DEEP LEARNING MODELS FOR CLASSIFICATION OF REMOTE SENSED IMAGES

Manasvi.A, Srimansi Ramesh Kumar, Dr. Revathy G

ABSTRACT

Systems for classifying land cover and land use (LCLU) based on deep learning are highly desired by the remote sensing community. Remotely sensed photographs of the natural world need to have their many different characteristics examined. Analyzing and understanding image attributes can be challenging due to factors such as nature of image, capabilities of the sensors used, and other elements such as seasons and weather. Deep learning algorithms are suggested to evaluate and comprehend remotely sensed images. Analyzing these images can be used to address the issues like urban planning, agricultural decision making etc.

However, there has been very little comparison research conducted on training pre-trained networks with deep learning models that have been developed from scratch. Thus, we propose comparing different deep learning models like convolution neural network feature extractor developed from scratch, transfer learning and fine-tuning different architectures such as Inceptionv3, DenseNet, MobileNet, EfficientNet, ResNet for land use land cover classification. After being created and trained on the UCM dataset, the performances of each deep learning model were compared using performance assessment metrics like recall, accuracy, f1-score, precision, and confusion matrix. After analyzing and comparing the performances of all the models the paper discovered that the fine-tuned deep learning model outperformed all other deep learning models in terms of accuracy performance on the UCM dataset.

INDEX TERMS: Remote sensed image classification, Convolutional neural network, Deep learning, Performance comparison, Transfer learning, Fine-tuning

I. INTRODUCTION

The LCLU categorization learning system is largely used in today's fast-paced world for environmental monitoring, agricultural decision-making, and urban planning. A crucial difficulty is the classification of the LCLU using remotely sensed (RS) images. The demand for land use is rising as the world's population is growing substantially. To make the best use of this area, a learning system may be implemented. Using cutting-edge sensor technology, remotely sensed images are satellite data that are gathered from the environment of the earth. The difficulty might be resolved using the deep learning (DL) technique.

The deep learning is a modern, specialized machine learning (ML) method that has impressive performance gains for automatically extracting picture characteristics from big datasets. As a result, deep learning (DL) is a recently-focused study topic employed in many domains, including object identification, recognition, and classification. Convolutional neural networks (CNNs), transfer learning (TL), and fine-tuning are DL techniques that are suggested in this research and improve the appeal of the classification problem.

CNN (Convoluted Neural network) is a deep learning architecture consisting of convolutional calculations and deep structures due to which they are powerful feature extractors and are a good solution for classification performance improvement in remotely sensed images. For larger datasets training the CNN from scratch takes up long time and a large amount of processing power.

To overcome these limitations, we use approaches like Transfer learning and Fine Tuning using pre-trained models. In Transfer learning we reuse an already existing pre-trained model instead of training the model from scratch which in turn saves a lot of time and computational power. In Transfer learning method only the fully connected layer is trained and the dense layer is replaced by a classifier with neurons equal to the number of classes that the model needs to classify, i.e., 21 neurons and the activation layer is softmax as it is a multiclass classification. Fine tuning is a slight variation of transfer learning where additionally the pre-trained layer is also trained.

Various pre-trained models are used to compare and analyse the performance of the models. Models used are InceptionV3, DenseNet201, MobileNet, EfficientNetB7 and ResNet152V2. InceptionV3, also known as GoogLeNetv3 introduced factorization and global average pooling to reduce the number of parameters while being computationally efficient and achieving excellent performance. It outperforms VGG16 for larger datasets and is proved to give good results in image classification and object detection tasks. DenseNet201 has a compact architecture due to its parameter-efficient design in which it uses dense blocks, where each layer receives direct input from all preceding layers, promoting feature reuse and reducing vanishing gradient issues. It is known for its efficiency and strong generalization capabilities. MobileNet is a lightweight and optimized for low latency and resource-constrained environments, they utilize depth-wise separable convolutions to reduce the number of computations significantly. EfficientNetB7 is known for its state-of-the-art performance in various computer vision tasks, including image classification and object detection. This model uses a compound scaling method to balance width, depth, and resolution for optimal performance. ResNet152V2 is an extended version of ResNet where it has 152 layers and introduces several architectural modifications, including bottleneck blocks and skip connections. Additionally, it addressed the vanishing gradient problem by utilizing skip connections, enabling the training of very deep networks.

In this paper the dataset is trained on all these architectures and they are compared and evaluated. The Convoluted Neural Network has been developed from scratch which contains three convolutional blocks. The pre-trained models were trained on “ImageNet” large dataset. EfficientNetB7 was selected from the EfficientNet family as it achieved the 84.4% accuracy on “ImageNet” which remains to be the highest among the EfficientNet family.

Various evaluations metrics such as accuracy, confusion matrix, precision, recall, f1-score and support.

II. DATASET AND TOOLS

University of California Merced (UCM) dataset was used in this paper on different architectures. This dataset is publicly available. It is a land cover and land use dataset collected and manually labelled. There are 21 categories/classes into which the entire dataset is divided. Each class in the dataset contains 500 images each with a resolution of 256 x 256 pixels. Classes of the dataset include images of agriculture, airplane, beach, tennis court and etc.

Python language was used for this model development. Libraries such as Pandas, TensorFlow and Keras were used.

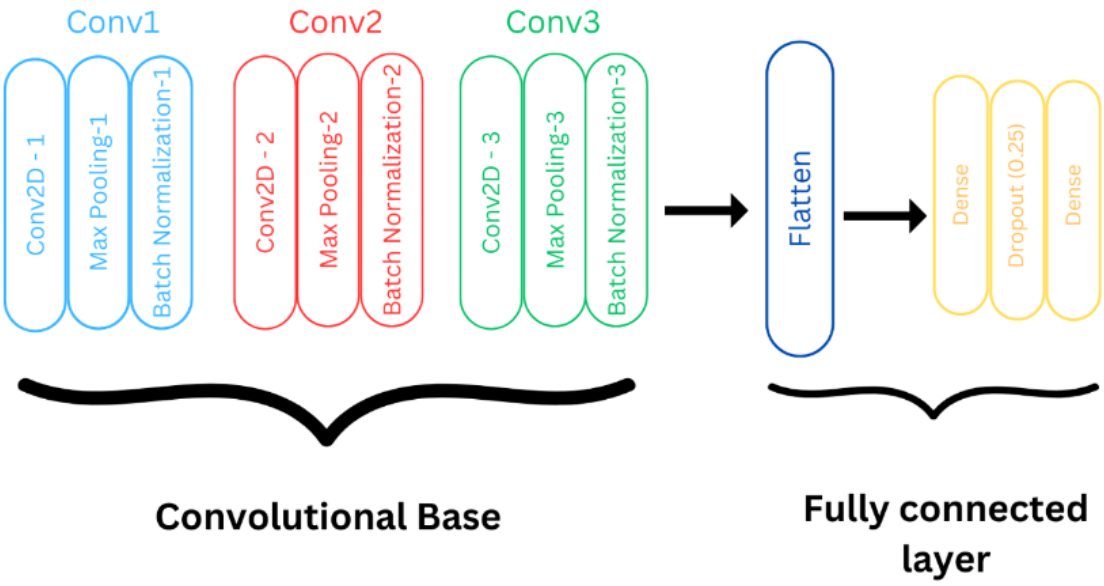
Tool used are Google colab and Kaggle as notebooks

III. PROPOSED METHODS

Works that were previously published where limited to few architectures whereas evaluating and comparing the DL models on a larger range of pre-trained model and CNN model developed from scratch were not widely investigated.

1) THE CONVOLUTIONAL NEURAL NETWORK (CNN) ALGORITHM

A Convolutional Neural Network (CNN) is a type of Deep Learning architecture commonly used for image classification and recognition tasks. It consists of multiple layers, including Convolutional layers, Pooling layers, and fully connected layers. The Convolutional layer applies filters to the input image to extract features, the Pooling layer down samples the image to reduce computation, and the fully connected layer makes the final prediction.



In this paper we have built the CNN model from scratch with three sets of conv2D, maxpooling and batch normalisation layers. Following these sets there is a flatten layer, two dense layers and dropout layer.

2) TRANSFER LEARNING METHOD:

In transfer learning, the deep learning model is replaced by an image embedding that transfers the knowledge learned from the ImageNet dataset to our classification problem. In this paper, we have used image embeddings like EfficientNetB7, DenseNet, MobileNet, Inceptionv3, Resnet15V2 and compared the performances of these architectures.
Transfer learning has been trained by making the layers of the pre trained architecture as non-trainable i.e., pre_trained_model.trainable=false.
We trained the model by adding one global average pooling layer and two dense layers with relu and softmax as the activation functions.

Pre-trained architectures

The following pre-trained networks can classify images into 1000 categories like keyboard, mouse, animals etc. Thus, the network has a rich feature representation for a wide range of images.

1. Inceptionv3:

The pre-trained model is a 48-layer deep convolutional neural network model. Input size for this model is 299-by-299. In the first part, it extracts general features from input images and in the second part it classifies the images based on the features from first part.

2. DenseNet201:

The pre-trained model is 201-layer deep convolutional neural network model. Input size for this model is 224-by-224. This network has densely connected layers that can capture complex dependencies between features and is suitable for classifying highly grained images.

3. MobileNet:

MobileNet is a lightweight architecture. This network has 30 layers with a convolutional layer with stride 2, a depth wise layer, a pointwise layer that doubles the number of channels, depth wise layer with stride 2 and a pointwise layer that doubles the number of channels. Input size for this model is 224-by-224.

4. EfficientNetB7:

The pre-trained model is a 813-layer deep convolutional neural network. This model is a mobile friendly convolutional model that uses an efficient scaling method. This scaling method scales the dimensions of depth/width and resolution using the effective compound coefficient.

5. ResNet152V2:

   The pre-trained model is a 152-layer deep convolutional neural network. This network addresses the problem of vanishing gradient descents by introducing residual connections. These connections allow the model to skip certain layers and pass the input to deeper layers. This enables the network to learn more complex representations and improve the gradient flow during training.

IV. EXPERIMENTAL RESULTS AND DISCUSSIONS

The training and experimentation were done on a laptop with Intel Core i3-4000M CPU with 4GB RAM connected to Colaboratory in its Tesla K80 GPU. The dataset was split into training, testing and validation dataset at 60%, 20% and 20% respectively. Hyper parameters were set with respective to each model for the better accuracy.

The model was trained and evaluated using accuracy, precision, recall, f1-score, and confusion matrix metrics. Using Confusion matrix metrics we can assess the class performance individually, whether the predicted result is correctly classified or not. Categorical-cross-entropy loss function was used to figure out the errors and enhance the accuracy.

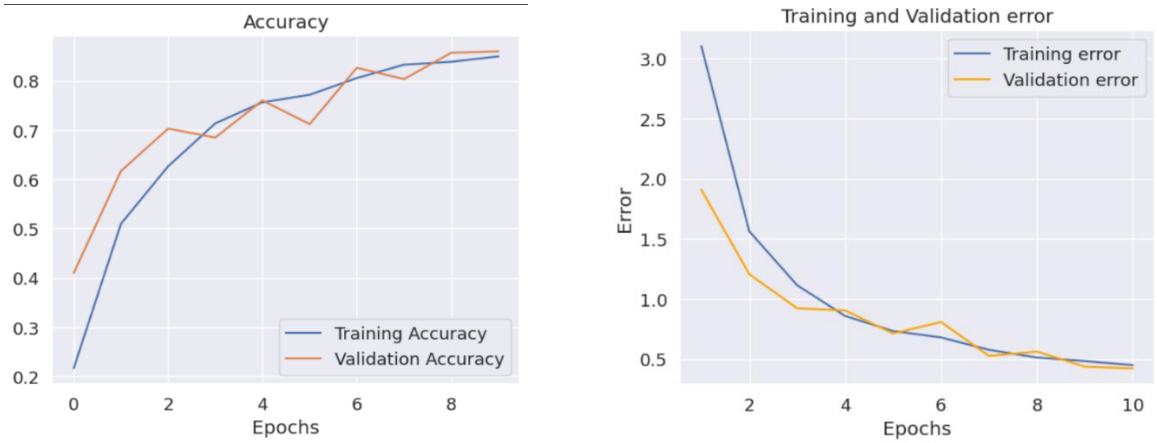


Figure 2. Training and validation accuracies and losses in InceptionV3

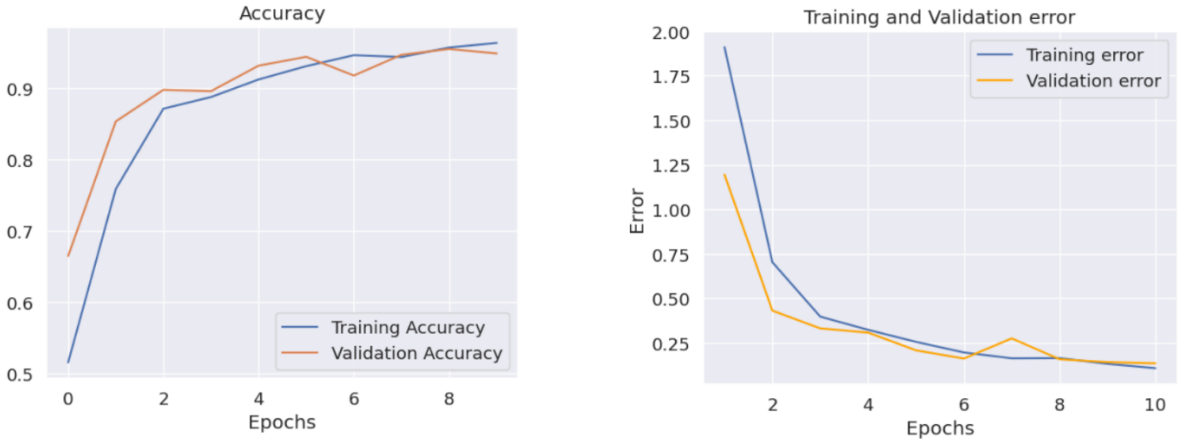


Figure 3. Training and validation accuracies and losses in MobileNet

Figure 4. Training and validation accuracies and losses in ResNet152V2

Figure 5. Training and validation accuracies and losses in DenseNet201

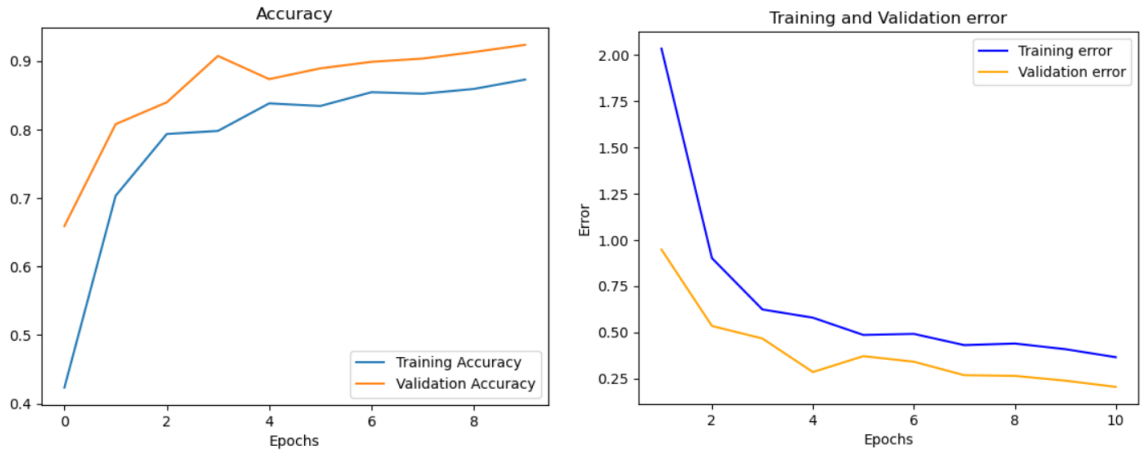


Figure 6. Training and validation accuracies and losses in EfficientNetB7

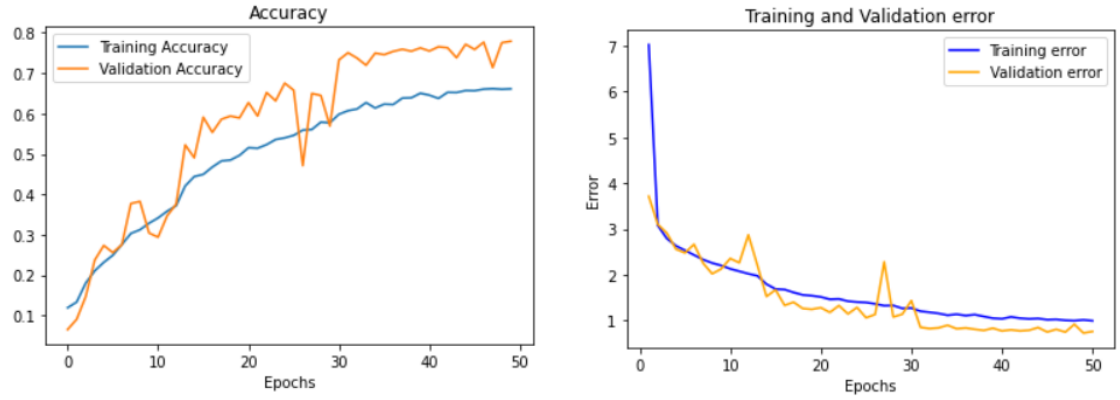


Figure 7. Training and validation accuracies and losses in CNN

We have used metrics like f1-score which is the harmonic mean of precision and recall to generalize the performance of each class and the average performance of the model. Accordingly, the category of buildings has a perfect f1-score of 100% in the models built on pre-trained architectures. Other categories like agriculture, golf course and overpass also have perfect f1-scores in few pre trained architecture models.

The accuracy vs epochs graphs show the variation of training and validation accuracy with epochs. The blue line depicts the training accuracy whereas the orange line depicts the validation accuracy. Both the lines seem to increase with an increase in epochs but with slight fluctuations.

Table 2. Classification report of Inceptionv3

	precision	recall	f1-score	support
0	1.00	0.94	0.97	50
1	1.00	0.86	0.92	50
2	0.87	0.92	0.89	50
3	0.98	0.96	0.97	50
4	0.73	0.86	0.79	50
5	1.00	1.00	1.00	50
6	0.91	0.40	0.56	50
7	0.94	0.98	0.96	50
8	0.96	0.86	0.91	50
9	0.96	0.88	0.92	50
10	0.94	0.98	0.96	50
11	0.93	0.52	0.67	50
12	0.69	0.82	0.75	50
13	0.73	0.88	0.80	50
14	0.93	0.80	0.86	50
15	1.00	0.92	0.96	50
16	0.93	0.84	0.88	50
17	0.65	1.00	0.79	50
18	0.77	0.96	0.86	50
19	0.95	0.82	0.88	50
20	0.65	0.90	0.76	50
accuracy			0.86	1050
macro avg	0.88	0.86	0.86	1050
weighted avg	0.88	0.86	0.86	1050

Table 3. Classification report of MobileNet

	precision	recall	f1-score	support
0	1.00	1.00	1.00	50
1	0.98	0.98	0.98	50
2	0.98	0.92	0.95	50
3	1.00	0.90	0.95	50
4	0.84	0.92	0.88	50
5	1.00	1.00	1.00	50
6	0.84	0.94	0.89	50
7	0.96	0.96	0.96	50
8	0.88	1.00	0.93	50
9	0.98	0.88	0.93	50
10	1.00	1.00	1.00	50
11	0.94	0.98	0.96	50
12	0.97	0.66	0.79	50
13	0.98	0.98	0.98	50
14	1.00	0.84	0.91	50
15	1.00	0.96	0.98	50
16	0.78	1.00	0.88	50
17	0.98	1.00	0.99	50
18	0.89	0.98	0.93	50
19	0.96	0.96	0.96	50
20	0.98	0.96	0.97	50
accuracy			0.94	1050
macro avg	0.95	0.94	0.94	1050
weighted avg	0.95	0.94	0.94	1050

Table 4. Classification report of ResNet

	precision	recall	f1-score	support
0	1.00	1.00	1.00	50
1	0.98	1.00	0.99	50
2	0.94	0.98	0.96	50
3	0.96	0.98	0.97	50
4	0.82	1.00	0.90	50
5	1.00	1.00	1.00	50
6	0.83	0.78	0.80	50
7	0.96	0.98	0.97	50
8	0.57	1.00	0.72	50
9	0.86	0.98	0.92	50
10	1.00	0.98	0.99	50
11	1.00	0.84	0.91	50
12	0.85	0.90	0.87	50
13	0.98	0.84	0.90	50
14	1.00	0.30	0.46	50
15	1.00	1.00	1.00	50
16	0.96	0.94	0.95	50
17	0.94	0.96	0.95	50
18	0.96	0.94	0.95	50
19	0.96	0.90	0.93	50
20	1.00	0.88	0.94	50
accuracy			0.91	1050
macro avg	0.93	0.91	0.91	1050
weighted avg	0.93	0.91	0.91	1050

Table 5. Classification report of EfficientNet

	precision	recall	f1-score	support
0	1.00	1.00	1.00	50
1	0.98	1.00	0.99	50
2	0.98	0.94	0.96	50
3	0.96	0.98	0.97	50
4	0.88	0.90	0.89	50
5	1.00	1.00	1.00	50
6	0.79	0.54	0.64	50
7	0.98	0.98	0.98	50
8	0.92	0.98	0.95	50
9	0.98	0.96	0.97	50
10	1.00	0.98	0.99	50
11	0.96	0.96	0.96	50
12	0.76	0.74	0.75	50
13	0.97	0.70	0.81	50
14	0.98	0.92	0.95	50
15	0.98	1.00	0.99	50
16	0.92	0.94	0.93	50
17	1.00	0.98	0.99	50
18	0.79	1.00	0.88	50
19	0.92	0.94	0.93	50
20	0.72	0.96	0.82	50
accuracy			0.92	1050
macro avg	0.93	0.92	0.92	1050
weighted avg	0.93	0.92	0.92	1050

Table 7. Classification report of DenseNet

	precision	recall	f1-score	support
0	1.00	1.00	1.00	50
1	0.98	0.98	0.98	50
2	0.98	0.96	0.97	50
3	0.96	0.96	0.96	50
4	0.93	0.78	0.85	50
5	1.00	1.00	1.00	50
6	1.00	0.42	0.59	50
7	0.94	0.98	0.96	50
8	0.96	0.88	0.92	50
9	0.94	0.96	0.95	50
10	1.00	1.00	1.00	50
11	0.78	0.98	0.87	50
12	0.81	0.86	0.83	50
13	0.78	1.00	0.88	50
14	0.89	0.96	0.92	50
15	1.00	1.00	1.00	50
16	1.00	0.86	0.92	50
17	0.89	1.00	0.94	50
18	0.96	0.96	0.96	50
19	0.94	0.94	0.94	50
20	0.88	0.98	0.92	50
accuracy			0.93	1050
macro avg	0.93	0.93	0.92	1050
weighted avg	0.93	0.93	0.92	1050

Table 8. Classification report of CNN

	precision	recall	f1-score	support
agricultural	0.90	0.94	0.92	50
airplane	0.81	0.84	0.82	50
baseballdiamond	0.67	0.82	0.74	50
beach	0.76	0.84	0.80	50
buildings	0.60	0.62	0.61	50
chaparral	0.94	1.00	0.97	50
denseresidential	0.53	0.42	0.47	50
forest	0.92	0.98	0.95	50
freeway	0.83	0.58	0.68	50
golfcourse	0.64	0.58	0.61	50
harbor	0.94	0.98	0.96	50
intersection	0.71	0.72	0.71	50
mediumresidential	0.46	0.52	0.49	50
mobilehomepark	0.60	0.86	0.70	50
overpass	0.79	0.82	0.80	50
parkinglot	0.98	1.00	0.99	50
river	0.91	0.82	0.86	50
runway	0.86	0.88	0.87	50
sparseresidential	0.77	0.74	0.76	50
storagetanks	0.51	0.48	0.49	50
tenniscourt	0.81	0.44	0.57	50
accuracy			0.76	1050
macro avg	0.76	0.76	0.75	1050
weighted avg	0.76	0.76	0.75	1050

In addition to precision, recall, f1-score and graphs, we also deployed confusion matrix to evaluate individual class performances in each deep learning model. Most of the classes seem to be classified accurately as true-positive with few exceptions in each model.

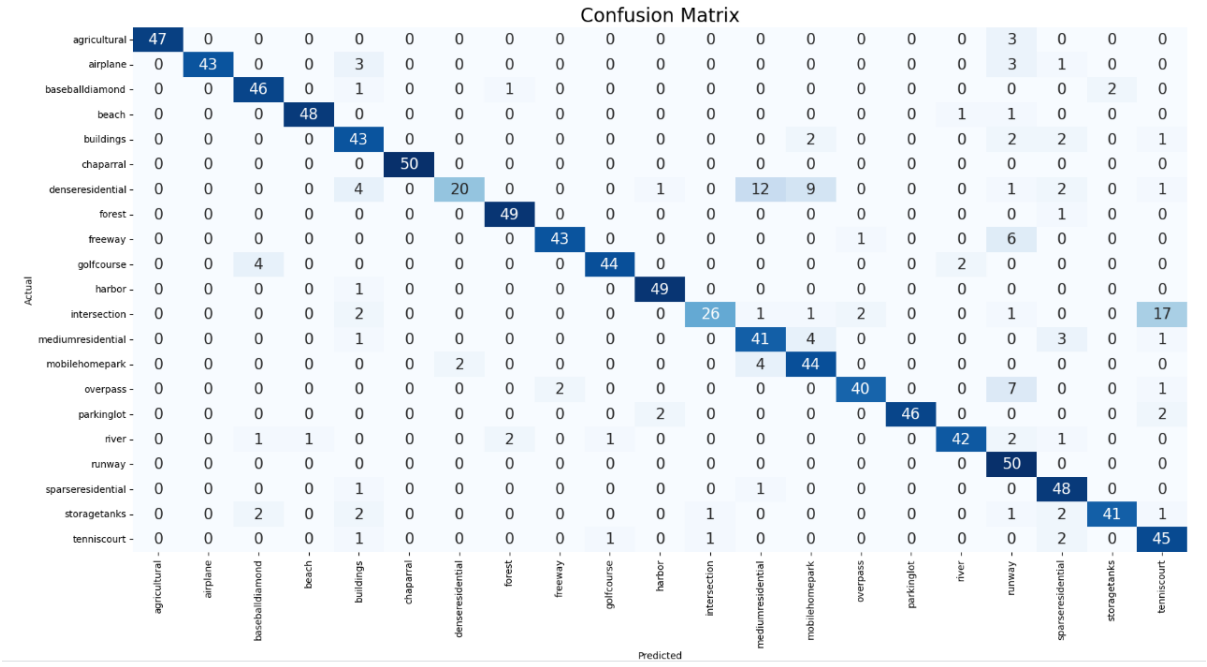


Figure 7. Confusion matrix of Inceptionv3

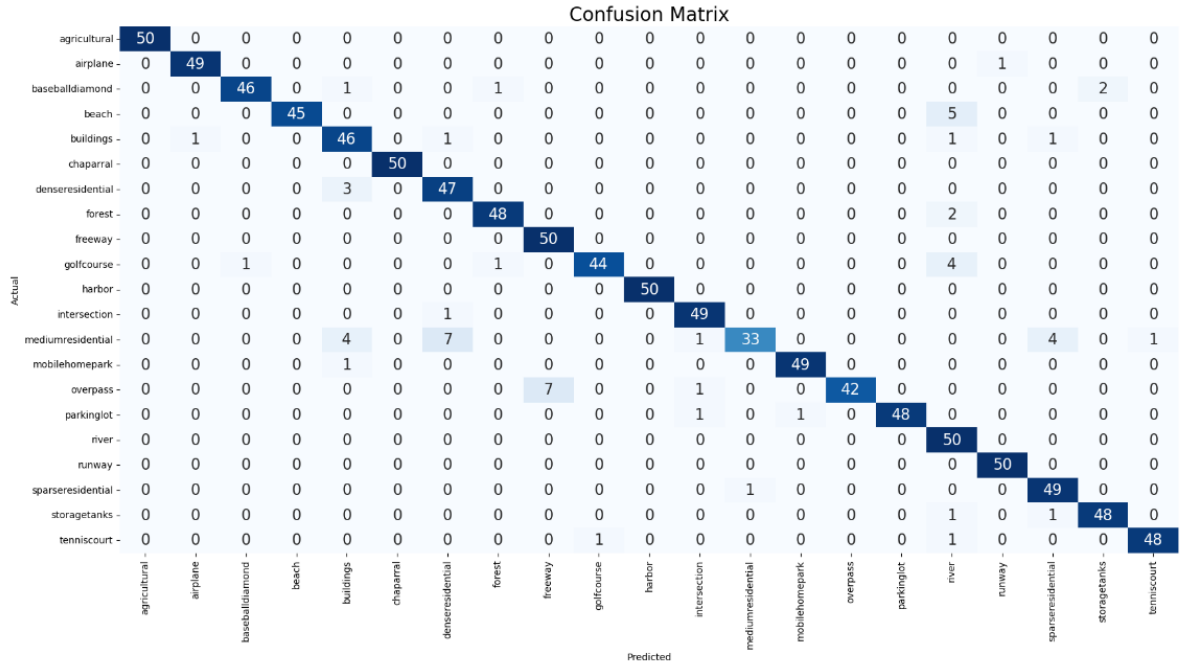


Figure 8. Confusion matrix of MobileNet

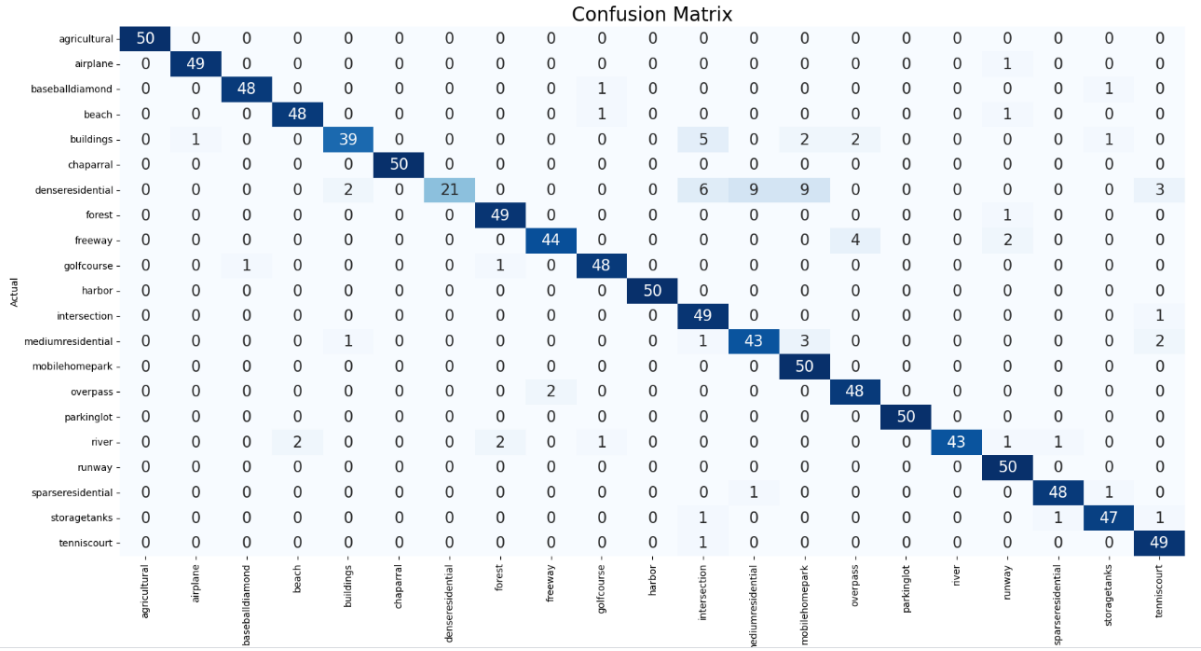


Figure 9. Confusion matrix of DenseNet201

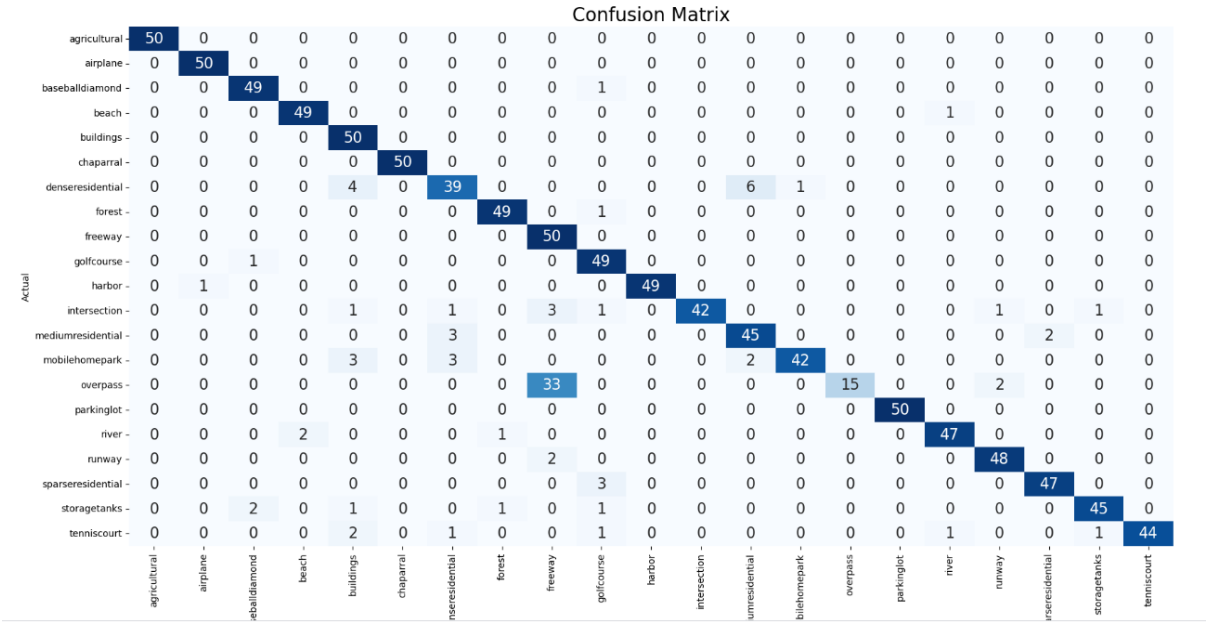


Figure 10. Confusion matrix of ResNet152V2

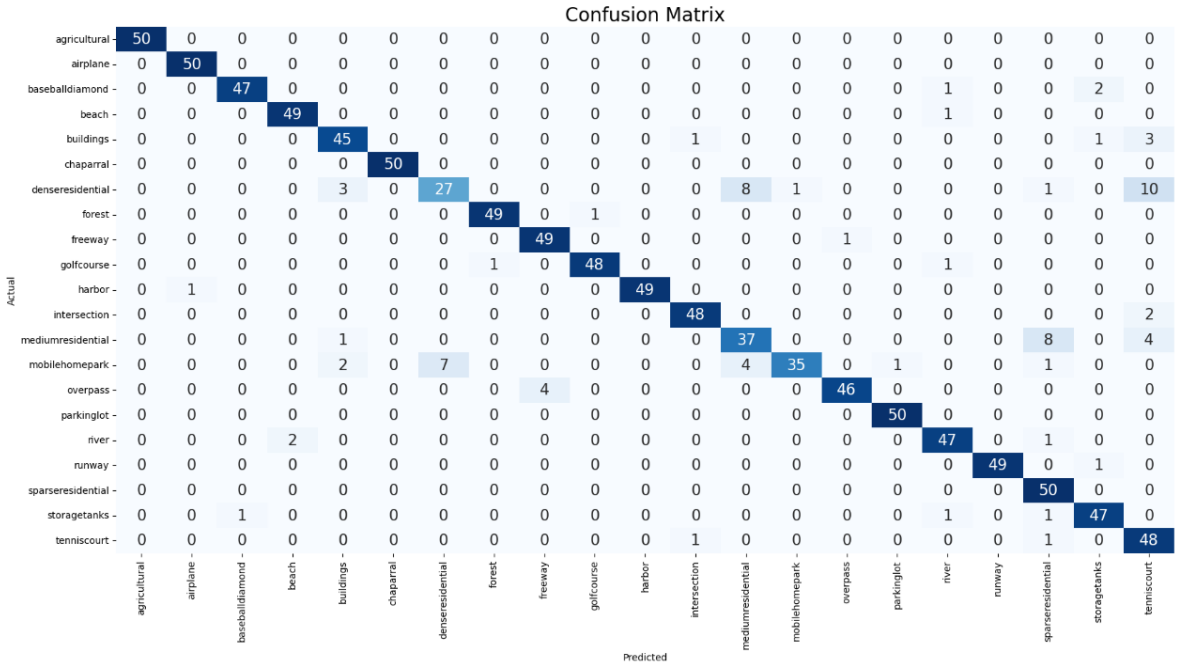


Figure 11. Confusion matrix of EfficientNetB7

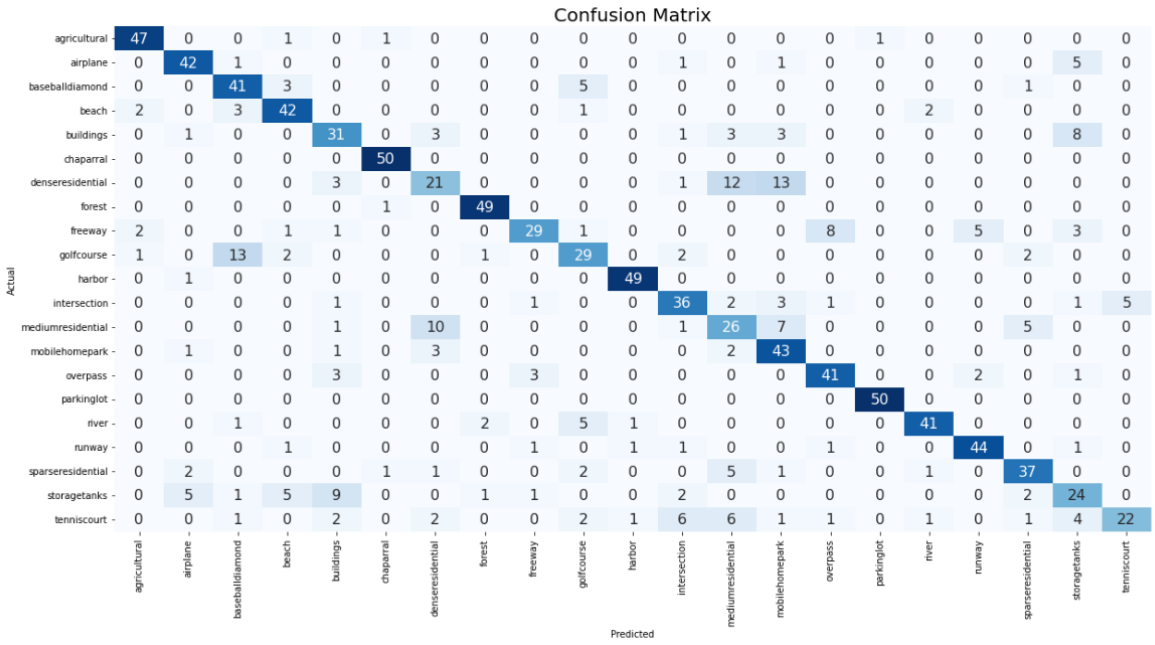


Figure 12. Confusion matrix of CNN

DL Model	Precision	Recall	F1-score	Accuracy
CNN	0.76	0.76	0.75	77.90
	Pre-trained architectures			
Inceptionv3	0.89	0.87	0.87	86.33
DenseNet201	0.93	0.93	0.92	92.24
EfficientNetB7	0.94	0.93	0.93	92.95
MobileNet	0.94	0.93	0.93	93.38
ResNet152V2	0.95	0.95	0.95	95.52

Table 1. Performance evaluation of all models using performance metrics like precision, recall, f1-score and accuracy

The overall accuracies of each deep learning model have been summarized in Table 1. The ResNet152V2 model has outperformed all the other models with an accuracy of 95.52%. We also observed that transfer learning with the pre-trained architectures gave better results than the conventional CNN model.

V. CONCLUSION

In this paper, we have compared the CNN model trained from scratch, transfer learning and fine tuning applied to different pre-trained architectures for LCLU classification problems using remotely sensed images. The models were trained on UCM dataset and their performances were evaluated using metrics such as accuracy, precision, recall, f1-score, and confusion matrix. We also noted that the pre-trained architectures gave better results than the CNN model. Among all the pre-trained architectures, the ResNet152V2 architecture has outperformed with an accuracy of 95.52% and it has also been backed by a precision, recall, and f1-score equal to 0.95.