DEEP LEARNING BASED BUILT-UP EXTRACTION USING MULTISPECTRAL SATELLITE IMAGERY

A

Project Report

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

Student Name:	**Roll No:**
ADARSH MAINDOLA	180204
ARISHTHA KIRTI	180211
MILAN KANDWAL	180229
MUKESH BAHUGUNA	180231

under

the guidance of

DR. RASHMI SAINI (ASSOCIATE PROFESSOR)

GOVIND BHALLABH PANT INSTITUTE OF

ENGINEERING AND TECHNOLOGY,

Pauri (UK)

CANDIDATE'S DECLARATION

I/We hereby certify that the project work entitled **"Deep learning based built-up extraction using multispectral satellite images"** in partial fulfilment of the requirements for the award of the Degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING and submitted to the Department of Computer Science & Engineering, Pauri, Uttarakhand. is an authentic record of our work carried out during a period from **February 2022** to **June 2022** under the supervision of **Dr Rashmi Saini, Associate Professor, Department of Computer Science and Engineering.**

The matter presented in this project has not been submitted by us for the award of any other degree of this or any other University.

Student Name:	**Roll No:**
ADARSH MAINDOLA	180204
ARISHTHA KIRTI	180211
MILAN KANDWAL	180229
MUKESH BAHUGUNA	180231

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Dr Rashmi Saini
Project Guide

Mr. Vivek Kumar Tamta
Project Coordinator
Computer Science & Engineering,
G.B Pant Institute of Engineering & Technology,
Pauri (Uttarakhand).

Prof Dr H.S Bhadauria
Head of Department
Computer Science &Engineering,
G.B Pant Institute of Engineering & Technology,
Pauri (Uttarakhand).

ACKNOWLEDGEMENT

We wish to express our deep gratitude to our guide **Dr Rashmi Saini,** for all advice, encouragement, and constant support she has given us throughout our project work. This work would not have been possible without her support and valuable suggestions.

We sincerely thank our respected Program Coordinator of the Department, **Mr Vivek Kumar Tamta,** Assistant Professor Department of Computer Science & Engineering for his great support in Coordinating project Schedules, Resource equipment and information. Liaising with group members to Define project requirements, scope, and objective in doing our project.

We are also grateful to **Prof Dr H.S Bhadauria,** Head of Computer Science and Engineering for giving us the necessary facilities to carry out our project work successfully.

We would like to thank all our **friends** for their help and constructive criticism during our project work. Finally, we have no words to express our sincere gratitude to our **parents** who have shown us this world and for every support they have given us.

Student Name:	**Roll No:**
ADARSH MAINDOLA	180204
ARISHTHA KIRTI	180211
MILAN KANDWAL	180229
MUKESH BAHUGUNA	180231

ABSTRACT

Automatic extraction of buildings has found its applications in various areas like land use land cover mapping, change detection, urban planning, disaster management and many other socio-economic activities. Remote sensing and GIS techniques play a major role in such applications. However, application areas of remote sensing have been considerably increased with availability of sub-metre resolution data from high-resolution earth satellites, such as, Sentinel 2, IKONOS, World View, and QUICKBIRD.

High Resolution Satellite Imagery facilitates to identify various urban related features, such as, road, tree, building and other natural and man-made structures, present on the earth's surface. Since, manual extraction of buildings using satellite imagery is time consuming and costly affair, hence, automated methods for building extraction have been emerging as a time and cost-effective solution with minimal user involvement.

The objective of this study was to develop a land use/land cover classifier using machine learning/deep learning techniques to extract features of multispectral satellite images such as buildings, vegetation, water bodies etc. The satellite images were acquired from USGS Earth explorer website. After pre-processing of downloaded data, images were classified into 5 land use classes namely Farmland, Water bodies, Buildings, Vegetation and Highway, to form labelled dataset for model training. Further accuracies of different deep learning models used in the study were compared for different multispectral data.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1. Introduction

1.1. Satellite Imagery

Remotely sensed satellite imagery is becoming increasingly common as satellites equipped with technologically advanced sensors are continually being sent into space by public agencies and private companies around the globe. Satellites are used for applications such as military and civilian earth observation, communication, navigation, weather, research, and more. Currently, more than 3,000 satellites have been sent to space, with over 2,500 of them originating from Russia and the United States. These satellites maintain different altitudes, inclinations, eccentricities, synchronies, and orbital centres, allowing them to image a wide variety of surface features and processes Satellite imagery is a critical tool for visualizing ground conditions. Whether you need a foundational map for an app or a comprehensive dataset for business intelligence. There are four types of resolution that characterize any particular remote sensor

The spatial resolution of a satellite image, is a direct representation of the ground coverage for each pixel shown in the image. Spatial resolution is determined by the sensors' instantaneous field of view (IFOV). The IFOV is essentially the ground area through which the sensor is receiving the electromagnetic radiation signal and is determined by height and angle of the imaging platform.

Spectral resolution denotes the ability of the sensor to resolve wavelength intervals, also called bands, within the electromagnetic spectrum. The spectral resolution is determined by the interval size of the wavelengths and the number of intervals being scanned.

Temporal resolution is the amount of time between each image collection period and is determined by the repeat cycle of the satellite's orbit. Temporal resolution can be thought of as true-nadir or off-nadir. Areas considered true-nadir are those located directly beneath the sensor while off-nadir areas are those that are imaged obliquely

The fourth and final type of resolution, radiometric resolution, refers to the sensitivity of the sensor to variations in brightness and specifically denotes the number of grayscale levels that can be imaged by the sensor.

1.2. Satellite Imagery with Machine Learning and Deep Learning

In terms of raw data, the earth observation industry is undeniably exploding. Investments in freely available data from satellite constellations like MODIS, Landsat, and Sentinel have democratized access to timely satellite imagery of the entire globe (albeit at a lower resolution than you're accustomed to seeing on Google Maps). Meanwhile, cloud providers like AWS and Google Cloud have gone so far as to store satellite data for free, further accelerating global usage of these images.

The trouble, naturally, is that interpreting the content of satellite imagery is not an easy task. In the field of remote sensing, researchers have been applying algorithmic techniques to the challenge of earth imagery interpretation for over 70 years. Until relatively recently, even simple tasks like identifying building footprints or distinguishing tree canopy in urban areas were laborious sub-specialties of the field. Then, in 2012, the "deep learning revolution," opened up an entirely novel frontier of useful algorithms that could be applied to satellite imagery with state-of-the-art results.

Chapter 2. Literature Review

Author	Description	Technology Used	Remarks
M.P. Vaishnnave, etc.	A Study on Deep Learning Models for Satellite	1. AlexNet 2. GoogleNet 3. VggNet 4. VggNet-16	By analysing the existing literature for deep learning, this paper discusses some literature gaps to enable the research community to develop new data-driven algorithms.
Sachin Padmanabhan	Convolutional Neural Networks for Image Classification and Captioning	. AlexNet . VggNet	VggNet model gave better accuracy compared to AlexNet.
Darius Phiri, etc.	Sentinel-2 Data for Land Cover/Use Mapping	1. VggNet-16 2. VggNet-19	The study has shown that due to the high spatial resolution, Sentinel-2 data can achieve high accuracies compared to other medium spatial resolution satellite images, such as Landsat.
Abhishek Verma	Compressed residual-VGG16 CNN model for big data places image recognition	1. VggNet-16 2. Residual Squeeze VggNet-16	In comparison to VGG16 the proposed model is 88.4% smaller in size and 23.86% faster in the training time.
Arun Kumar Dubey	Automatic facial recognition using VGG16 based transfer learning model	1. VggNet-16	The proposed model has shown 94.8% accuracy on CK+ and 93.7% on the JAFFE dataset and found superior to existing techniques.

Sheldon Mascarenhas, etc.	A comparison between VGG16, VGG19 and ResNet50 architecture frameworks for Image Classification	1. VggNet-16 2. VggNet-19 3. ResNet50	On comparing the accuracies of the three models for the same image classification problem ResNet50 gives the highest accuracy and thus has the best architecture.
Luka Rumora, etc.	Impact of Various Atmospheric Corrections on Sentinel-2 Land Cover Classification Accuracy Using Machine Learning Classifiers	1. SVM classifier 2. RF algorithm 3. CB algorithm	SVM classification method outperformed all other methods with radiometric indices included, but also without included radiometric indices.
Anuvi Rawat, etc.	Deep learning-based models for temporal satellite data processing: Classification of paddy transplanted fields	1. 1-D CNN 2. CNN-LSTM	This study indicates that 1-D CNN based deep learning models provide an effective solution to handle mono/bi-sensor temporal remote sensing data of medium spatial resolution with small size training datasets.
Zichao Jiang	A novel crop weed recognition method based on transfer learning from VGG16 implemented by Keras	1. VggNet-16	The accuracy rate on the training set is 98.99%, and the accuracy on the verification set is 91.08%.
Ishrat Zahan Mukti, etc.	Transfer Learning Based Plant Diseases Detection Using ResNet50	1. AlexNet 2. VggNet-16 3. VggNet-19 4. ResNet50	ResNet50 has given the best performance of 99.80 % training accuracy.

Chapter 3. Machine Learning and Deep Learning Techniques

Deep learning is a subset of machine learning which is based on learning through hierarchy of representation, which correspond to levels of features, factors or concepts, where higher level features are derived from lower-level features and same lower-level features can help to derive many higher-level features.

Deep learning is based on neural networks, a type of data structure loosely based on the structure of a biological neuron. Neural networks have been around for decades, but have caught much attention in recent years due to the advancements in the algorithms and computational power of our systems. One definition for an artificial neural network was provided first in [1], where the author stated that "*a neural network is a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs*".

Deep learning has proven to be an incredibly versatile technology–everything from apps on your phone that make you look younger, to the Alexa voice assistant, to Tesla's Autopilot feature. Deep learning finds its extensive use in analysis of satellite imagery.

3.1. Training via Convolutional Neural Network

A convolutional neural network is an artificial neural network most commonly employed for visual imagery. A convolutional neural network consists of an input layer, hidden layers and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution. In a convolutional neural network, the hidden layers include layers that perform convolutions. Typically, this includes a layer that performs a dot product of the convolution kernel with the layer's input matrix.

[1] https://www.digitalglobe.com/resources/satellite-information

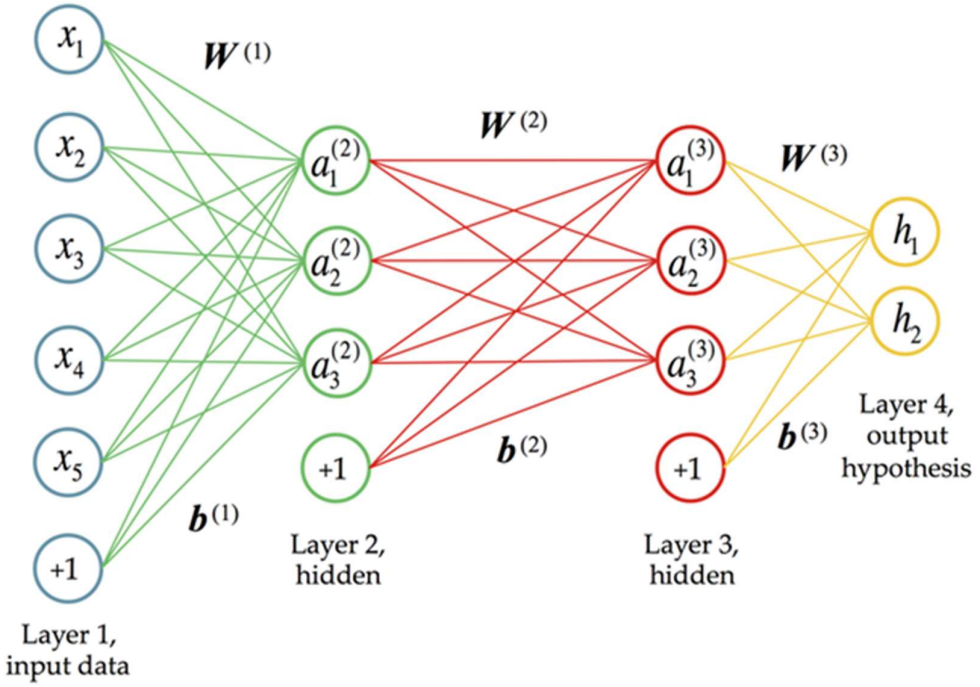


Figure 1: Deep learning architecture

A neural network fundamentally consists of learnable parameters like in a single linear classifier - termed weights and biases. Consider a simple linear function of the form shown in Equation 3.1. An input tensor (the data input, a flattened image vector [2] when dealing with images) is multiplied with an appropriately sized variable weight matrix, and added with a bias vector. If x_i denotes the input vector and W_i denotes the corresponding weight matrix, b_i its bias vector, the linear function returns a score, Yi as:

$$Y_i = W_i x_i + b_i \qquad (3.1)$$

This function essentially mapped the data x_i from image space into a score, Yi for that particular image. In an image classification task, y_i can be understood to encode the classifier's confidence that the image x_i belongs to a particular label. For image segmentation, on the other hand, the problem is formulated as a per-pixel classification. Hence, yi would represent the score given to a particular pixel in the image, xi as belonging to a particular class.

[2] Images can be represented as a matrix of numbers, specifying real values across different channels/bands. For an RGB image of size a × b, this would mean a matrix of size a × b × c.

3.2. VGG 16 model

VGG16 is a simple and widely used Convolutional Neural Network (CNN) Architecture used for ImageNet, a large visual database project used in visual object recognition software research.

The VGG16 model achieved 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes.

VGG16 is used in many deep learning image classification techniques and is popular due to its ease of implementation. VGG16 is extensively used in learning applications due to the advantage that it has.

The VGG16 architecture is depicted in figure, shown below:

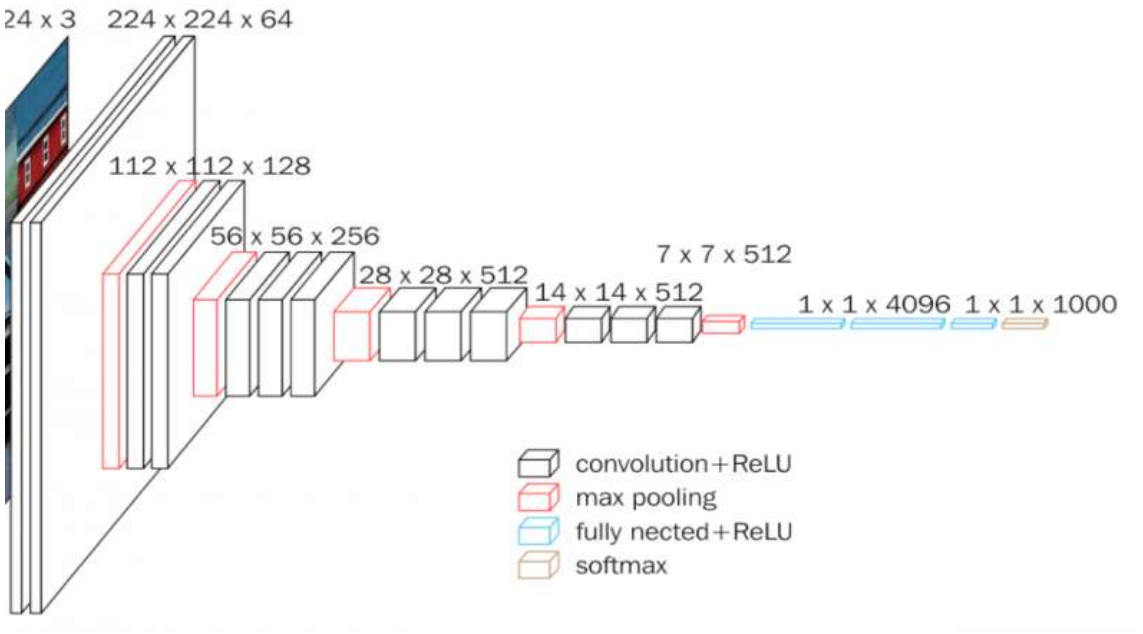


Figure 2: VGG-16 model architecture

The input to cov1 layer is of fixed size 224 x 224 RGB image. The image is passed through a stack of convolutional (conv.) layers, where the filters were used with a very small receptive field: 3×3 (which is the smallest size to capture the notion of left/right, up/down, centre). In one of the configurations, it also utilizes 1×1 convolution filters, which can be seen as a linear transformation of the input channels (followed by non-linearity). The convolution stride is

fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e., the padding is 1-pixel for 3×3 conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2×2-pixel window, with stride 2.

3.3. DenseNet

A convolutional neural network can be called a DenseNet if it utilises dense connections between the layers through Dense Blocks, where we connect all layers (with matching feature map sizes) directly with each other. DenseNet have several compelling advantages: they alleviate the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse, and substantially reduce the number of parameters.

3.4. Optimizers

During the training process, we tweak and change the parameters (weights) of our model to try and minimize that loss function, and make our predictions as correct and optimized as possible.

Optimizers tie together the loss function and model parameters by updating the model in response to the output of the loss function. In simpler terms, optimizers shape and Mold your model into its most accurate possible form by futzing with the weights. The loss function is the guide to the terrain, telling the optimizer when it's moving in the right or wrong direction.

3.5. Gradient Decent Algorithms

This algorithm is used across all types of machine learning to optimize. It's fast, robust, and flexible.

1. Calculate what a small change in each individual weight would do to the loss function
2. Adjust each individual weight based on its gradient
3. Keep doing steps #1 and #2 until the loss function gets as low as possible

The tricky part of this algorithm (and optimizers in general) is understanding gradients, which represent what a small change in a weight or parameter would do to the loss function. Gradients are partial derivatives, and are a measure of change.

3.6. The learning rate

Changing our weights too fast by adding or subtracting too much can hinder our ability to minimize the loss function. We don't want to make a jump so large that we skip over the optimal value for a given weight.

To make sure that this doesn't happen, we use a variable called "the learning rate." This thing is just a very small number, usually something like 0.001 that we multiply the gradients by to scale them. This ensures that any changes we make to our weights are pretty small.

3.7. Regularization

The term 'regularization' refers to a set of techniques that regularizes learning from particular features for traditional algorithms or neurons in the case of neural network algorithms. It normalizes and moderate's weights attached to a feature or a neuron so that algorithms do not rely on just a few features or neurons to predict the result. This technique helps to avoid the problem of overfitting.

To understand regularization, let's consider a simple case of linear regression. Mathematically, linear regression is stated as below:

$$y = w_0 + w_1x_1 + w_2x_2 + \ldots\ldots + w_nx_n$$

where y is the value to be predicted;

$x_1, x_2, \ldots, x_n$ *are features that decides the value of y;*

w_0 is the bias;

$w_1, w_2, \ldots, w_n$ are the weights attached to $x_1, x_2, \ldots, x_n$ relatively.

Now to build a model that accurately predicts the *y* value, we need to optimize above mentioned bias and weights. To do so, we need to use a loss function and find optimized parameters using gradient descent algorithms and its variants.

There are three main regularization techniques, namely:

1. Ridge Regression (L2 Norm)
2. Lasso (L1 Norm)
3. Dropout

3.8. Ridge Regression (L2 Regularization)

Ridge regression is also called L2 norm or regularization. When using this technique, we add the sum of weight's square to a loss function and thus create a new loss function which is denoted thus:

$$\text{Loss} = \sum_{j=1}^{m} \left(Yi - Wo - \sum_{i=1}^{n} Wi\, Xji \right)^2 + \lambda \sum_{i=1}^{n} Wi^2$$

As seen above, the original loss function is modified by adding normalized weights. Here normalized weights are in the form of squares. You may have noticed parameters λ along with normalized weights. λ is the parameter that needs to be tuned using a cross-validation dataset. When you use λ=0, it returns the residual sum of square as loss function which you chose initially. For a very high value of λ, loss will ignore core loss function and minimize weight's square and will end up taking the parameters' value as zero. Now the parameters are learned

using a modified loss function. To minimize the above function, parameters need to be as small as possible. Thus, L2 norm prevents weights from rising too high.

3.9. Lasso Regression (L1 Regularization)

Also called lasso regression and denoted as below:

$$\text{Loss} = \sum_{j=1}^{m}\left(Yi - Wo - \sum_{i=1}^{n} Wi\,Xji\right)^2 + \lambda \sum_{i=1}^{n} |Wi|$$

This technique is different from ridge regression as it uses absolute weight values for normalization. λ is again a tuning parameter and behaves in the same as it does when using ridge regression. As loss function only considers absolute weights, optimization algorithms penalize higher weight values.

In ridge regression, loss function along with the optimization algorithm brings parameters near to zero but not actually zero, while lasso eliminates less important features and sets respective weight values to zero. Thus, lasso also performs feature selection along with regularization.

3.10. Dropout

Dropout is a regularization technique used in neural networks. It prevents complex co-adaptations from other neurons. In neural nets, fully connected layers are more prone to overfit on training data. Using dropout, you can drop connections with *1-p* probability for each of the specified layers. Where *p* is called **keep probability parameter** and which needs to be tuned.

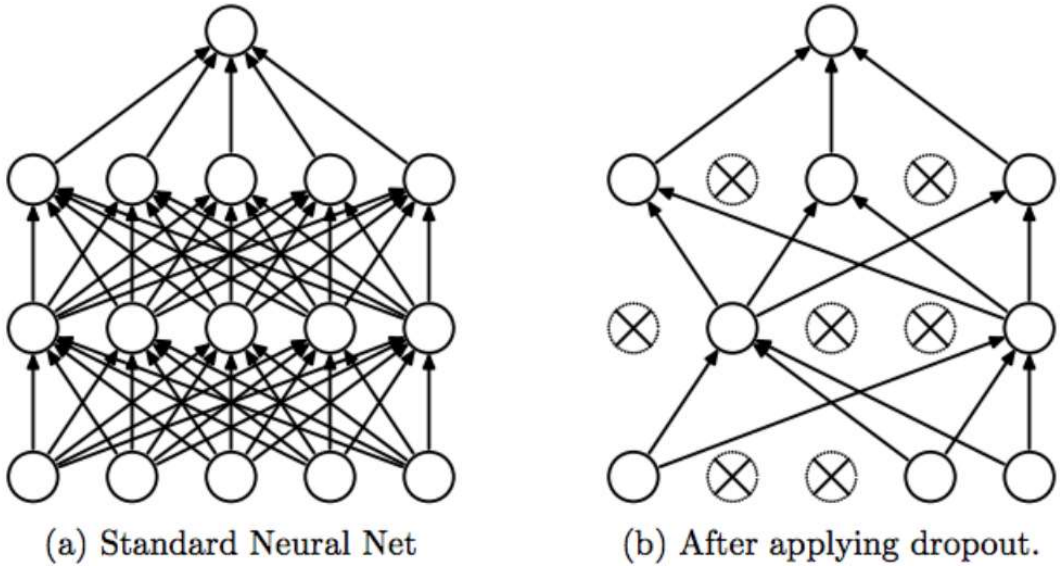


Figure 3: Neural network with and without dropout

With dropout, you are left with a reduced network as dropped out neurons are left out during that training iteration. Dropout decreases overfitting by avoiding training all the neurons on the complete training data in one go. It also improves training speed and learns more robust internal functions that generalize better on unseen data. However, it is important to note that Dropout takes more epochs to train compared to training without Dropout (If you have 10000 observations in your training data, then using 10000 examples for training is considered as 1 epoch). Along with Dropout, neural networks can be regularized also using L1 and L2 norms. Apart from that, if you are working on an image dataset, image augmentation can also be used as a regularization method. For real-world applications, it is a must that a model performs well on unseen data. The techniques we discussed can help you make your model learn rather than just memorize.

3.11. Confusion matrix

The confusion matrix is a matrix used to determine the performance of the classification models for a given set of test data. It can only be determined if the true values for test data are known. The matrix itself can be easily understood, but the related terminologies may be confusing.

Since it shows the errors in the model performance in the form of a matrix, hence also known as an error matrix. Some features of Confusion matrix are given below:

- For the 2 prediction classes of classifiers, the matrix is of 2*2 table, for 3 classes, it is 3*3 table, and so on.
- The matrix is divided into two dimensions that are predicted values and actual values along with the total number of predictions.
- Predicted values are those values, which are predicted by the model, and actual values are the true values for the given observations.
- It looks like the below table:

n = total predictions	Actual: No	Actual: Yes
Predicted: No	True Negative	False Positive
Predicted: Yes	False Negative	True Positive

The above table has the following cases:

- True Negative: Model has given prediction No, and the real or actual value was also No.
- True Positive: The model has predicted yes, and the actual value was also true.
- False Negative: The model has predicted no, but the actual value was Yes, it is also called as Type-II error.
- False Positive: The model has predicted Yes, but the actual value was No. It is also called a Type-I error.

3.11.1 Need for Confusion Matrix in Machine learning

- It evaluates the performance of the classification models, when they make predictions on test data, and tells how good our classification model is.
- It not only tells the error made by the classifiers but also the type of errors such as it is either type-I or type-II error.

- With the help of the confusion matrix, we can calculate the different parameters for the model, such as accuracy, precision, etc.

3.12. Data Generators

A test data generator can be used to create either random data or structured and formatted data. Structured data is generally more useful for databases because these systems often save data in tables and columns that contain specific types of information; random data is not suited for this purpose.

Test data generators follow a specific procedure:

1. Program control flow graph construction
2. Path selection
3. Generation of test data

Once the path for the testing has been determined, the test data generator generates data that results in the execution of the selected path, with the aim of creating data meant to traverse the path chosen by the path selector. This is done through mathematical modelling.

There are different kinds of test data generators:

- Random test data generator - This is the simplest kind, which can be used to test many programs as it can just randomly generate a bit stream and have that be represented as the required data type.
- Goal-oriented generator - This one generates input for any path specified instead of just the usual way of generating input from the entry to the exit of the code. This type can find any input for any path and has little chance of generating infeasible paths.
- Path wise test data generator - This generator is assigned a specific path to follow instead of giving it a choice among many paths. This leads to a greater path knowledge and prediction of coverage. It is similar to the goal-oriented generator.
- Intelligent test data generator - This type depends on sophisticated analysis of the code to be tested in order for it to guide the search for test data. This may generate test data more quickly but the analysis portion requires great insight in order to anticipate the different situations that may arise.

Chapter 4. Dataset and Study Area

Sentinel-2 is an Earth observation mission from the Copernicus Programme that systematically acquires optical imagery at high spatial resolution (10 m to

60 m) over land and coastal waters. The mission is currently a constellation with two satellites, Sentinel-2A and Sentinel-2B; a third satellite, Sentinel-2C, is currently undergoing testing in preparation for launch in 2024.

The mission supports a broad range of services and applications such as agricultural monitoring, emergencies management, land cover classification or water quality.

The Sentinel-2 mission has the following key characteristics:

1. Multi-spectral data with 13 bands in the visible, near infrared, and short-wave infrared part of the spectrum
2. Systematic global coverage of land surfaces from 56° S to 84° N, coastal waters, and all of the Mediterranean Sea
3. Revisiting every 10 days under the same viewing angles. At high latitudes, Sentinel-2 swaths overlap, and some regions will be observed twice or more every 10 days, but with different viewing angles.
4. Spatial resolution of 10 m, 20 m and 60 m
5. 290 km field of view
6. Free and open data policy

- There are 13 Sentinel 2 bands in total. Each band is 10, 20, or 60 meters in pixel size.
- Sentinel 2 consists of 2 satellites. First came Sentinel 2A which was launched in 2015. Next came Sentinel 2B in 2017.
- Two additional satellites (Sentinel 2C and 2D) are planned to launch in 2020 and 2021. This will make a total of four Sentinel-2 satellites.
- Overall, these 2 additional satellites will cut the revisit time in half.

  The availability of medium resolution satellite imagery (i.e., Sentinel-2A) provides the rapid, low-cost and more accurate mapping. This report presents the use of satellite imagery (Sentinel-2A) for semantic segmentation in the area of Gautam Buddha Nagar (UP, India). Sentinel-2A images for a season of 2022 are being used for this study. The season of this region is from December to February. This study also uses multi-band (i.e., 2, 3, 4, 5,8A, etc.) of the sentinel 2A image. Image treatments use "ArcGIS" and

SNAP, two open-source image processing software. The procedures include image enhancement, clipping, and classification. The classification consists of pre-processing, processing and post-processing tasks. Then, classification results evaluated by confusion-matrix. The result shows the potential use of Sentinel 2A to map and extract features in the study area

4.1. Raster / Composite File - (3 Bands):

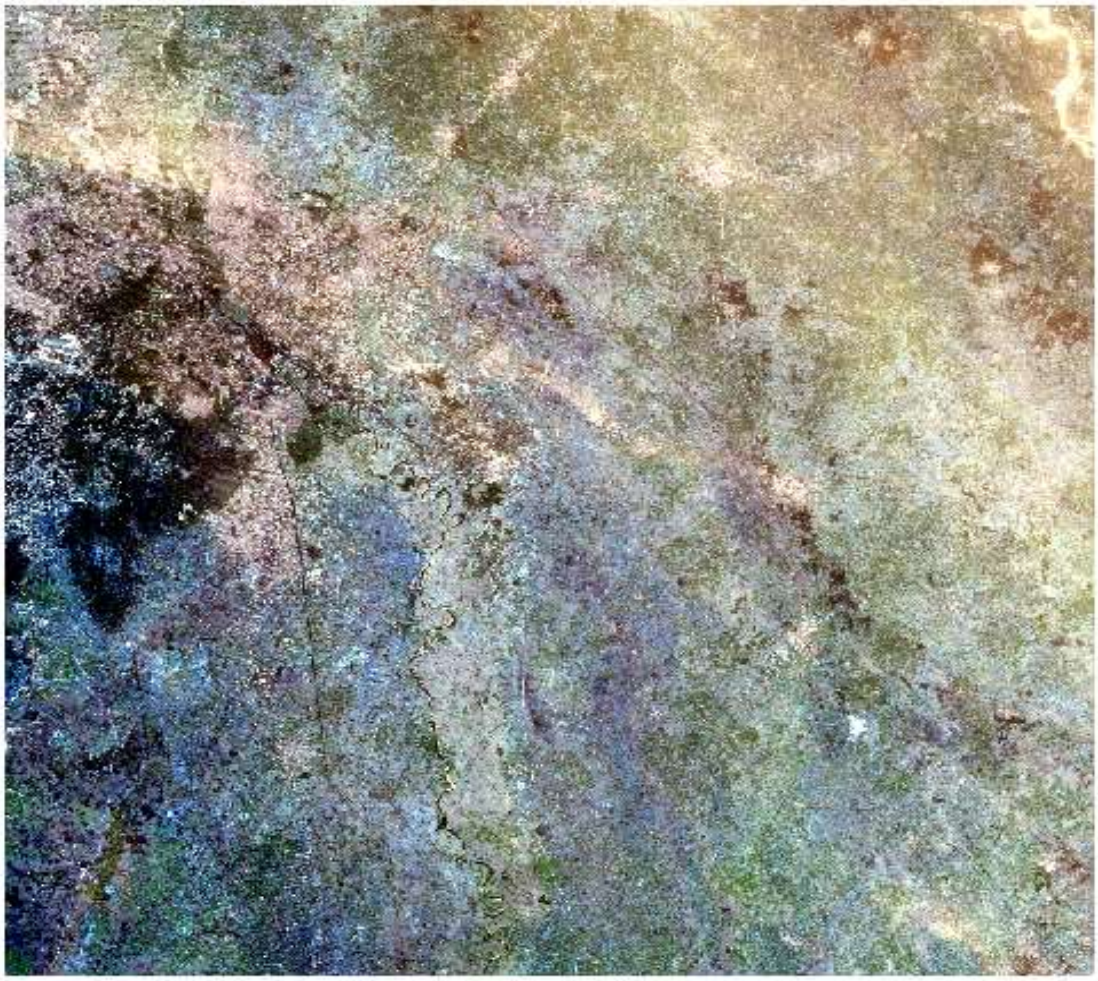


Figure 4: 3 bands raster composite file

4.2. Raster / Composite File - (10 Bands):

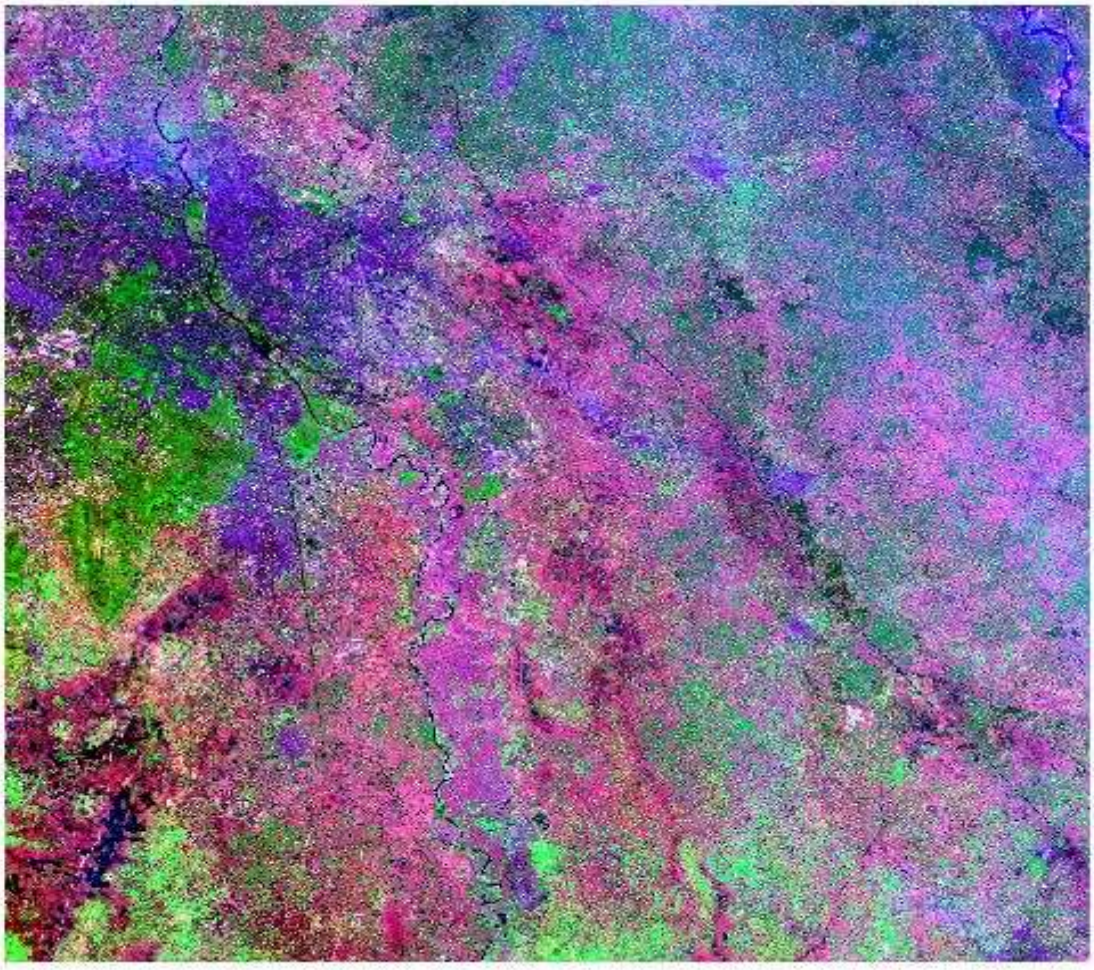


Figure 5: 10 bands raster composite file

4.3. Raster Extracted by Mask:

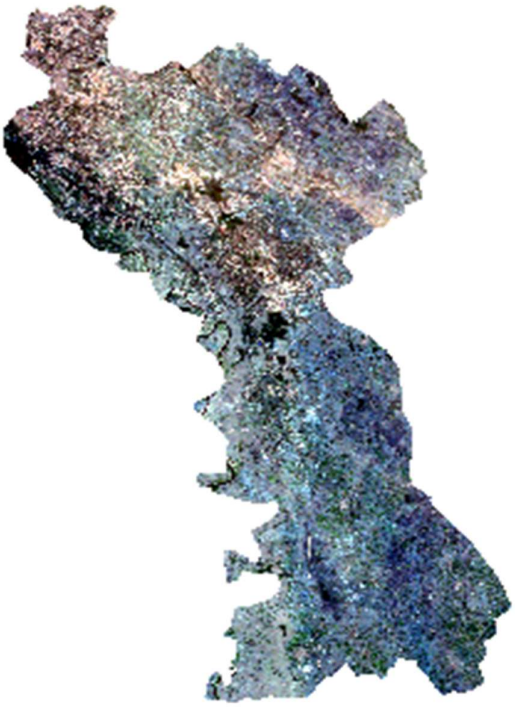


Figure 6: Extracted raster by mask

The availability of medium resolution satellite imagery (i.e., Sentinel-2A) provides the rapid, low-cost and more accurate mapping. This report presents the use of satellite imagery (Sentinel-2A) for semantic segmentation in the area of Gautam Buddh Nagar (UP, India). Sentinel-2A images for a season of 2022 are being used for this study. The season of this region is from December to February. This study also uses multi-band (i.e., 2,3,4,5,8A, etc.) of the sentinel 2A image. Image treatments use "ArcGIS" and SNAP, two open-source image processing software. The procedures include image enhancement, clipping, and classification. The classification consists of pre-processing, processing and post-processing tasks. Then, classification results evaluated by confusion-matrix. The result shows the potential use of Sentinel 2A to map and extract features in the study area.

4.4. Dataset:

We have created a RGB dataset consists of **4178** images belonging to **five** classes:

- Buildings: **1063** images
- Vegetation: **978** images
- Farmland: **965** images
- Highway: **840** images
- Water Body: **332** images

Each image has 64 x 64 resolution.

We need to split our dataset into two parts: training dataset and validation dataset. The purpose of splitting data is to avoid overfitting which is paying attention to minor details/noise which is not necessary and only optimizes the training dataset accuracy.

We have dedicated 70% of the dataset as the training data and the remaining 30% as the Validating data, which makes the split ratio as 0.7: 0.3 of train to validation set.

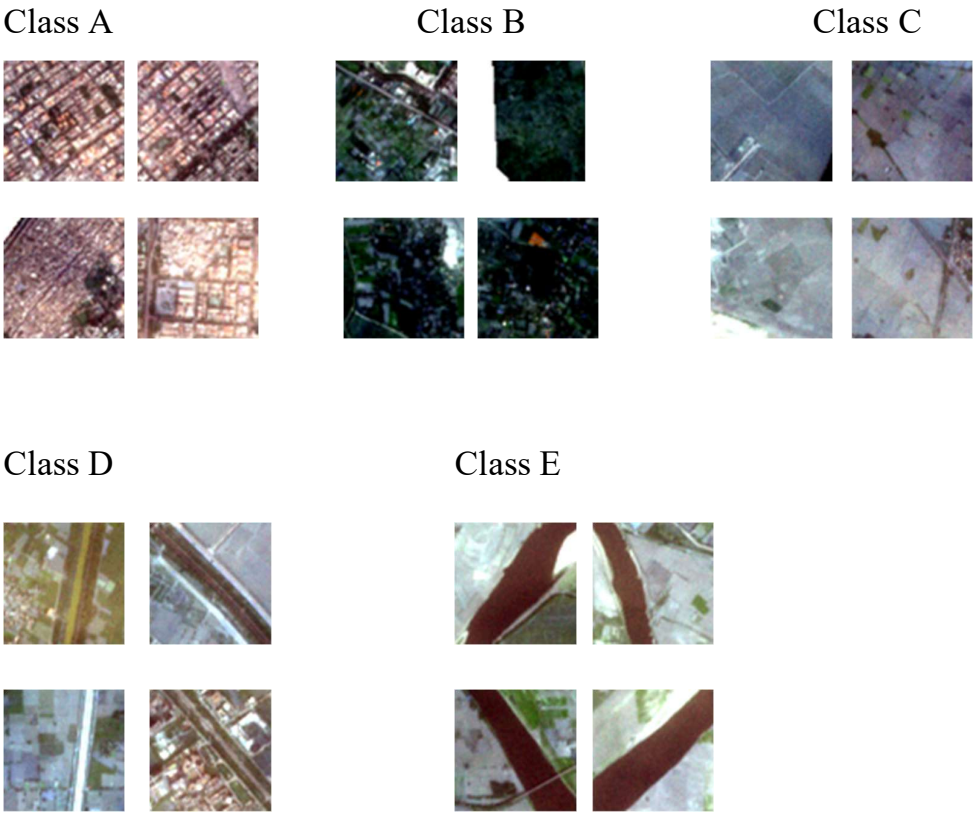


Figure 7: RGB dataset images

The Multispectral dataset we created consists of **4178** images belonging to **five** classes:

- Buildings: **1063** images
- Vegetation: **978** images
- Farmland: **965** images
- Highway: **840** images
- Water Body: **332** images

Each image has 64 x 64 resolution.

We need to split our dataset into two parts: training dataset and validation dataset. The purpose of splitting data is to avoid overfitting which is paying attention to minor details/noise which is not necessary and only optimizes the training dataset accuracy.

We have dedicated 70% of the dataset as the training data and the remaining 30% as the Validating data, which makes the split ratio as 0.7: 0.3 of train to validation set.

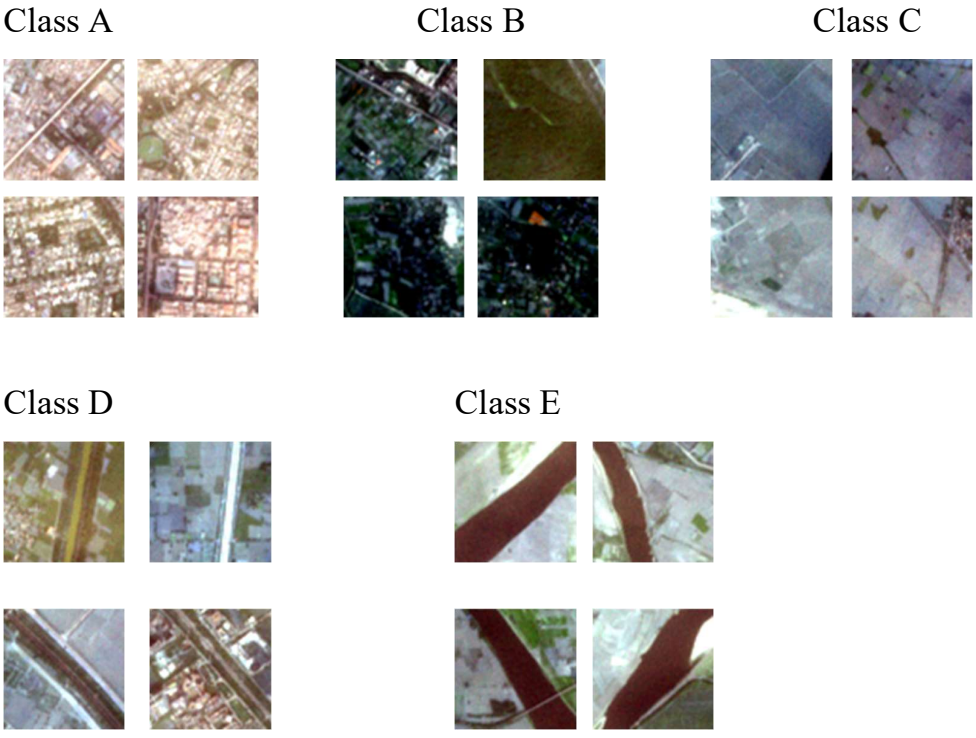


Figure 8: 10 band dataset images

Chapter 5. Methodology:

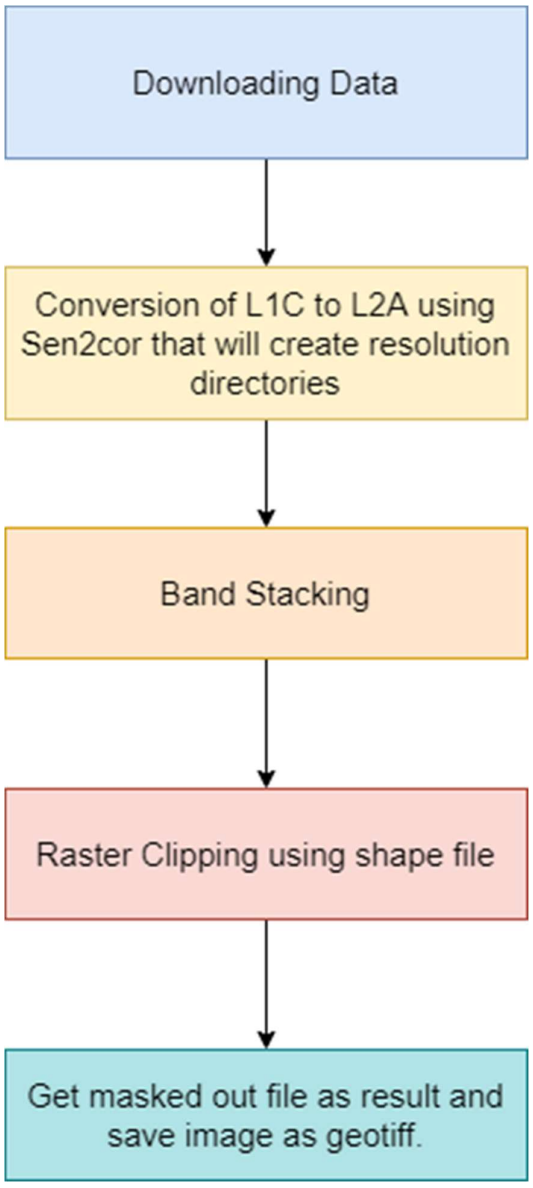


Figure 9: Flow chart for creating raster or composite file

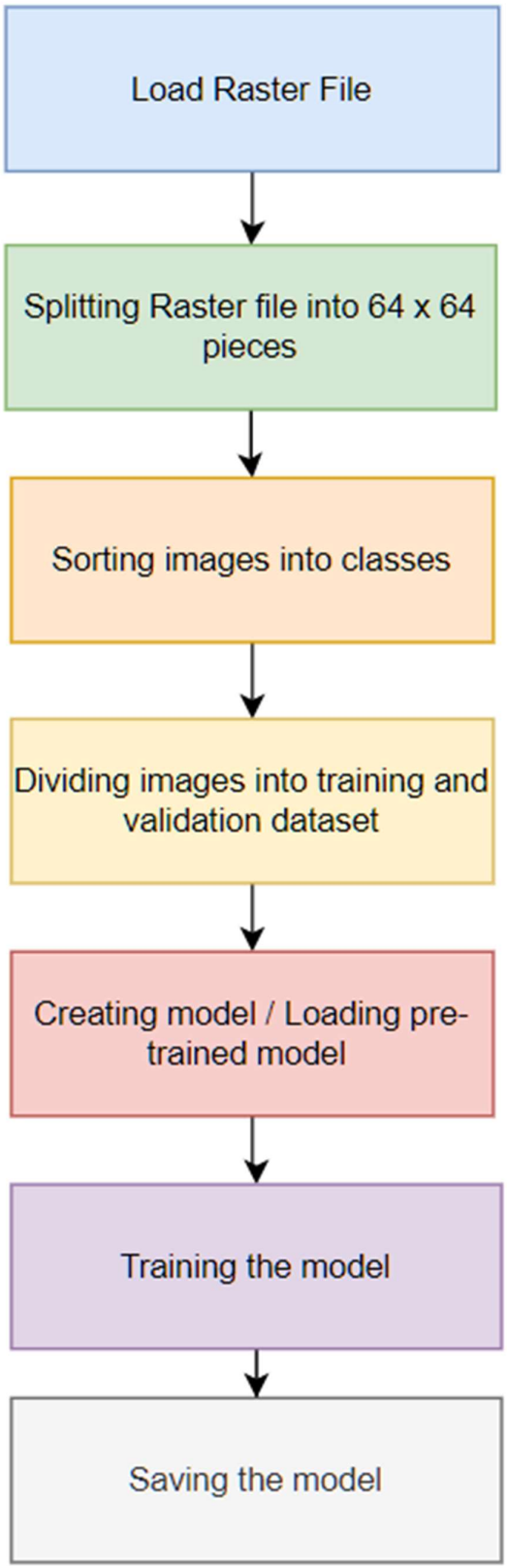


Figure 10: Flow chart for training model

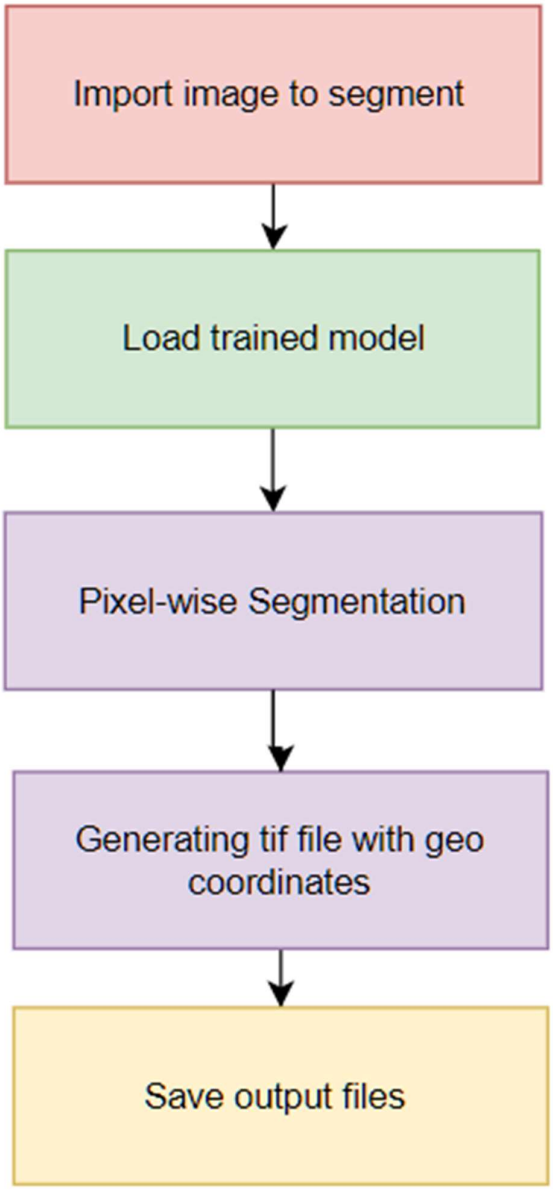


Figure 11: Flow chart for classifying images

Chapter 6. Output

6.1. Metrics for RGB Dataset

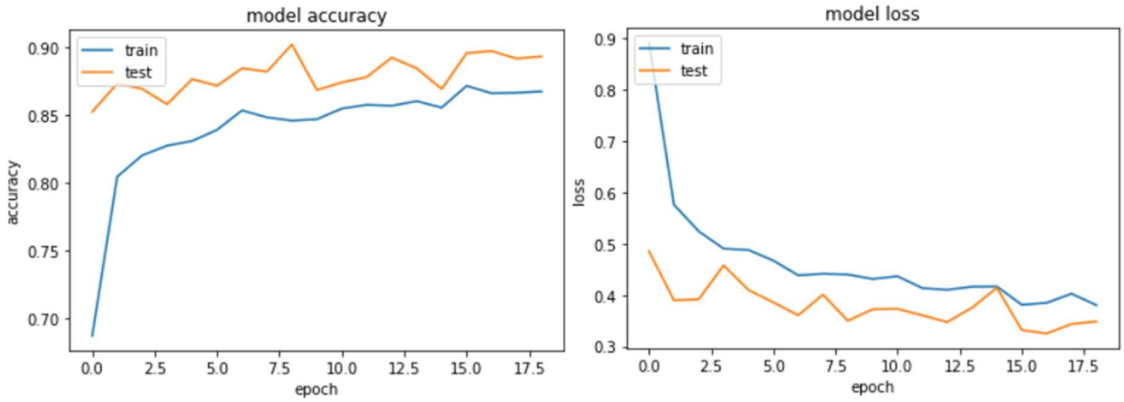
6.1.1 VGG – 16

Accuracy –

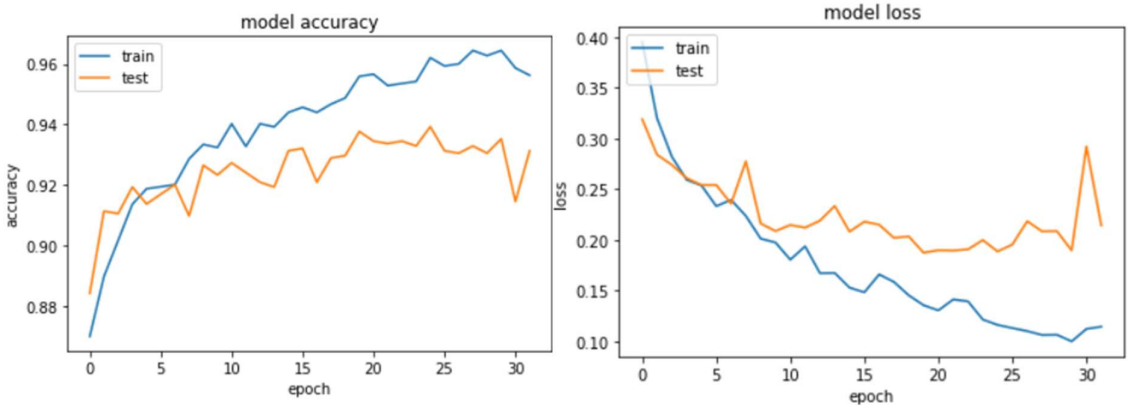
Train – 96.21

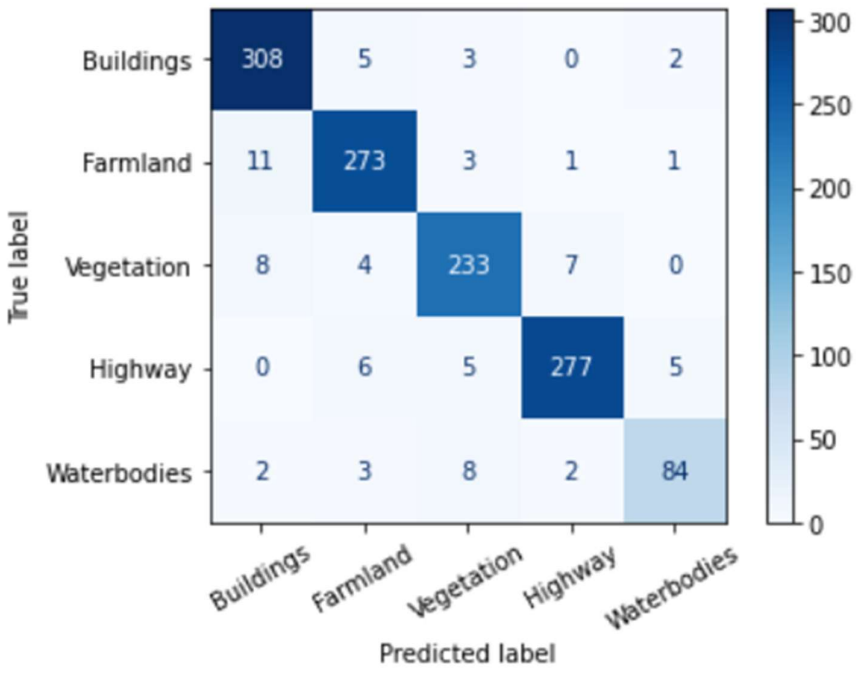
Val – 93.925

Base Model –



Fine Tuning –





	precision	recall	f1-score	support
Buildings	0.94	0.97	0.95	318
Farmland	0.94	0.94	0.94	289
Vegetation	0.92	0.92	0.92	252
Highway	0.97	0.95	0.96	293
Waterbodies	0.91	0.85	0.88	99
accuracy			0.94	1251
macro avg	0.94	0.93	0.93	1251
weighted avg	0.94	0.94	0.94	1251

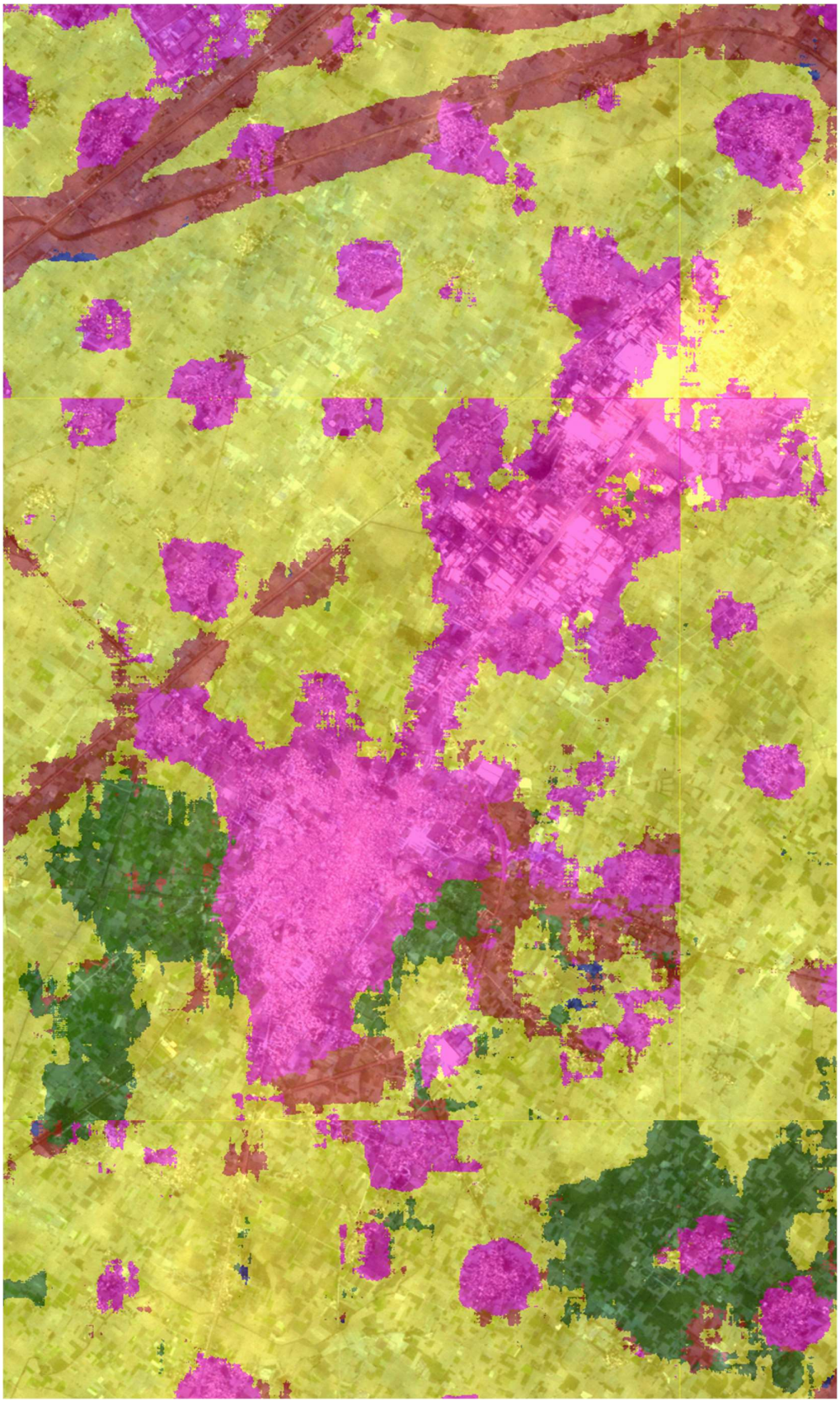


Figure 12: Processed RGB image using VGG-16

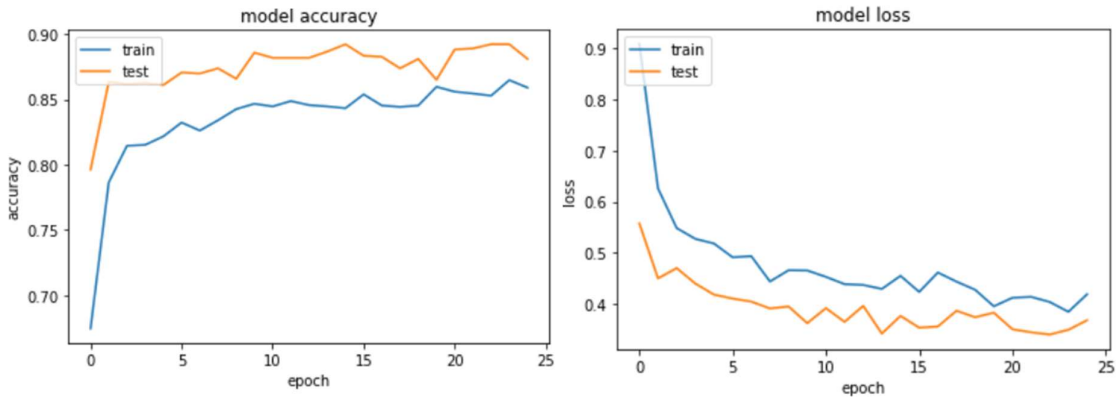
6.1.2 VGG – 19

Accuracy –

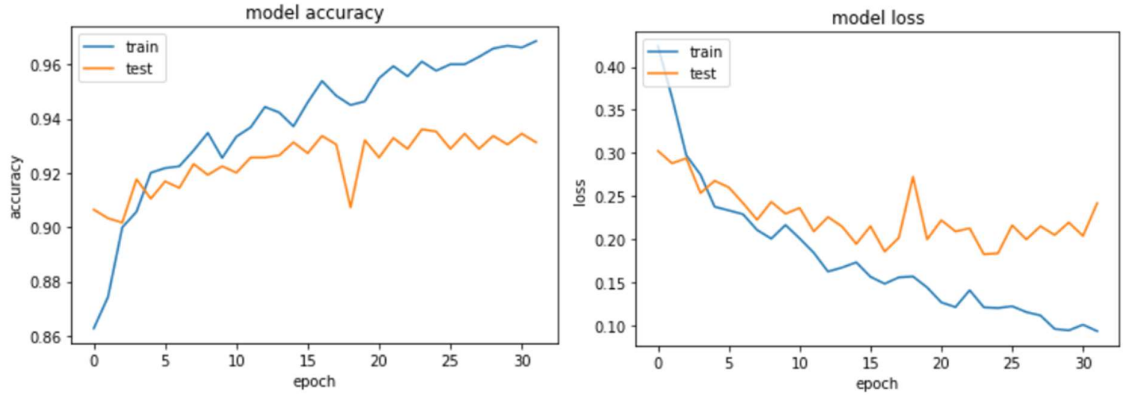
Train – 96.43

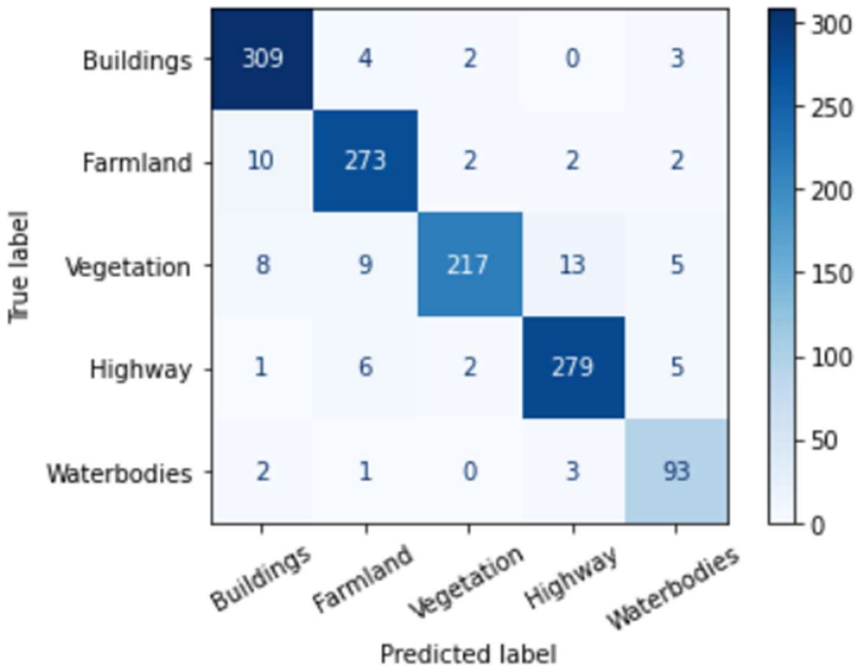
Val – 93.569

Base Model –



Fine Tuning -





	precision	recall	f1-score	support
Buildings	0.94	0.97	0.95	318
Farmland	0.93	0.94	0.94	289
Vegetation	0.97	0.86	0.91	252
Highway	0.94	0.95	0.95	293
Waterbodies	0.86	0.94	0.90	99
accuracy			0.94	1251
macro avg	0.93	0.93	0.93	1251
weighted avg	0.94	0.94	0.94	1251

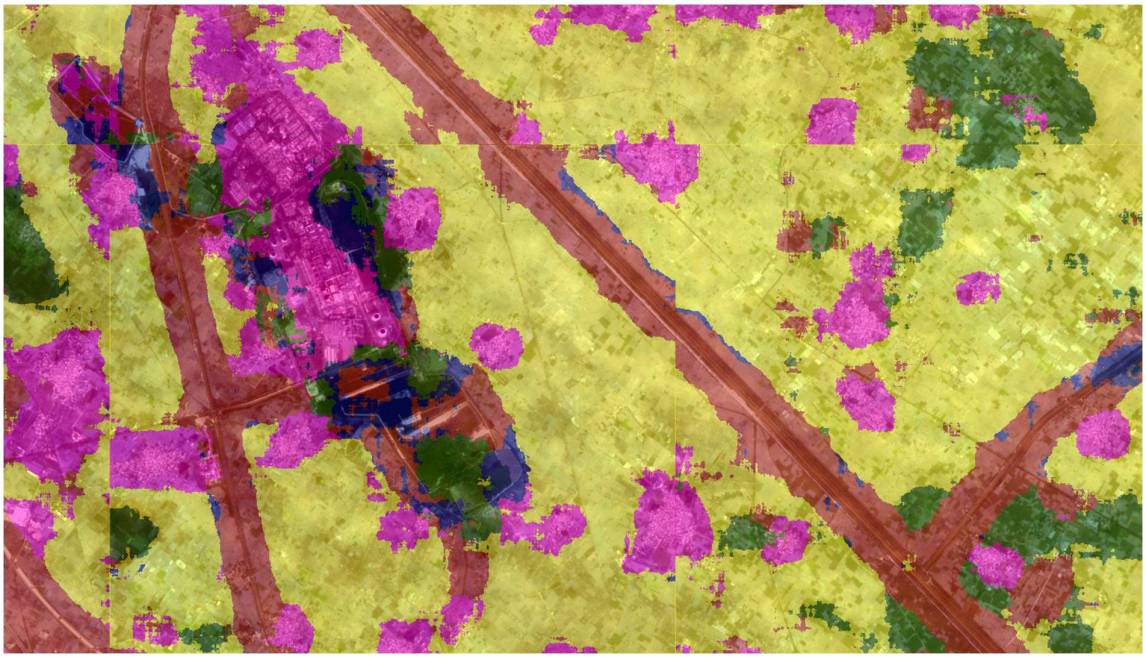


Figure 13: Processed RGB image using VGG-19

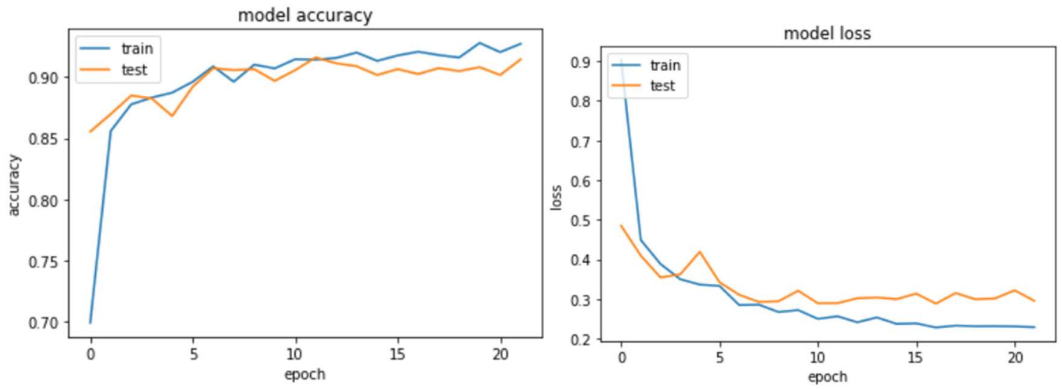
6.1.3 DenseNet

Accuracy –

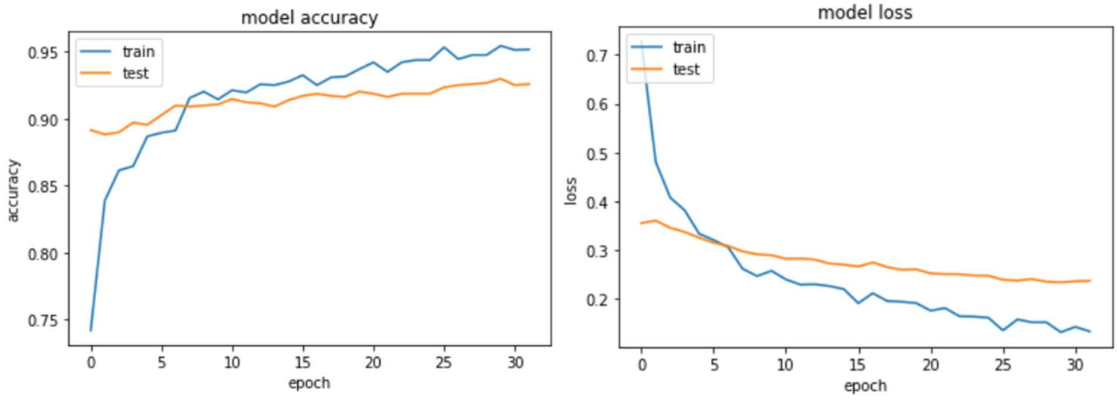
Train – 93.72

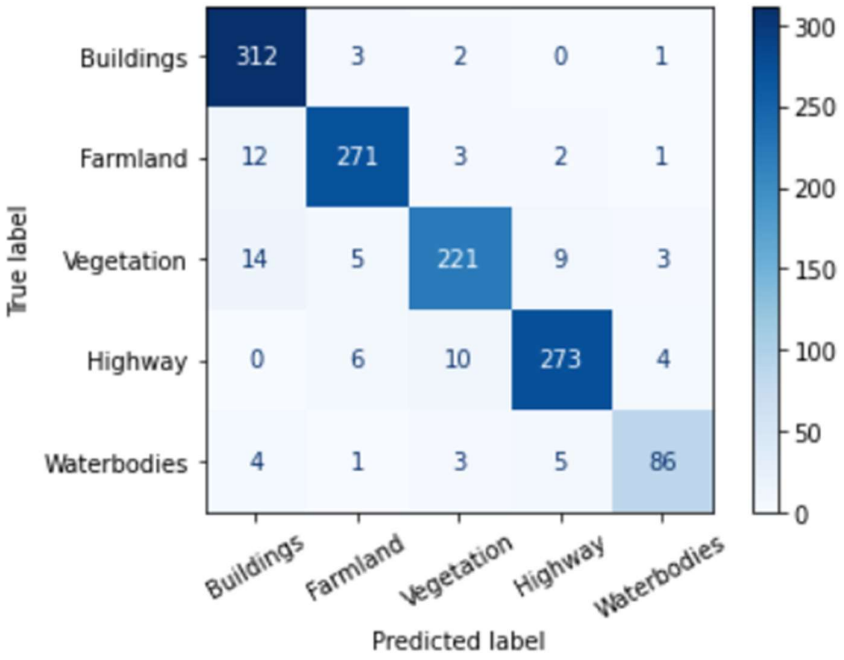
Val – 86.797

Base Model –



Fine Tuning -


model accuracy
train
test
accuracy
epoch
model loss
train
test
loss
epoch


Buildings
Farmland
Vegetation
Highway
Waterbodies
True label
Predicted label
312 3 2 0 1
12 271 3 2 1
14 5 221 9 3
0 6 10 273 4
4 1 3 5 86
300
250
200
150
100
50
0

	precision	recall	f1-score	support
Buildings	0.91	0.98	0.95	318
Farmland	0.95	0.94	0.94	289
Vegetation	0.92	0.88	0.90	252
Highway	0.94	0.93	0.94	293
Waterbodies	0.91	0.87	0.89	99
accuracy			0.93	1251
macro avg	0.93	0.92	0.92	1251
weighted avg	0.93	0.93	0.93	1251

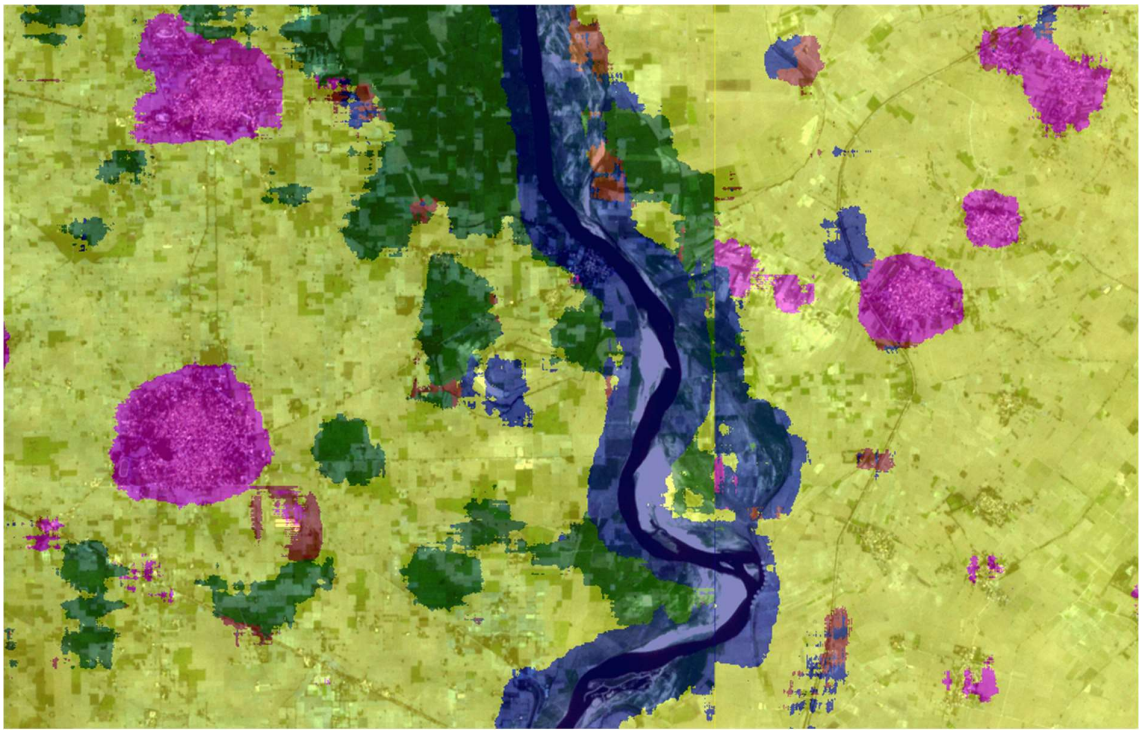


Figure 14: Processed RGB image using DenseNet

6.2. Metrics for Multispectral Dataset -

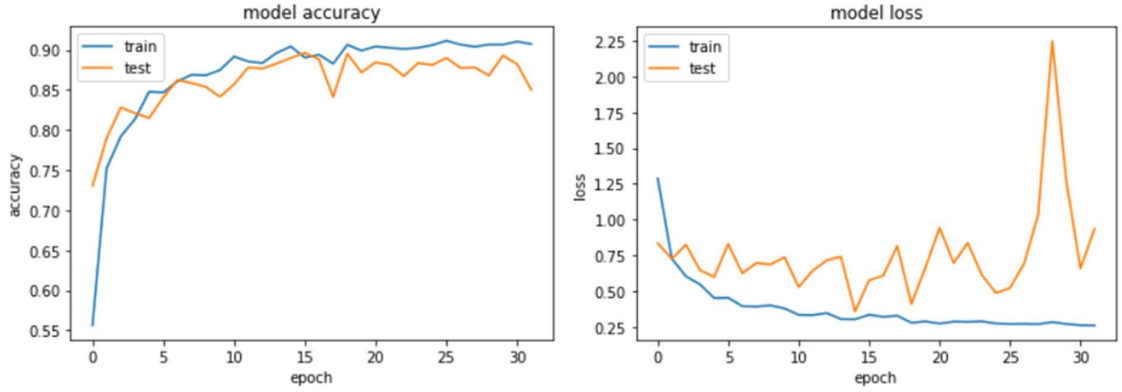
6.2.1 DenseNet -

Accuracy –

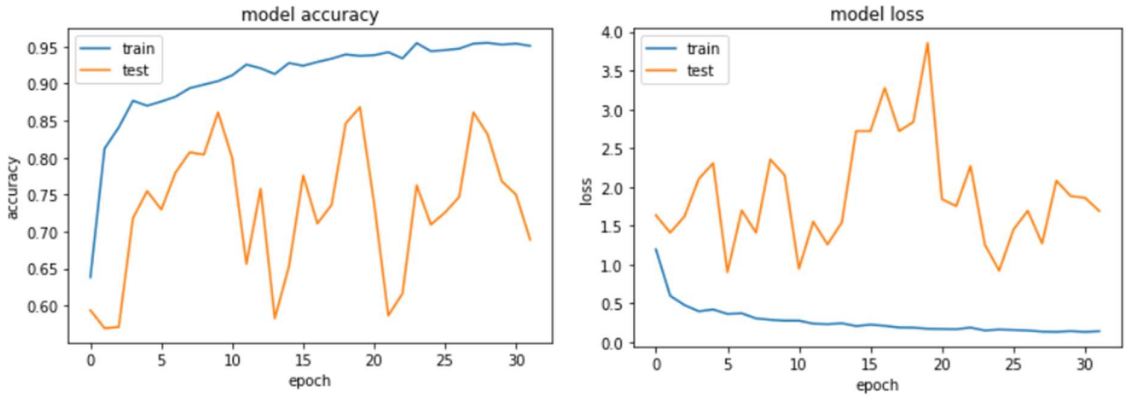
Train – 93.72

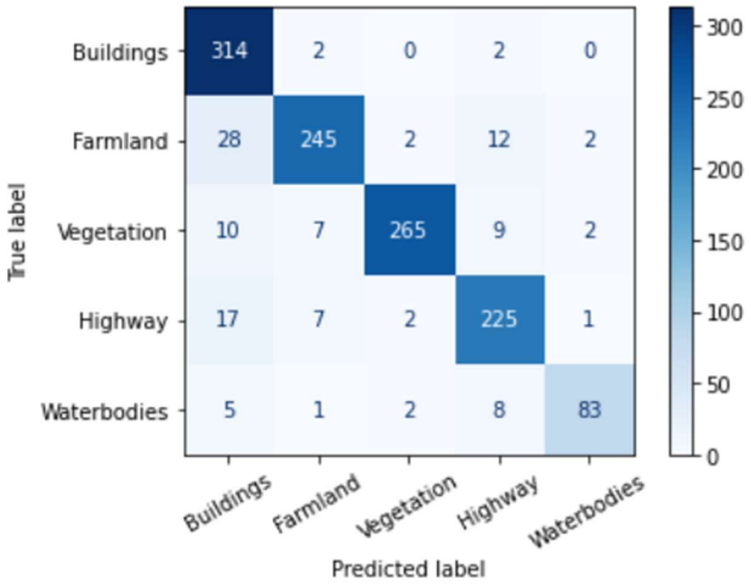
Val – 86.797

Base Model –



Fine Tuning –





	precision	recall	f1-score	support
Buildings	0.84	0.99	0.91	318
Farmland	0.94	0.85	0.89	289
Vegetation	0.98	0.90	0.94	293
Highway	0.88	0.89	0.89	252
Waterbodies	0.94	0.84	0.89	99
accuracy			0.90	1251
macro avg	0.91	0.89	0.90	1251
weighted avg	0.91	0.90	0.90	1251

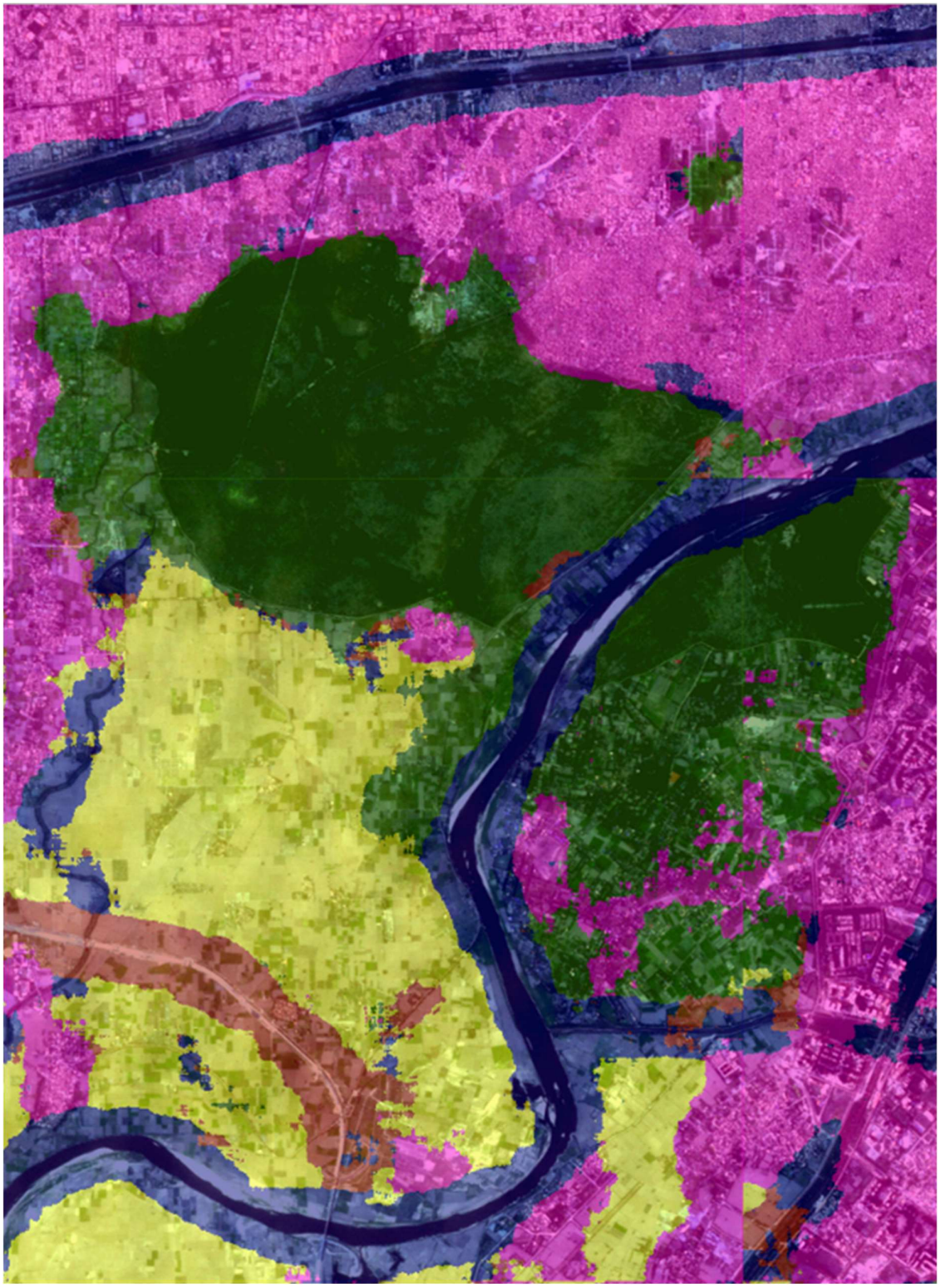


Figure 15: Processed 10-band multispectral image using Densenet

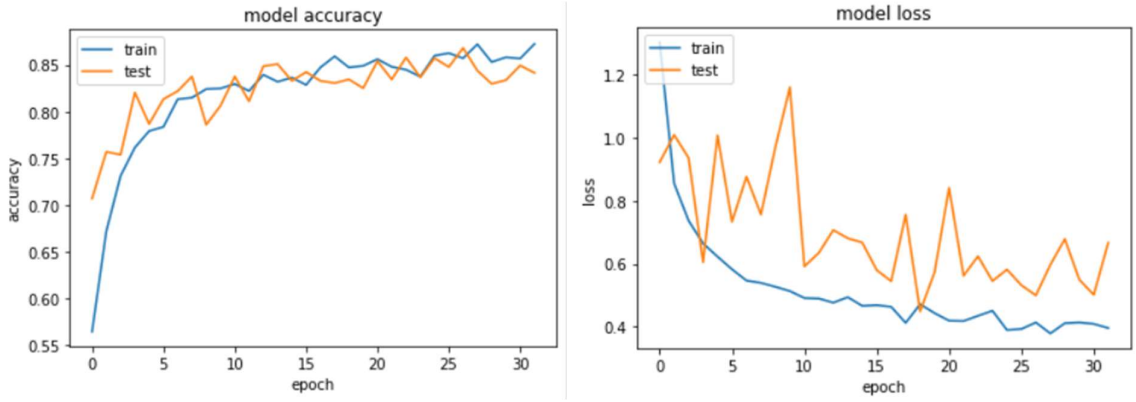
6.2.2 VGG – 16

Accuracy –

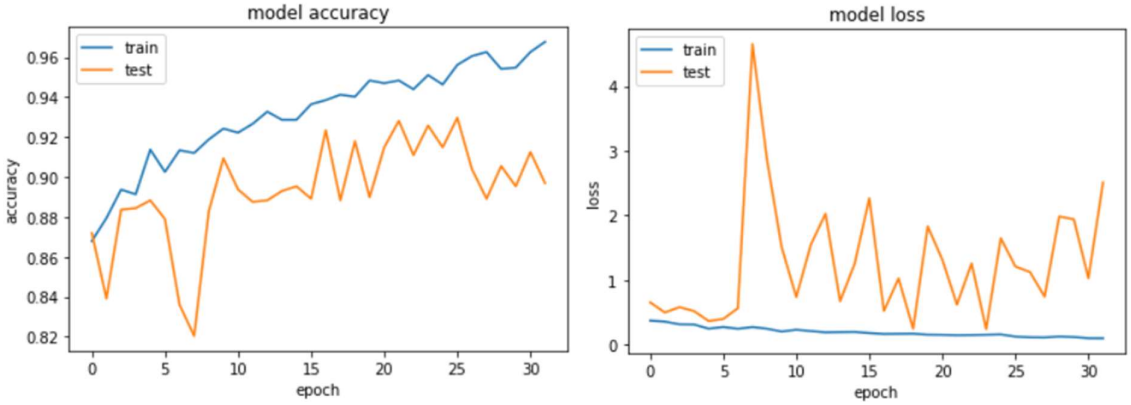
Train – 96.77

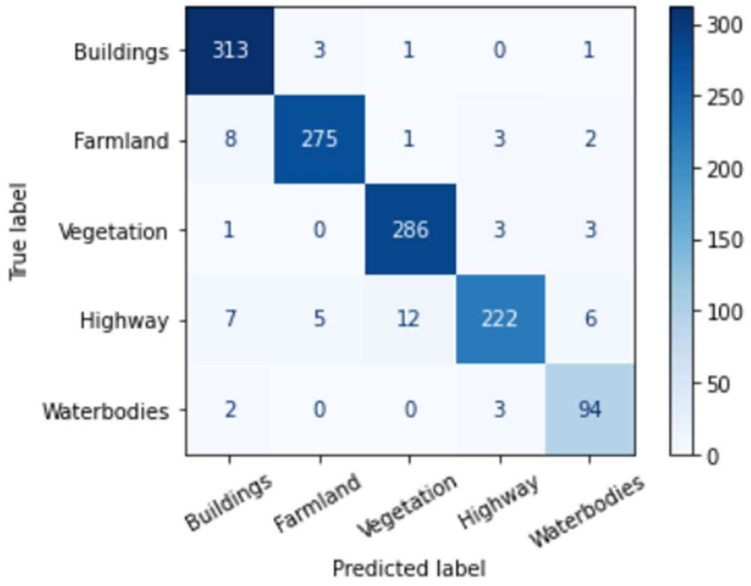
Val– 92.769

Base Model –



Fine Tuning –





	precision	recall	f1-score	support
Buildings	0.95	0.98	0.96	318
Farmland	0.97	0.95	0.96	289
Vegetation	0.95	0.98	0.96	293
Highway	0.96	0.88	0.92	252
Waterbodies	0.89	0.95	0.92	99
accuracy			0.95	1251
macro avg	0.94	0.95	0.95	1251
weighted avg	0.95	0.95	0.95	1251

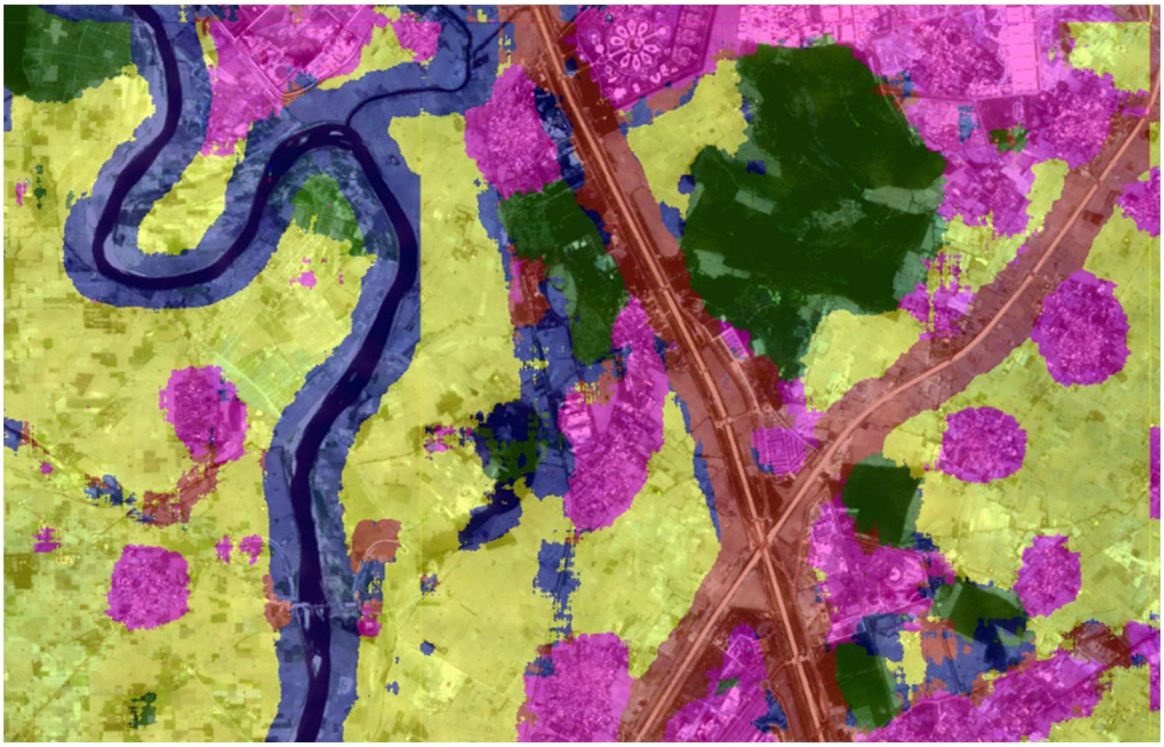


Figure 16: Processed 10-band multispectral image using VGG-16

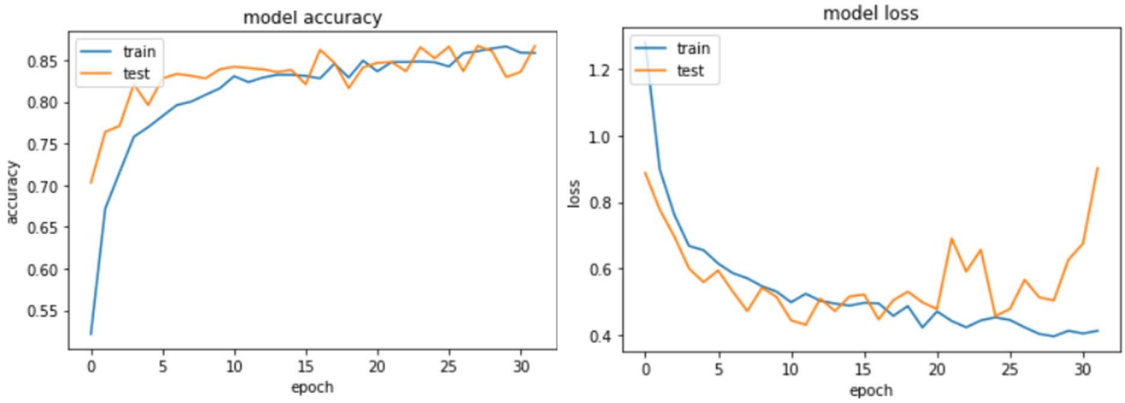
6.2.3 VGG – 19

Accuracy –

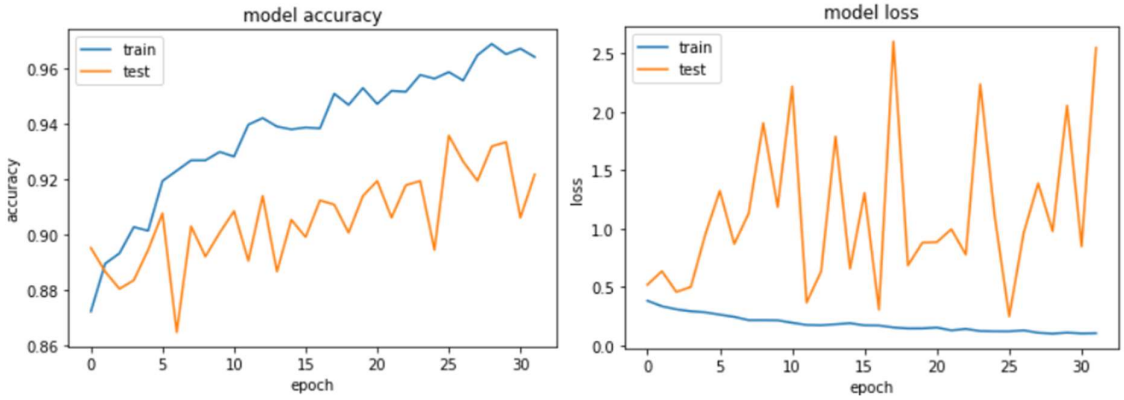
Train – 96.569

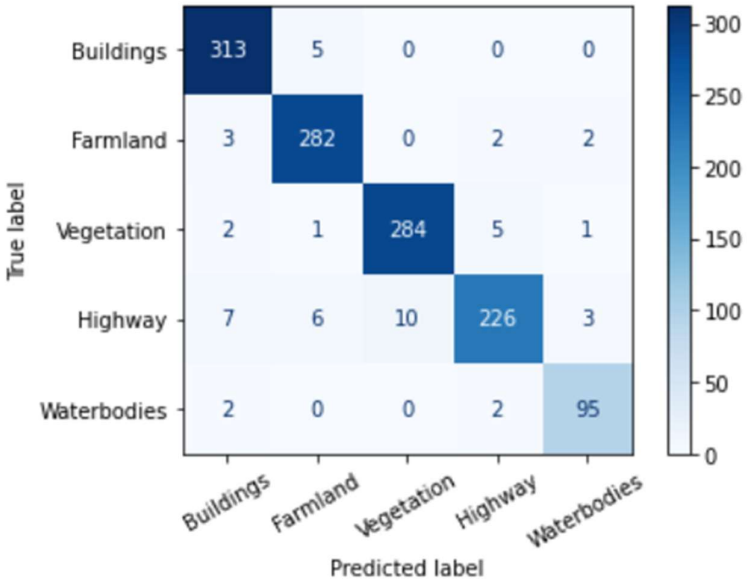
Val – 94.569

Base Model –



Fine Tuning –





	precision	recall	f1-score	support
Buildings	0.96	0.98	0.97	318
Farmland	0.96	0.98	0.97	289
Vegetation	0.97	0.97	0.97	293
Highway	0.96	0.90	0.93	252
Waterbodies	0.94	0.96	0.95	99
accuracy			0.96	1251
macro avg	0.96	0.96	0.96	1251
weighted avg	0.96	0.96	0.96	1251

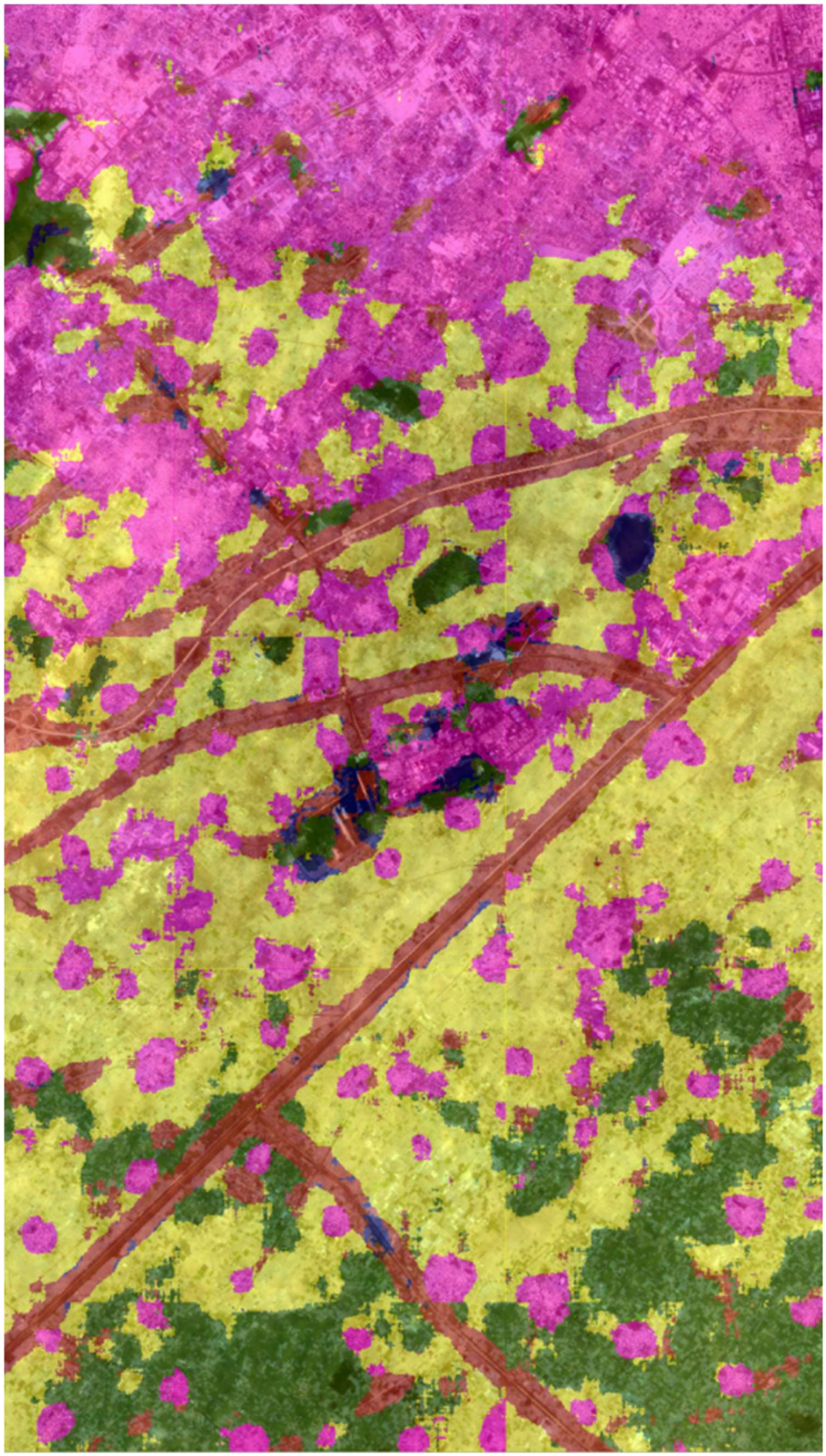


Figure 17: Processed 10-band multispectral image using VGG-19

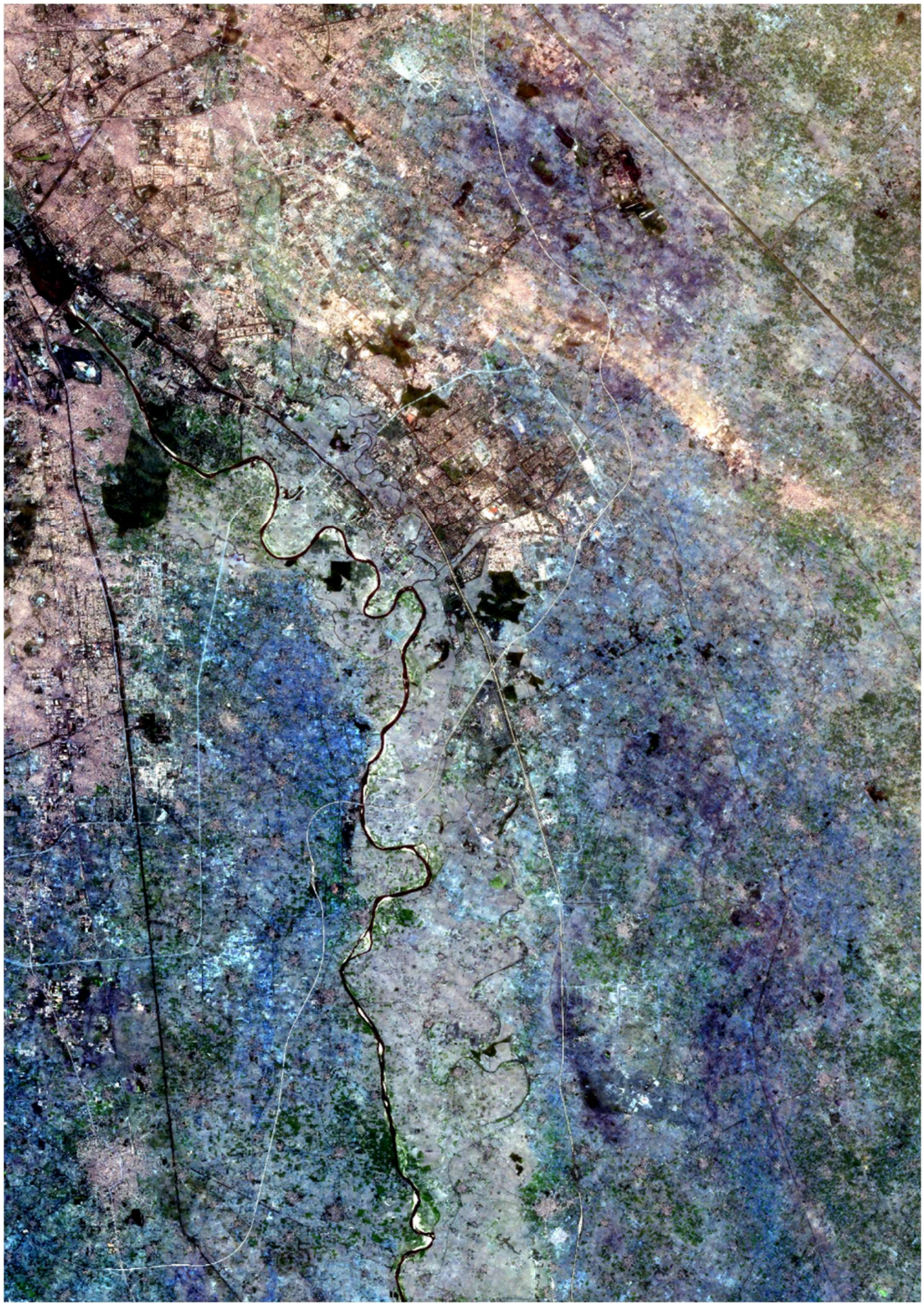


Figure 18: Gautam Buddh Nagar 47.18 Km x 64.84 Km (3,05,9 Km^2).

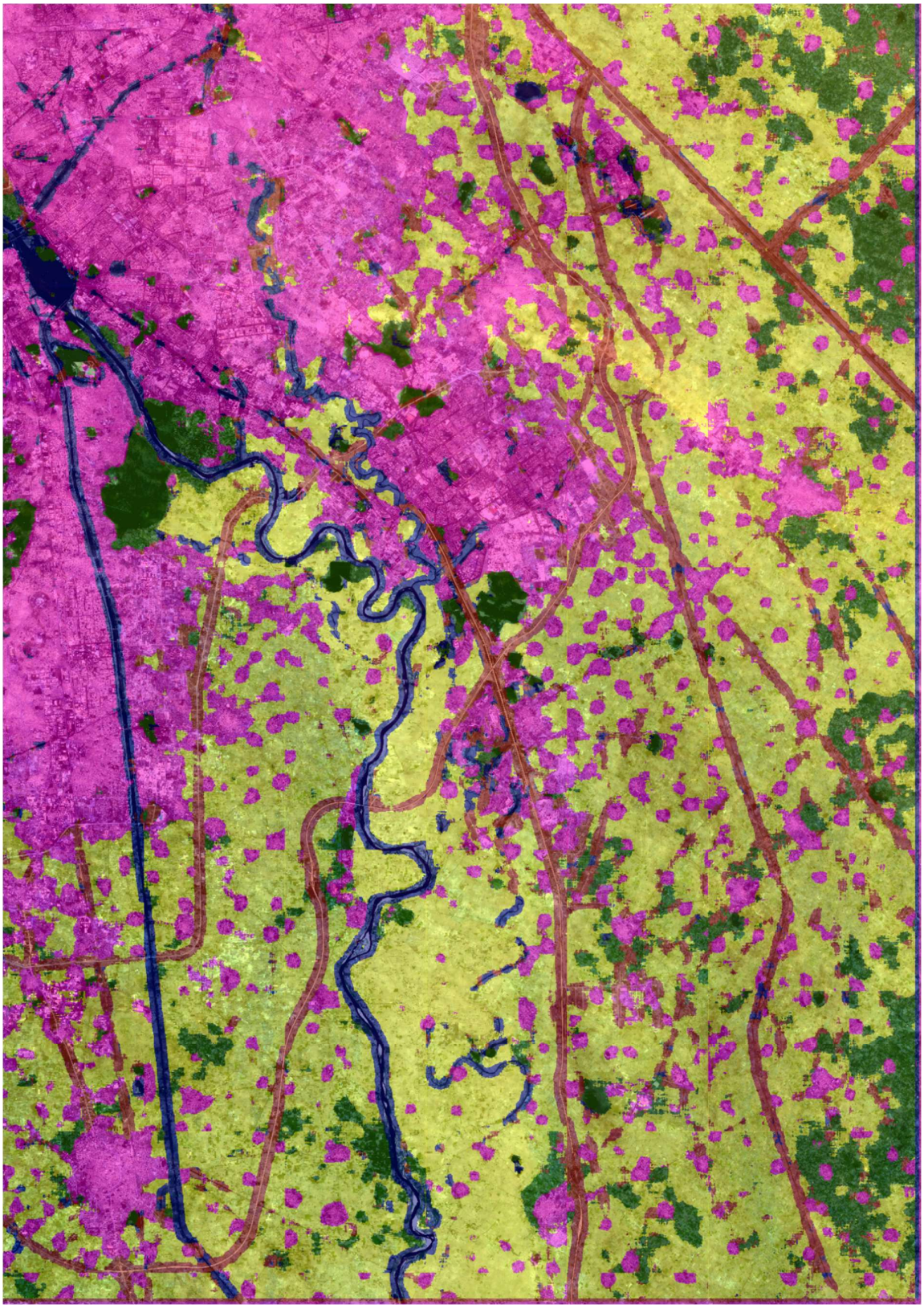


Figure 19: Gautam Buddh Nagar 47.18 Km x 64.84 Km (3,05,9 Km^2) using MS VGG-19 Model.

Conclusion

By analysing the observations made using the findings of the deep learning segmentation model, following points can be concluded:

- Multispectral data has higher accuracy than RGB data.
- Models trained using Transfer learning yield better results than models trained from scratch
- Sometimes models trained from scratch may not converge without any significant modification
- Using generators for data augmentation speeds up the training process

Future work:

The accuracy of land use classifier developed in this project depends greatly on both quality and quantity of the dataset used. Thus, later on we will be training our model with better quality data with the aim to maximize the evaluation metrics (accuracy, precision, recall).

Using a larger dataset also increases the accuracy of a classifier. So, increasing the size of dataset can also be done to gain the desired accuracy.

Also, we will be creating and training our model on a four-band dataset. The four band dataset images will include Red, Green, Blue and Infrared bands.

References:

1. M. P. Vaishnave, K. Suganya Devi, A study on deep learning models for satellite imagery, https://www.ripublication.com/ijaer19/ijaerv14n4_06.pdf
2. Wahabou Abdou, Albert Dipanda, Application of Deep Learning Architectures for Satellite Image Time Series Prediction, https://www.mdpi.com/2072-4292/13/23/4822
3. Shivaprakash Muruganandham, Implementation of Training Convolutional Neural Networks, https://arxiv.org/abs/1506.01195
4. Karen Simonyan and Andrew Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, https://arxiv.org/abs/1409.1556
5. Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger, Densely Connected Convolutional Networks, https://arxiv.org/abs/1608.06993
6. https://www.researchgate.net/publication/346716209_A_Convolutional_Neural_Network_Classifier_VGG-19_Architecture_for_Lesion_Detection_and_Grading_in_Diabetic_Retinopathy_Based_on_Deep_Learning
7. Recurrent neural networks → Rußwurm, Körner (arXiv:1802.02080)
8. Abhishek Verma, Compressed residual-VGG16 CNN model for big data places image recognition, https://ieeexplore.ieee.org/document/8301729
9. Luka Rumora, Impact of Various Atmospheric Corrections on Sentinel-2 Land Cover Classification Accuracy Using Machine Learning Classifiers, https://www.mdpi.com/2220-9964/9/4/277
10. Sheldon Mascarenhas, A comparison between VGG16, VGG19 and ResNet50 architecture frameworks for Image Classification, https://ieeexplore.ieee.org/document/9687944
11. Darius Phiri, Sentinel-2 Data for Land Cover/Use Mapping, https://www.mdpi.com/2072-4292/12/14/2291
12. Arun Kumar Dubey, Automatic facial recognition using VGG16 based transfer learning model, https://www.tandfonline.com/doi/abs/10.1080/02522667.2020.1809126
13. Ishrat Zahan Mukti, Dipayan Biswas, Transfer Learning Based Plant Diseases Detection Using ResNet50, https://ieeexplore.ieee.org/abstract/document/9068805

14. Anuvi Rawat, Anil Kumar, Priyadarshi Upadhyay, Shashi Kumar,Deep learning-based models for temporal satellite data processing: Classification of paddy transplanted fields, https://www.researchgate.net/publication/348641879_Deep_learning-based_models_for_temporal_satellite_data_processing_Classification_of_paddy_transplanted_fields

15. Zichao Jiang, A novel crop weed recognition method based on transfer learning from VGG16 implemented by Keras, https://www.researchgate.net/publication/337872324_A_Novel_Crop_Weed_Recognition_Method_Based_on_Transfer_Learning_from_VGG16_Implemented_by_Keras