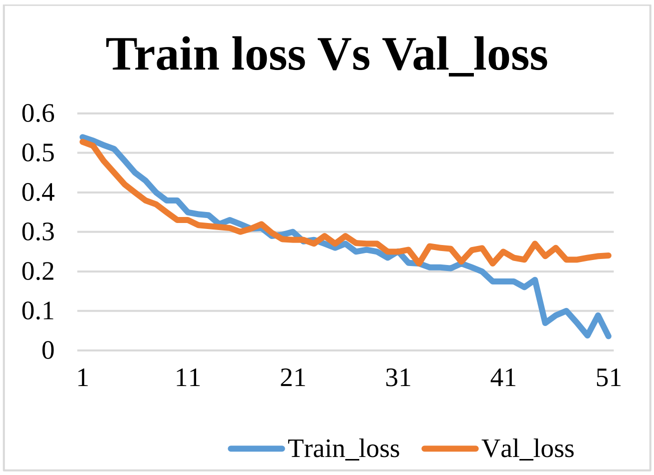
(a)



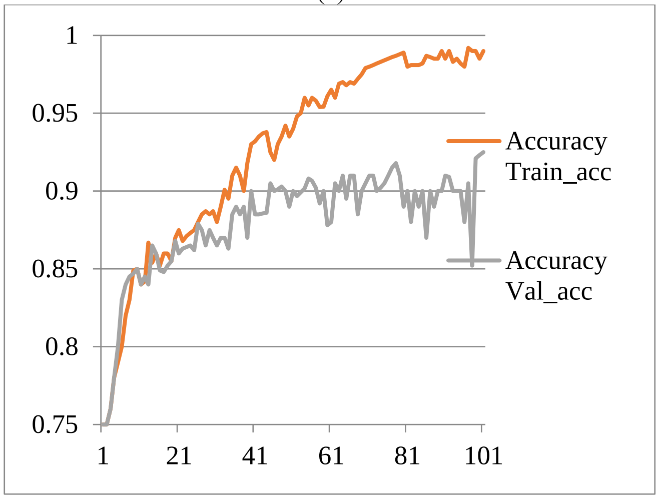
(b)

Figure 7. (a) Loss during 100 epochs at a learning rate of 0.0001; (b) Accuracy with learning rate 0.0001, 100 epochs

Figures 7 (a) and 7 (b) show the difference between the training loss and the validation loss, as well as the difference between the training and validation accuracy, with a learning rate of 0.0001 epochs per second. Figure 7 (a) shows the error rate inversely proportional to the amount of model learning. When there is an increase in model learning, there is a corresponding drop in the error rate. Figure 7 (b) illustrates that the model's accuracy improves as the amount of learned information increases. The accuracy of the model is around 98.90% of the time.



(a)



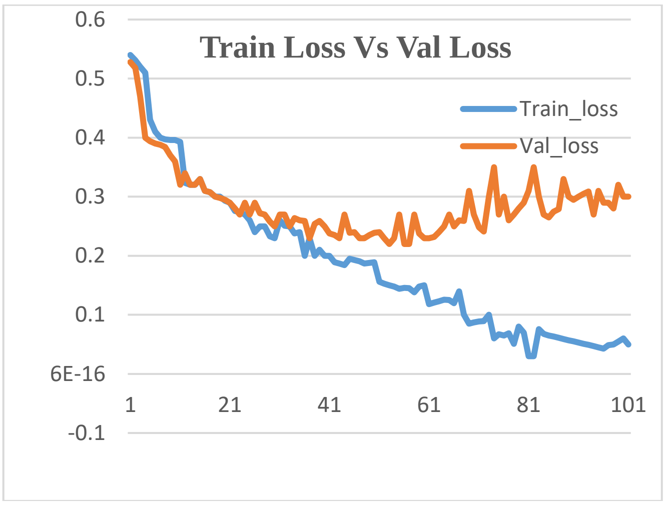
(b)

Figure 8. (a) Loss with learning rate 0.001, 50 epochs; (b) Accuracy with learning rate 0.001, 50 epochs

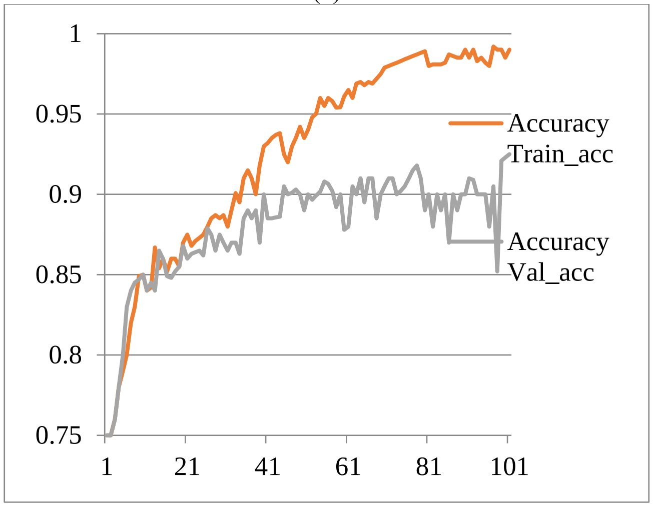
These results show that the computing algorithm acts as a primary role. For example, the weight correction used in the learning rate is computed using a training parameter. Figures 8 (a) and 8 (b) demonstrate the contrast between training loss/validation loss and training accuracy/validation accuracy on 50 epochs and a 0.001 learning rate. Figure 8 (a) depicts the error rate inverse to the learning of the model. Whenever model learning increases, the error rate decreases. Figure 8 (b) illustrates that the model accuracy increases as knowledge increases. The accuracy of the model reaches 98.98%.

Figures 9 (a) and 9 (b) show the difference between training loss and validation loss, as well as between training accuracy and validation accuracy, with a learning rate of 0.001 on 100 iterations. The error rate inversely related to the learning of the model is shown in Figure 9 (a). When there is an increase in model learning, there is a corresponding drop in the error rate. Figure 9 (b) illustrates that the model's accuracy improves as the amount of learned data increases. The accuracy of the model is around 98.99% of the time.

Figures 10 (a) and 10 (b) compare training loss/validation loss and training accuracy/validation accuracy on 50 epochs and 0.001 learning rate. Figure 10 (a) depicts the error rate inverse to the learning of the model. Whenever model learning increases, the error rate decreases. Figure 10 (b) illustrates that the model accuracy increases as learning rises. The accuracy of the model reaches 98.98%.



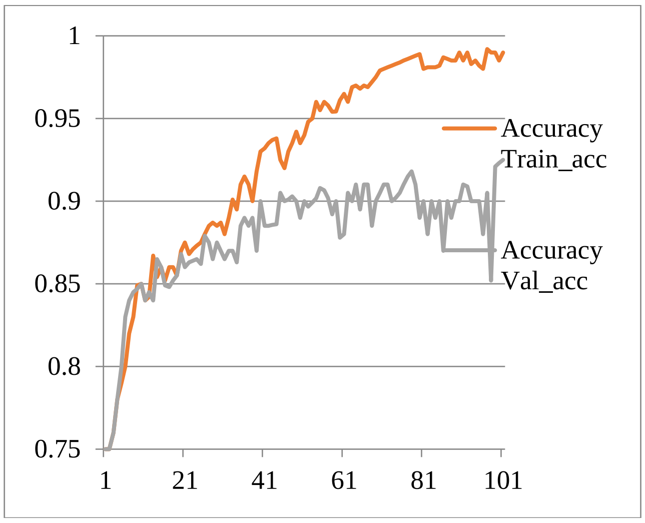
(a)



(b)

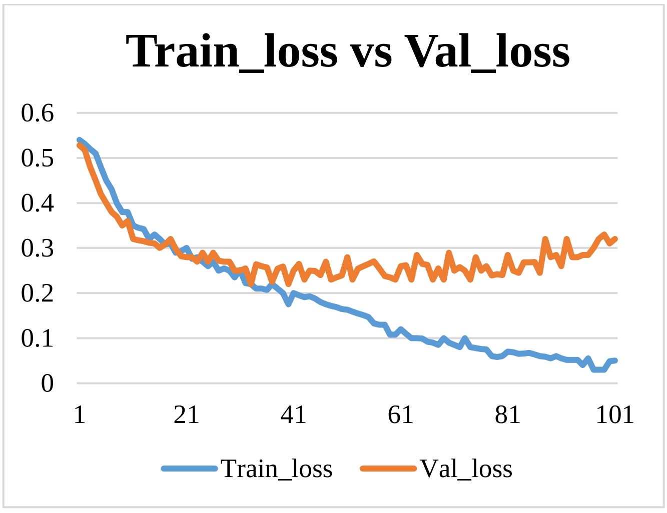
Figure 9. (a) Loss with learning rate 0.001, 100 epochs; (b) Accuracy with learning rate 0.001, 100 epochs

Figures 11 (a) and 11 (b) show the comparison between training loss and validation loss at 100 epochs and 0.01 learning rate. Figure 11 (a) shows the error rate as a function of model learning. Error rates go down as model learning speeds up. Figure 11 (b) shows that the model becomes more accurate with more training. The model achieves a 98.99% degree of accuracy.

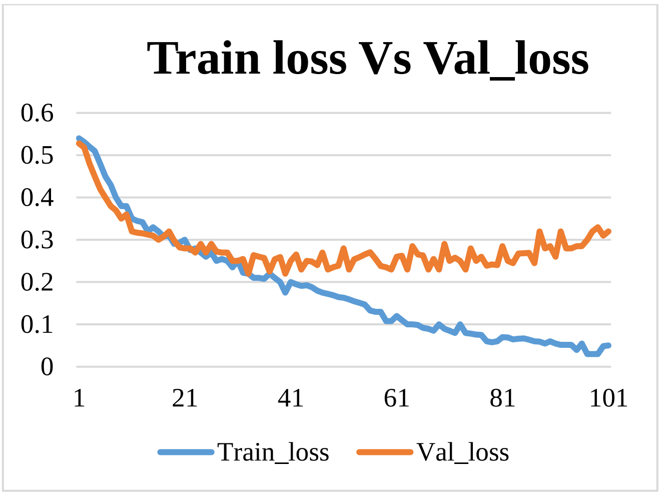


(b)

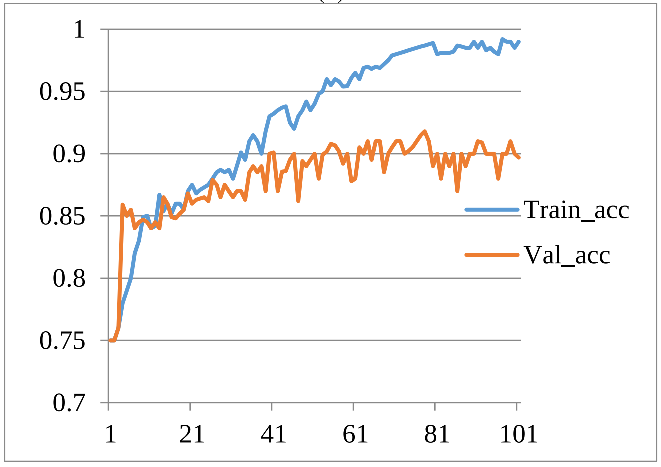
Figure 10. (a) Loss with learning rate 0.001, 50 epochs; (b) Accuracy with learning rate 0.01, 50 epochs



(a)



(a)



(b)

Figure 11. (a) Loss with learning rate 0.01, 100 epochs; (b) Accuracy with learning rate 0.01, 100 epochs

Table 3 shows that a dataset and meaningful epoch and learning rate settings were used to train the model. The dataset and the relevant epoch value affect the experiment's results. An exact outcome may be seen in the value of the critical epoch. Several epochs and learning rates are used in the investigation. This paper provides results for two different period lengths and three different learning rate settings. The experiment results with two epochs and three learning rate settings are shown in Table 3. The envisioned paradigm contrasts with the gold standard classification model and CNN based hybrid model. The results, summarized in Table 4, show that the suggested model is superior to the alternatives. Table 5 shows the comparison with literature and other state of art algorithms.

The results show that the suggested neural network ensemble performs better than competing models. Accuracy, F1, Recall, and precision value are only a few of the metrics that have been compared.

Table 3. Tuning parameter values and accuracy

Dataset Size	Dimension	Epoch	LR	Accuracy (%)
5631	256×256 px	50	0.0001	98.97%
		50	0.001	98.90%
		50	0.01	98.98%
		100	0.0001	98.99%
		100	0.001	98.98%
		100	0.01	98.99%

Table 4. Comparative analysis

	Model	AUC	CA	F1	Precision	Recall
K FOLD 5	SVM	0.9992	0.9856	0.9856	0.9857	0.9856
	CNN	0.9997	0.9895	0.9895	0.9895	0.9895
	Logistic Regression	0.9999	0.9929	0.9929	0.9929	0.9929
	AdaBoost	0.9469	0.9208	0.9208	0.9208	0.9208
	SVCNN (Proposed Model)	0.9997	0.9896	0.9896	0.9896	0.9896
K FOLD 10	SVM	0.9992	0.9856	0.9856	0.9857	0.9856
	CNN	0.9997	0.9895	0.9895	0.9895	0.9895
	Logistic Regression	0.9999	0.9929	0.9929	0.9929	0.9929
	AdaBoost	0.9469	0.9208	0.9208	0.9208	0.9208
	SVCNN (Proposed Model)	0.9997	0.9896	0.9892	0.9895	0.9895
K FOLD 20	SVM	0.9990	0.9853	0.9853	0.9853	0.9853
	CNN	0.9997	0.9897	0.9897	0.9897	0.9897
	Logistic Regression	0.9999	0.9934	0.9934	0.9934	0.9934
	AdaBoost	0.9475	0.9217	0.9217	0.9218	0.9217
	SVCNN (Proposed Model)	0.9997	0.9897	0.9897	0.9897	0.9897

Table 5. Comparative analysis with literature and other state of art algorithms

Reference	Model	Results
[30]	Adaboost, XGBoost, GBDT, LR, DT, RF, SVM, NB, LR	Recall=0.9699 and F1-score =0.9582, algorithm shows the highest performance.
[31]	Neural Networks, NB, k-NN, DT, and SVM	Both kNN and SVM algorithms worked 95.56% of the time.
[32]	Boost XG	The model has a 95% accuracy rate.
[33]	SVM, ANN, and kNN	k-NN performed better, achieving an accuracy of 77.15%.
[34]	ANN, NB, SVM, RF, and k-NN	RF fared better, with 97.57% accuracy.
[35]	GBDT, RF, DT, and adaptive boosting methods	The GBDT model's accuracy was 69%.
[36]	NN, LB, RF, SVM, and GBM	SVM fared better, achieving 89% accuracy.
	Proposed (CNN+SVM)	98.96%

7. CONCLUSION, LIMITATION AND FUTURE WORK

The discovery of Cloudy area, Desert area, Green area and Water area is an exciting topic of study that has been investigated using hybrid approach. This body of work explores a further method of locating Cloudy area, Desert area, Green area and Water area by using solid artificial intelligence model. In this paper, we draw a comparison between various models for the detection of Cloudy area, Desert area, Green area and Water area by using the intensity of light (flux) along with Artificial Intelligence techniques and Machine Learning algorithms. In addition to that, an Ensemble-CNN model was presented for the same thing. Our suggested model exceeds all of them with an accuracy of 98.96%, but most portray good outcomes. 98.96% and 98.97% in the K fold, correspondingly 5, 10, and 20.

This model classifies only four types of data only if there will change in dataset, this model will not work properly.

In the future, one of our goals is to research additional machine learning models and artificial intelligence methods to locate Cloudy area, Desert area, Green area and Water area.

REFERENCES

[1] LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep learning. Nature, 521(7553): 436-444. https://doi.org/10.1038/nature14539

[2] ImageNet, I.L.S.V.R.C. (2010). Large scale visual recognition challenge (ILSVRC). http://www.image-net.org/challenges/LSVRC.

[3] Lowe, D.G. (2004). Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60: 91-110. https://doi.org/10.1023/B:VISI.0000029664.99615.94

[4] Dalal, N., Triggs, B. (2005). Histograms of oriented gradients for human detection. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 1: 886-893. https://doi.org/10.1109/CVPR.2005.177

[5] LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D. (1989). Backpropagation applied to handwritten zip code recognition. Neural Computation, MIT Press, 1(4): 541-

551. https://doi.org/10.1162/neco.1989.1.4.541

[6] Krizhevsky, A., Sutskever, I., Hinton, G.E. (2012). Imagenet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, Lake Tahoe, Nevada, 25.

[7] Simonyan, K., Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv Preprint arXiv, 1409.1556. https://doi.org/10.48550/arXiv.1409.1556

[8] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, pp. 1-9. https://doi.org/10.1109/CVPR.2015.7298594

[9] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, pp. 2818-2826. https://doi.org/10.1109/CVPR.2016.308

[10] He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, pp. 770-778. https://doi.org/10.1109/CVPR.2016.90

[11] Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q. (2017). Dense connected convolutional neural networks. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, pp. 2261-2269. https://doi.org/10.1109/CVPR.2017.243

[12] Tensorflow: An open-source software library for machine intelligence. https://www.tensorflow.org/, accessed on Jan. 21, 2017.

[13] Keras, GitHub. https://github.com/fchollet/keras, accessed on June 25, 2023.

[14] He, K., Zhang, X., Ren, S., Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 37(9): 1904-1916. https://doi.org/10.1109/TPAMI.2015.2389824

[15] "UC Merced Land Use Dataset," University of California, Merced. http://weegee.vision.ucmerced.edu/datasets/landuse.html.

[16] Yang, Y., Newsam, S. (2010). Bag-of-visual-words and spatial extensions for land-use classification. In Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 270-279. https://doi.org/10.1145/1869790.1869829

[17] Liang, Y., Monteiro, S.T., Saber, E.S. (2016). Transfer learning for high resolution aerial image classification. In 2016 IEEE Applied Imagery Pattern Recognition Workshop (AIPR), Washington, DC, USA, pp. 1-8. https://doi.org/10.1109/AIPR.2016.8010600

[18] Saini, D., Garg, R., Malik, R., Prashar, D., Faheem, M. (2024). HFRAS: Design of a high-density feature representation model for effective augmentation of satellite images. Signal, Image and Video Processing, 18(2): 1393-1404. https://doi.org/10.1007/s11760-023-02859-7

[19] Pérez, A.F., Maghoul, P., Ashraf, A. (2024). A deep learning approach to satellite image time series coregistration through alignment of road networks. Neural Computing and Applications, 36(7): 3583-3593. https://doi.org/10.1007/s00521-023-09242-0

[20] SpaceNet on AWS. https://aws.amazon.com/public-datasets/spacenet/, accessed on June. 25, 2023.

[21] Chartock, E., LaRow, W., Singh, V. (2017). Extraction of building footprints from satellite imagery. Stanford University Report.

[22] Cheng, G., Han, J., Lu, X. (2017). Remote sensing image scene classification: Benchmark and state of the art. Proceedings of the IEEE, 105(10): 1865-1883. https://doi.org/10.1109/JPROC.2017.2675998

[23] Ciecholewski, M. (2024). Review of segmentation methods for coastline detection in SAR Images. Archives of Computational Methods in Engineering, 31(2): 839-869. https://doi.org/10.1007/s11831-023-10000-7

[24] Zhang, F., Du, B., Zhang, L. (2014). Saliency-guided unsupervised feature learning for scene classification. IEEE Transactions on Geoscience and Remote Sensing, 53(4): 2175-2184. https://doi.org/10.1109/TGRS.2014.2357078

[25] Zou, Q., Ni, L., Zhang, T., Wang, Q. (2015). Deep learning based feature selection for remote sensing scene classification. IEEE Geoscience and Remote Sensing Letters, 12(11): 2321-2325. https://doi.org/10.1109/LGRS.2015.2475299

[26] 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 Bellevue, WA, USA, pp. 1-10. https://doi.org/10.1145/2820783.2820816

[27] Ju, C., Bibaut, A., van der Laan, M. (2018). The relative performance of ensemble methods with deep convolutional neural networks for image classification. Journal of Applied Statistics, 45(15): 2800-2818. https://doi.org/10.1080/02664763.2018.1441383

[28] Albert, A., Kaur, J., Gonzalez, M.C. (2017). Using convolutional networks and satellite imagery to identify patterns in urban environments at a large scale. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax NS Canada, pp. 1357-1366. https://doi.org/10.1145/3097983.3098070

[29] Robinson, C., Hohman, F., Dilkina, B. (2017). A deep learning approach for population estimation from satellite imagery. In Proceedings of the 1st ACM SIGSPATIAL Workshop on Geospatial Humanities, Beach CA USA, pp. 47-54. https://doi.org/10.1145/3149858.3149863

[30] Zhang, P., Cui, W., Wang, H., Du, Y., Zhou, Y. (2021). High-efficiency machine learning method for identifying foodborne disease outbreaks and confounding factors. Foodborne Pathogens and Disease, 18(8): 590-598. https://doi.org/10.1089/fpd.2020.2913

[31] Min, H.J., Mina, H.A., Deering, A.J., Bae, E. (2021). Development of a smartphone-based lateral-flow imaging system using machine-learning classifiers for detection of Salmonella spp. Journal of Microbiological Methods, 188: 106288. https://doi.org/10.1016/j.mimet.2021.106288

[32] Nguyen, M., Long, S.W., McDermott, P.F., Olsen, R.J., Olson, R., Stevens, R.L., Tyson, G.H., Zhao, S., Davis,

J.J. (2018). Using machine learning to predict antimicrobial minimum inhibitory concentrations and associated genomic features for nontyphoidal Salmonella. BioRxiv, 380782. https://doi.org/10.1101/380782

[33] Polat, H., Topalcengiz, Z., Danyluk, M.D. (2020). Prediction of Salmonella presence and absence in agricultural surface waters by artificial intelligence approaches. Journal of Food Safety, 40(1): e12733. https://doi.org/10.1111/jfs.12733

[34] Amado, T.M., Bunuan, M.R., Chicote, R.F., Espenida, S.M.C., Masangcay, H.L., Ventura, C.H., Tolentino, L.K.S., Padilla, M.V.C., Madrigal, G.A.M., Enriquez, L.A.C. (2019). Development of predictive models using machine learning algorithms for food adulterants bacteria detection. In 2019 IEEE 11th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM), Laoag, Philippines, pp. 1-6. https://doi.org/10.1109/HNICEM48295.2019.9072907

[35] Wang, H., Cui, W., Guo, Y., Du, Y., Zhou, Y. (2021). Machine learning prediction of foodborne disease pathogens: Algorithm development and validation study. JMIR Medical Informatics, 9(1): e24924. https://doi.org/10.2196/24924

[36] Njage, P.M.K., Henri, C., Leekitcharoenphon, P., Mistou, M.Y., Hendriksen, R.S., Hald, T. (2019). Machine learning methods as a tool for predicting risk of illness applying next-generation sequencing data. Risk Analysis, 39(6): 1397-1413. https://doi.org/10.1111/risa.13239