THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

Data Science Program

Capstone Report - Spring 2022

Mapping Deprived Areas Using Deep Neural Networks and Low-Resolution Satellite Images

Abdulaziz Gebril,
Mina Hanna,
Mojahid Osman

supervised by
Amir Jafari

Abstract

Mapping of deprived areas has been a key research topic over the past years as it helps optimize the development planning and decision making by governments and other policy makers. This research aims to utilize free to use satellite images to map deprived areas with relatively smaller sized window of 100m2.

The project will utilize various geographic transformation techniques, image processing, classification algorithms and deep neural networks to classify and map deprived areas. First, cloud free maps were generated to help with having clear images for further processing. Second, image processing, augmentation and other geographic transformations took place on training data maps other data sources to build a comprehensive, balanced dataset for deep learning training. Third, various machine learning techniques were applied across different tracks to qualify the best models that will be combined to provide the best results of the mapping. We reached f1-score around 0.8 using low resolution satellite imagery.

Contents

1 Introduction

Mapping deprived areas from satellite images and other data sources will help governments, international and independent organizations to categorize the level of deprivation, prioritize projects and closely monitor the progress. This domain has been a key research area for the past decade with a lot of promising results supporting development efforts in line with various global initiatives to end poverty and build sustainable cities. The advancement in machine learning techniques and availability of Geo-data offers an opportunity to achieve efficient models[1].

According to an IdeaMaps review [2], Convolutional Neural Network and Classical Machine Learning algorithms are the most used techniques in mapping deprived areas in recent key publications, but they are mostly limited due to their high computational requirements. This project aims to apply different machine learning techniques on low resolution and free satellite images along with distance covariate and contextual data and other free datasets to detect deprived areas on a 100m2 level. This approach will balance the equation to have a reasonably computational requirement without losing the ability to scale the solution.

The two core streams of the projects to detect deprived areas were as follows:

1- Processing and usage of raw satellite images
2- Utilizing covariate and contextual data

This report is focusing on the first stream to use labeled satellite images to train Convolutional Neural Network (CNN) and classify deprived areas. The labeled data was mainly provided by Ideamapsnetwork based on their ground observation work where deprived areas were mapped based on an agreed survey that will be conducted by a local team in different cities.

2 Problem Statement

The main question that is being addressed in this study is whether affordable low resolution images can be used to map deprived areas with accepted accuracy and performance ? The term “affordable” indicates that the images (or data sources in general) are royalty free with reasonable efforts conducted to accurately label them.

The classification of deprived areas might require country or regional based training due to the evolutions and specifics for the different deprived areas which will require extensive work to handle the following challenges:

3 **Imbalanced data:** since labeling the deprived areas requires an expensive and lengthy groundwork, the deprived areas mapped in the training dataset are few compared to non-deprived areas.
4 **Image quality:** free and available satellite imagery for some major cities is not clear due to clouds
5 **Labeling small areas of 100m2 in size**: extracting features for machine learning models can be very challenging when dealing with small images (10x10 pixels)

6 Data Source

The data that was used in training the Convolutional Neural Network are as follows:

- Sentinel-2 satellite image for major cities in Africa
- Enhanced satellite image constructed via google earth engine with reduced clouds
- Training map with labeled 10x10m areas with the following labels:
    - Built-up area
    - Deprived area

- Non-built-up area

The training map was constructed by Ideamap network through several iterations of on-ground surveys conducted via trained teams on a 100m2 levels.

Since NIR (Near-Infrared) channel provides vital information for land cover classification, especially concerning vegetation assessment, it was used in the construction of the final image that will be used in the training.

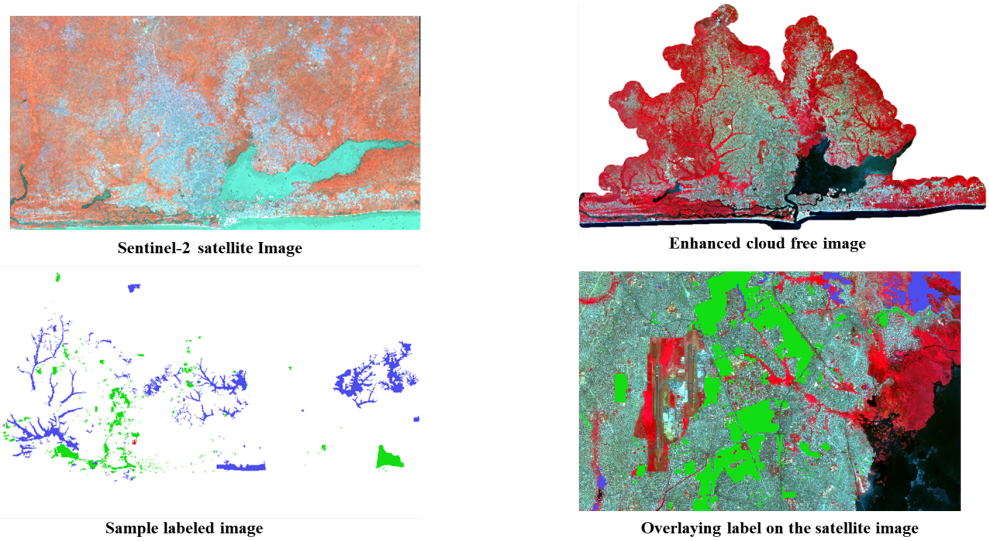


Figure 1 Data sources Summary

7 Related Work

Krishna et al [3] used Very High-Resolution Satellite images and applied CNN model to have a multi-class and Hierarchical class classifier able to classify informal areas with detailed labels like single/multi-story with overall accuracy reaching 0.71 and F-1 score of 0.70 for the multi-class classification.

Monika et al [4] used Very High-Resolution Satellite images and applied a two-step CNN model (where one is trained on binary classification and the other added data from Data-Driven Index of Multiple Deprivation with image augmentation and reached an accuracy of 0.984

8 Solution and Methodology

The solution is split into two phases, the first phase involves all the image extraction, transformation, and augmentation to prepare a dataset ready for deep learning models. The second phase is to train different deep learning models on the classification.

The following figure illustrates the two phases of the project and how they are used to construct a pipeline for different experiments that was conducted throughout the project.

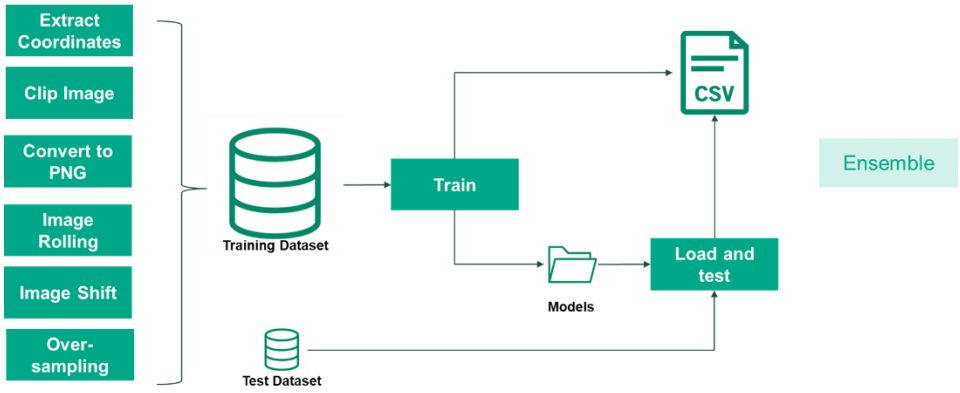


Figure 2 Solution approach

The following sections will go through the different steps in each phase to provide more details.

8.1 Extracting coordinates

After examining the data using different GIS applications, the first step is to extract all the data from the labeled image and export the data table that contains coordinates of the labeled box and the corresponding label. Each coordinate represents the center pixel in a 10x10 meter square.

	A	B	C
1	long	lat	Label
2	3.20417	6.91167	0
3	3.20167	6.91083	0
4	3.2025	6.91083	0
5	3.20333	6.91083	0
6	3.20417	6.91083	0

Figure -3 Sample of the extracted coordinates

Once the coordinates are extracted, a test dataset is extracted to be used to evaluate the model. This subset of the coordinates will not undergo any augmentation or further processing that will be explained in the next sections.

8.2 Clipping images

After the coordinate extraction, this step will clip the 10x10m map image that corresponds to each coordinate from the satellite image and store the labeled image for further processing.

Figure -4 The process of clipping labeled images

8.3 Generating more images

Due to the complexity of mapping 10x10 m2 areas, the deprived area labels are significantly less than non-deprived labels and this required to generate more images from the minority class (deprived) to construct the required dataset for model training.

This process involved different techniques which can be summarized in the following points:

1- Shifting images by 1 or 2 meters in 4 directions
2- Detect any adjacent labeled areas and roll through them to generate intersecting new images
3- Rotate the image in different directions

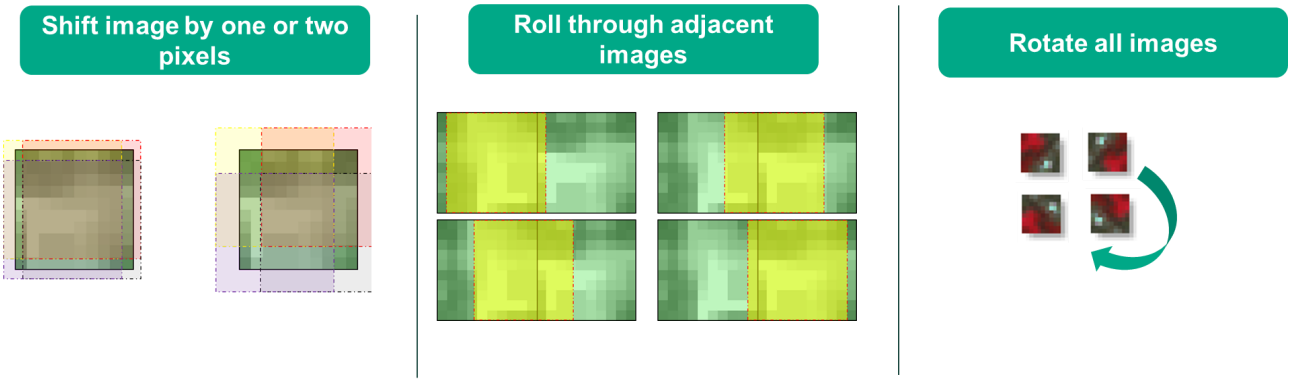


Figure 5 Techniques used to generate new images

8.4 Convert GeoTiff images to PNG

The final step in the image processing pipeline is to convert the GeoTiff images to PNG for model training. Since each GeoTiff might have its own bands (channels) order and ranges, the conversion has to take into account the usage of NIR band and also normalize the values of each pixel from 0-255 to fit with RGB standards.

Band	Min	Max
Band 1: B8	352.5000000000	5246.0000000000
Band 2: B4	422.0000000000	4056.0000000000
Band 3: B3	504.0000000000	3918.0000000000
Band 4: B2	228.0000000000	3577.0000000000

Figure 6 Sample bands from a GeoTiff file

8.5 Data distribution

Generated images were divided into training and test (later training is divided into training and validation) using random stratified sampling

8.6 Multi Layer Perceptron (MLP)

Multi layer perceptron (MLP) is one of the most used artificial neural networks (ANN). The fist algorithm used to train the (NN) neural network was developed in 1986 by Dr.Hinton and his colleagues when they developed the backpropagation algorithm that is used to train a multilayer neural network. Multilayer Perceptron (MLP) is a fully connected multi-layer neural network that consists of three types of layers, the input layer, output layer and hidden layer, as shown in next figure.

The input layer receives the input signal to be processed. The required task such as prediction and classification is performed by the output layer. A number of hidden layers that are placed in between the input and output layer are the true computational engine of the MLP.

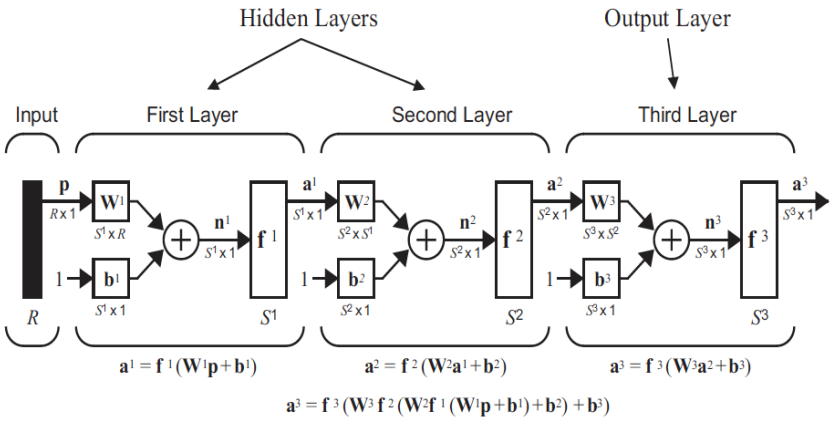


Figure 7 MLP Architecture

8.7 MLP Network

In this network we will be using the “Sequential'' model, so we can stack up layers by adding the layer one by one. As we are building a feedforward network we use the “Dense” layer, also called a fully connected layer. In each Dense layer we are adding an activation function, here in this layer we are using the ReLU activation function. The last layer is a “softmax” layer as it is mostly used for multiclass classification problems but also can work for Binary classification, which is our case model .

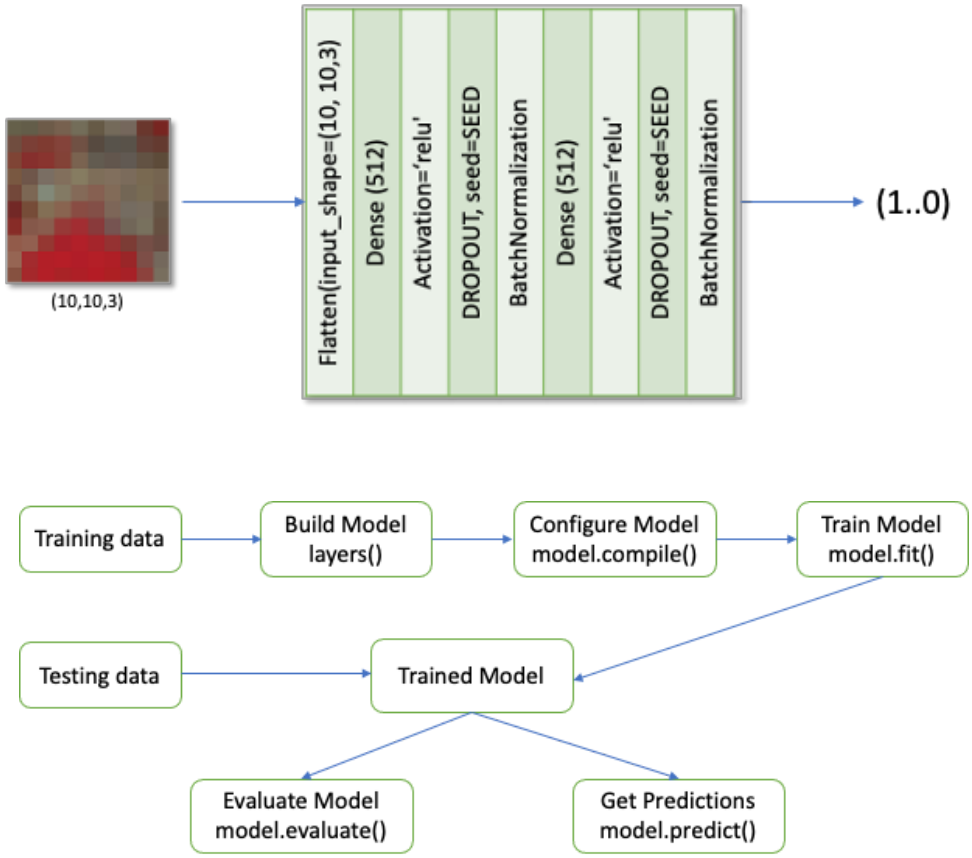


Figure -7 MLP Model Training process

8.8 Convolutional Neural Network

Convolutional Neural Networks come under the subdomain of Machine Learning which is Deep Learning. It was proposed by computer scientist Yann LeCun in the late 90s, when he was inspired by the human visual perception of recognizing things [5]. A convolution network is a multilayer feedforward network that has two- or three-dimensional inputs. It has weight functions that are not generally viewed as matrix multiplication (or inner product) operations, The principal layer type for convolution networks is the convolution layer that convolute a kernel over the image and create feature maps, Pooling layer that is used to subsample the output of the convolution layer which reduce the size while maintaining the features for next layers.

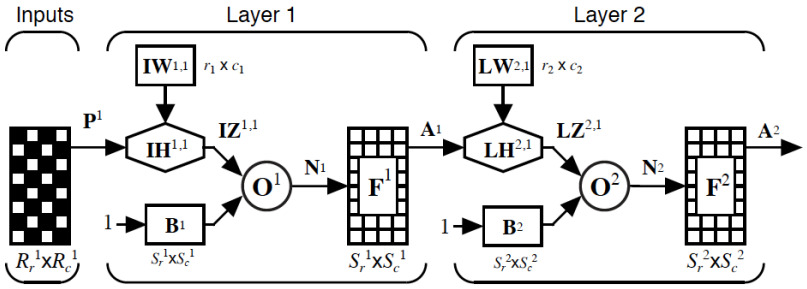


Figure 8: Convolution Neural Network Architecture

For an image v with dimension Rr x Rc and kernel W with size of r x c, the weight function is calculated as follows:

$$z_{i,j} = \sum_{k=1}^{r} \sum_{l=1}^{c} w_{k,l} v_{i+k-1,j+l-1}$$

8.9 Modeling Convolutional Neural Network

The following model architecture was used to train the Convolutional Neural Network on the satellite image captured from the previous steps:

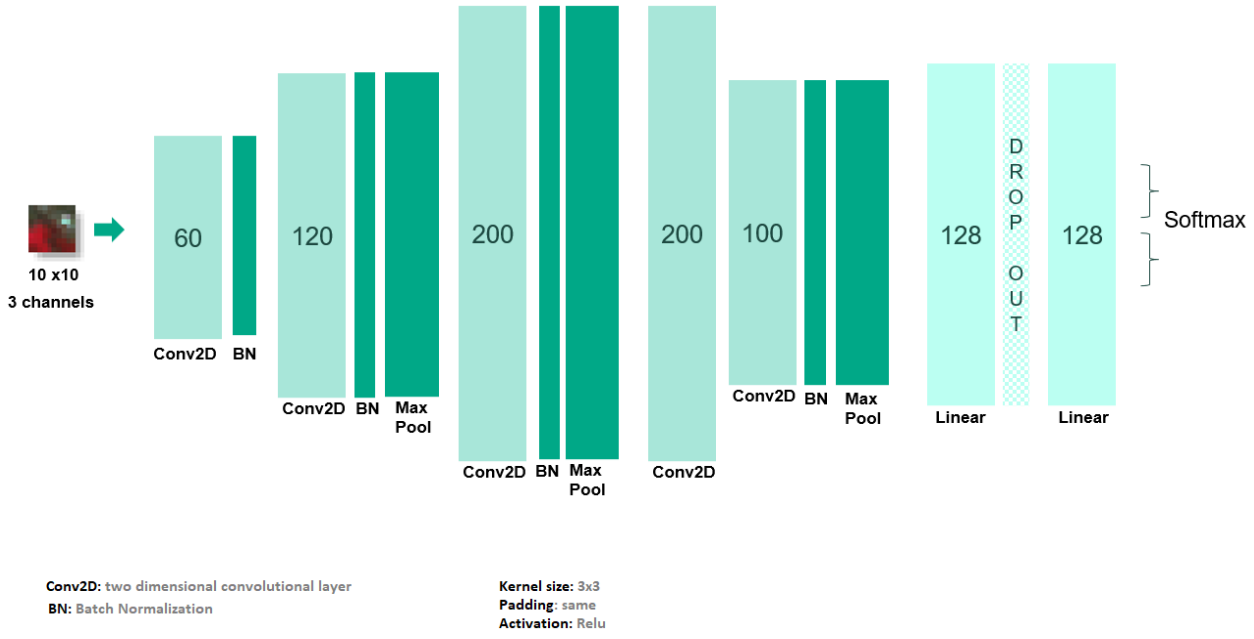


Figure 9: Convolution Neural Network Architecture used in the classification

The following was considered during the training process:

- Pooling, and Fattening layers will be applied. We also added a Dropout layer to reduce overfitting. Since it's a binary class problem, the Sigmoid activation function is applied, and we used "Softmax" with 2 outputs instead. Batch normalization is used to keep output close to mean of 0 and variance of 1 which prevents the saturations of the activation function.
- Padding with kept the same to ensure that the size (10x10) will go through the deeper layers, kernel size of all layers was set to 3 and the activation function was set to "Relu"
- We used 100 epochs with an early stopping option to avoid overfitting.
- Our main metrics for the model were accuracy, precision, recall and F1-score SK-learn package was used to obtain the confusion matrix, recall, precision, and accuracy.
  Accuracy: ratio of all correct classification against all data ***(TP + TN)/(TP+TN+FP+FN)***
  Precision: percentage of classification that are relevant ***(TP)/(TP+FP)***
  Recall: percentage of relevant classification ***(TP)/(TP+FN)***

  TP: True Positive **TN:** True Negative **FP:** False Positive **FN:** False Negative

8.10 Grid search

In order to optimize training time and conduct hyperparameter tuning for our models. Several models were trained and tested using Gridsearch CV from sklearn library. The training data was split into 5 folds and trained on several models. The best model chosen based on f1-macro scoring. The models trained had the same convolutional Architecture trained in the previous section. The output layer contains one neuron and a sigmoid activation function for binary classification. The parameters tuned are as follows

- Number of Neurons in each convolutional layer.
- Learning rate
- Number of epochs
- Dropout out rate in MLP layers

8.11 Autoencoder

An Autoencoder is an unsupervised learning neural network. It is primarily used for learning data compression and inherently learns an identity function. First introduced in the 1980s, it was promoted in a paper by Hinton & Salakhutdinov in 2006.

An autoencoder is composed of an encoder and a decoder sub-models. The encoder compresses the input and the decoder attempts to recreate the input from the compressed version provided by the encoder.

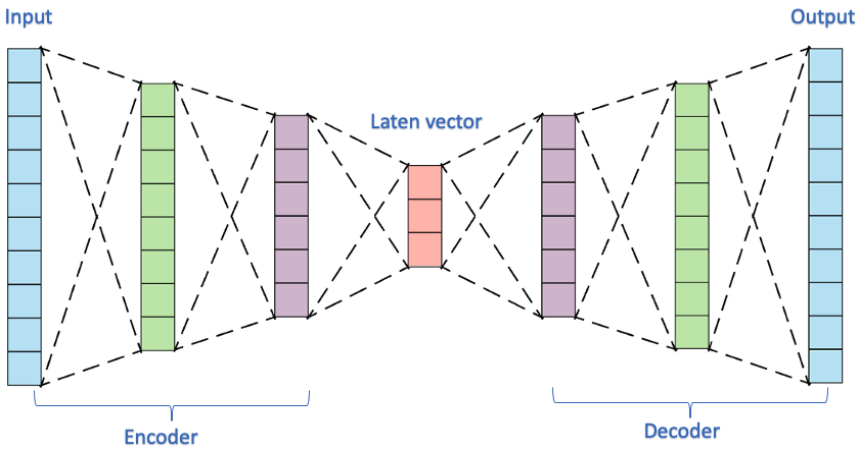


Figure 10: Auto-Encoder architecture

However, an autoencoder is used for generating new images that are similar to the input images, the encoder part can be used as a classifier, so after training the autoencoder model we can save the encoder model and use it later as a pretrained model.

In this project we use an autoencoder to generate more images for deprived areas (class 1), and new augmented images to the train date to see how the model performance will be changed.

233/233 [==========] - 1s 4ms/step - loss: 0.0020 - val_loss: 0.0021

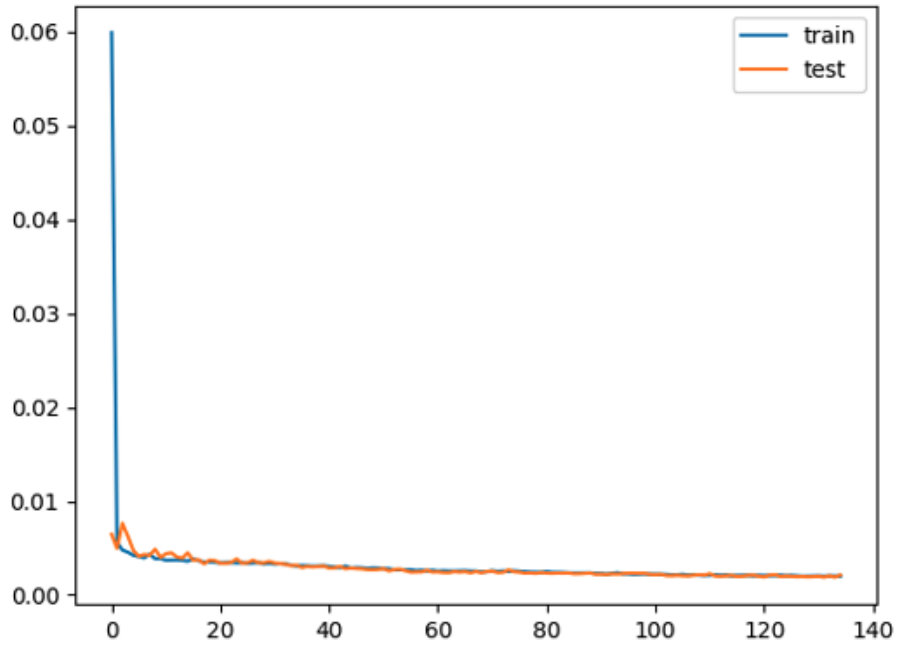


Figure 11: Auto-Encoder losses

8.12 Google Open Buildings

Google Research worked on developing a large-scale open dataset containing the outlines of buildings derived from high-resolution satellite imagery[6] to support several objectives such as population estimation, urban planning and other various applications[7]. The dataset contains 516 million building detection, across an area of 19.4 million square kilometers which represents about 64% of the African Continent.

Google Research utilized deep learning methods to detect buildings across the entire african continent using 50 cm satellite imagery. Experiments were carried out using a dataset of 100k satellite images across Africa containing 1.75M manually labeled building instances, and further datasets for pre-training and self-training. The pipelines created to train the models resulted in creating the open building dataset.

Buildings in this dataset are represented as polygons. The dataset contains various useful information for each building polygon . the building longitude and latitude of the building polygon centroid, area in square meters , confidence score assigned by model and the building polygon geometry which represents the coordinates forming the building. There is no information about the type of building, its street address, or any other details.

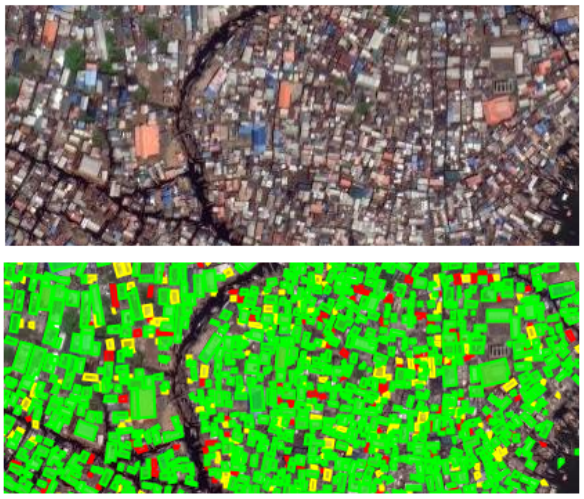


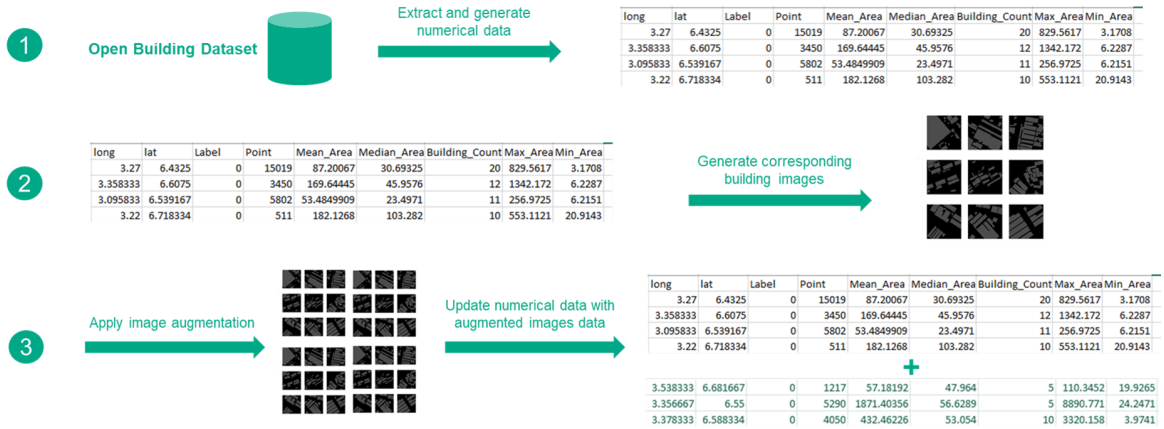
Figure 12: Google Open Building dataset can help mapping deprived areas by analyzing polygons in the dataset.

8.12.1 Pre-Processing

Initially , the data relevant to the region of study was extracted. The generated file is a csv file containing all information described above about building polygons within lagos. This is followed by identifying the building polygons within each label from the training data. Finally, Training lebel images were reconstructed using the geometry of the building polygons. In addition, Descriptive data was extracted and aggregated for each training label point. The output of this extraction included descriptive information about building polygons within the 100 square meter label. Information such as the following ;

1. Mean area in square meters of buildings polygons.
2. Median area in square meters of buildings polygons.
3. Maximum area in square meters of buildings polygons.
4. Minimum area in square meters of buildings polygons.
5. Count of building polygons.

Second step was to generate an image based on the polygons in the dataset which can be used for model training as well and apply the image augmentation on the generated images.



long	lat	Label	Point	Mean_Area	Median_Area	Building_Count	Max_Area	Min_Area
3.27	6.4325	0	15019	87.20067	30.69325	20	829.5617	3.1708
3.358333	6.6075	0	3450	169.64445	45.9576	12	1342.172	6.2287
3.095833	6.539167	0	5802	53.4849909	23.4971	11	256.9725	6.2151
3.22	6.718334	0	511	182.1268	103.282	10	553.1121	20.9143

long	lat	Label	Point	Mean_Area	Median_Area	Building_Count	Max_Area	Min_Area
3.27	6.4325	0	15019	87.20067	30.69325	20	829.5617	3.1708
3.358333	6.6075	0	3450	169.64445	45.9576	12	1342.172	6.2287
3.095833	6.539167	0	5802	53.4849909	23.4971	11	256.9725	6.2151
3.22	6.718334	0	511	182.1268	103.282	10	553.1121	20.9143
3.538333	6.681667	0	1217	57.18192	47.964	5	110.3452	19.9265
3.356667	6.55	0	5290	1871.40356	56.6289	5	8890.771	24.2471
3.378333	6.588334	0	4050	432.46226	53.054	10	3320.158	3.9741

Figure 12: Google Open Building data pre-processing pipeline

8.12.2 Hybrid model

Based on the various data sources obtained during this study, A hybrid model containing different architectures was developed and trained. The data used represented the satellite images , constructed open building images and descriptive and aggregated numerical data. The model initially inputs different forms of data into different architectures , this is followed by concatenating the outputs to be trained on multiple MLP layers before classifying.

Satellite images and Open building images were inputs for Two different CNN architectures, While Open building numerical data was an input for MLP model. Various architectures were trained and results will be discussed in the following section.

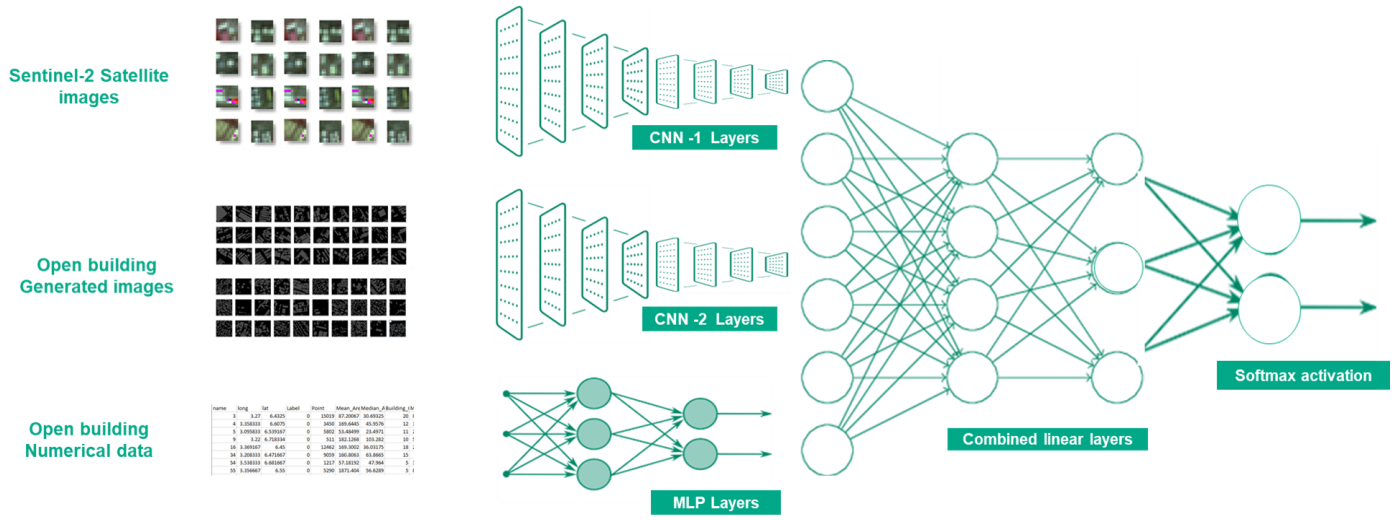


Figure 13: Hybrid model architecture

9 Results and Discussion

The following are the keys results that was observed through the model training and evaluation process:

- Instead of having three labels (Built-up, deprived, and non-built-up) the CNN and MLP models were only trained on two labels (built-up and deprived) since the detection of non-build-up areas like lakes, woods or sea can be achieved via other methodology outside the model scope (i.e., Google Open Building[7] or Open Street Maps)
- Pre-trained network (i.e. VGG16 or VGG19) were not showing any significant performance improvement when compared to standalone Convolution Neural Networks
- Adamax and Adagrad optimizers were showing better performance compared to other optimizers like Adam and RMSprop
- Slight increase in model performance with epochs more than 50 and using a decreasing learning rate
- Assigning class weights for each label to handle class imbalance did not show significant improvement when compared with image augmentation
- Google Open Building showing a reasonable performance especially when used on a pre-trained network (i.e. VGG16)
- Ensemble model was constructed using the best models in the different streams

- Hybrid model is showing promising performance but it requires more complex architecture and higher processing cost

9.1 Experimentation protocol

Since the number of images kept aside for testing the model was relatively small, it was essential to validate that model's performance to avoid inaccurate results due to under-representative data samples. To do so the models were trained using various random seeds to split the data differently and the KPIs consistency was achieved but to further validate the numbers, the following experiment was conducted:

1- Annotation of new labels using manual reviews on high resolution google maps and street views
2- Split the expanded dataset which will have more test images
3- Run the model with more data points
4- Validate the KPIs

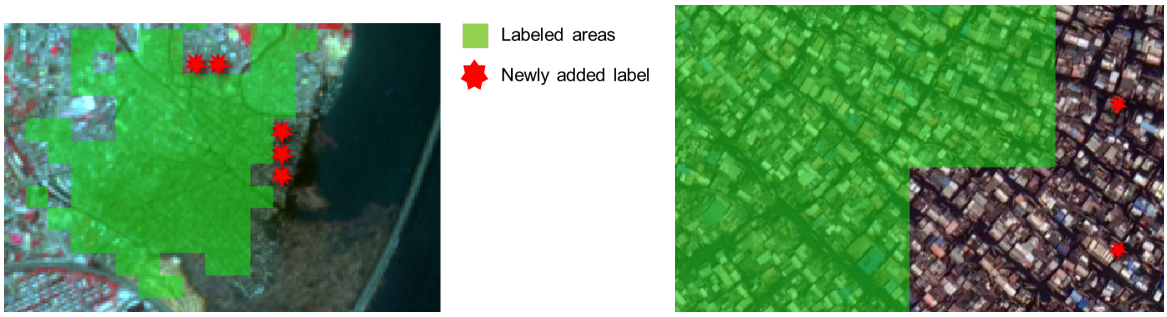


Figure 14: Generation of new labels using high resolution google maps images

9.2 MLP Results

All models were trained for 150 epochs with early stopping and the following table shows the summary of the best performing MLP models:

Model	Optimizer	Accuracy	Precision	Recall	F1 score	Cohen Kappa
Model -1	**RMSProp**	**0.952**	**0.090**	**0.12963**	**0.106**	**0.083**
Model -2	**Adamx**	**0.866**	**0.074**	**0.4444**	**0.127**	**0.093**
Model -3	**Adama**	**0.840**	**0.081**	**0.32111**	**0.144**	**0.087**

The MLP model using Adam optimizer has the highest recall across all models.

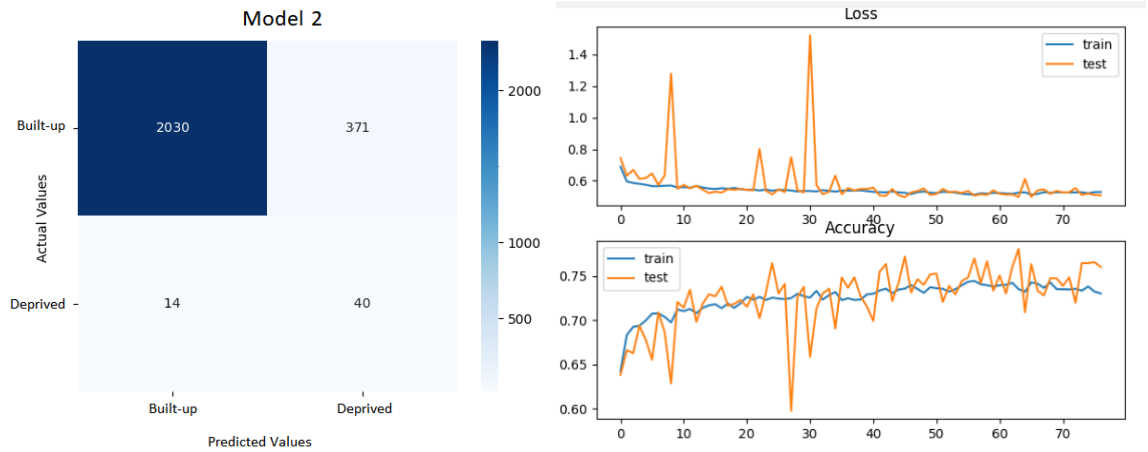


Figure 15: Confusion Matrix and training history of best performing MLP model

9.3 Convolutional Neural Network Results

Using the Sentinel-2 images with Adamax, Adam and RMSProp the best accuracy achieved was 0.78 with Adamax showing the best performance across the optimizers and using two labels (built-up and deprived) instead of using the three labels (built-up, deprived and non-built-up) in the classification. With this results the next training phase was focused the following models:

Model -1	Using raw image generated from google earth engine with reduce cloud
Model -2	Using pre-trained VGG16 network and images from google earth engine with reduce clouds
Model -3	Using image from google earth engine with reduced cloud and pixel dilation
Model -4	Increase the deprived images by shifting original image several times (1 and 2 pixels shift)
Model -5	Ensemble the best models together

The following are the results observed in the training:

	Training Performance				Testing Performance						
Model	**Training Acc**	**Training loss**	**Val. Acc**	**Val. Loss**	**Test Acc**	**Precision "1"**	**Recall "1"**	**f1 score "1"**	**MacroAvg Precision**	**MacroAvg Recall**	**MacroAvg F1 score**
Model-1	0.992	0.0365	0.938	0.2769	0.96	0.35	0.47	0.40	0.67	0.72	0.69
Model-2	0.9947	0.0248	0.955	0.2273	0.96	0.33	0.34	0.33	0.65	0.66	0.66
Model-3	0.957	0.18	0.942	0.12	0.89	0.22	0.72	0.34	0.60	0.81	0.64
Model-4	0.9955	0.0145	0.973	0.1117	0.96	0.32	0.63	0.43	0.66	0.80	0.70
Model -5 (Ensemble)					0.95	0.43	0.60	0.5	0.7	0.78	0.74

The Ensemble model used above was based on the average Softmax logit output of the best three models (using F1 score as a metric).

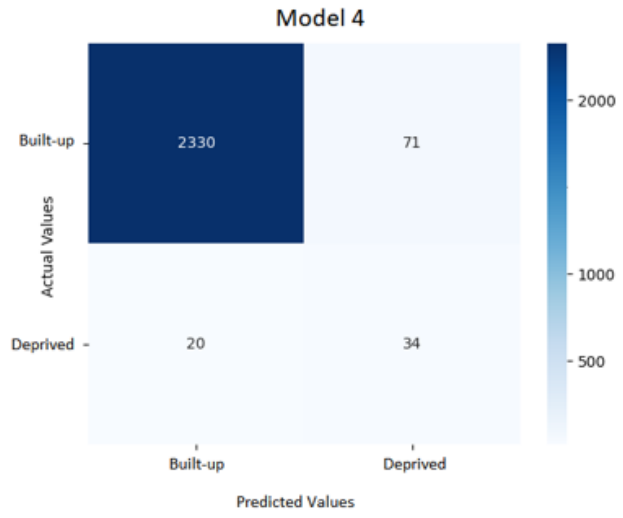


Figure 16: Confusion matrix of the best performing model

9.4 MLP using autoencoder generated images Results

In these models we use MLP with images generated using autoencoder, all models were trained for 200 epochs with early stopping and the following table shows the summary of the best performing MLP models:

Model	Training Acc	Training loss	Val. Acc	Val. Loss	Test Acc	Precision "1"	Recall "1"	f1 score "1"	MacroAvg Precision	MacroAvg Recall	MacroAvg F1 score
Model-1	**0.968**	**0.0909**	**0.828**	**0.9607**	**0.82**	**0.07**	**0.61**	**0.13**	**0.53**	**0.72**	**0.52**
Model-2	**0.973**	**0.0743**	**0.8222**	**0.2273**	**0.82**	**0.06**	**0.54**	**0.11**	**0.53**	**0.68**	**0.51**
Model-3	**0.966**	**0.0949**	**0.8575**	**0.4424**	**0.91**	**0.09**	**0.65**	**0.14**	**0.54**	**0.62**	**0.55**
Model-4	**0.9697**	**0.0775**	**0.5358**	**0.1117**	**0.86**	**0.1**	**0.67**	**0.17**	**0.55**	**0.77**	**0.55**

9.5 Grid Search Results

All models were trained using Adagrad optimization. Models are ordered from best to least F1 macro score. It was noticed that increasing the number of neurons, decreasing learning rate , and increasing number of epochs during training generally resulted in better performance. All models were trained on reduced cloud images extracted from Google Earth.

Mean Test F1 Macro Score	Parameters
0.875137	{'activation': 'relu', 'dropout': 0.2, 'epochs': 40, 'lr': 0.0005, 'n_neurons': (50, 300, 512, 512, 200)}
0.872576	{'activation': 'relu', 'dropout': 0.2, 'epochs': 40, 'lr': 0.0003, 'n_neurons': (50, 300, 512, 512, 200)}
0.86989	{'activation': 'relu', 'dropout': 0.2, 'epochs': 40, 'lr': 0.0005, 'n_neurons': (50, 300, 300, 200, 100)}
0.857536	{'activation': 'relu', 'dropout': 0.2, 'epochs': 40, 'lr': 0.0003, 'n_neurons': (50, 300, 300, 200, 100)}
0.852301	{'activation': 'relu', 'dropout': 0.2, 'epochs': 30, 'lr': 0.0005, 'n_neurons': (50, 300, 300, 200, 100)}
0.850049	{'activation': 'relu', 'dropout': 0.1, 'epochs': 30, 'lr': 0.0005, 'n_neurons': (50, 300, 300, 200, 100)}
0.843803	{'activation': 'relu', 'dropout': 0.2, 'epochs': 30, 'lr': 0.0003, 'n_neurons': (50, 300, 300, 200, 100)}
0.839305	{'activation': 'relu', 'dropout': 0.2, 'epochs': 30, 'lr': 0.0005, 'n_neurons': (50, 200, 200, 100, 100)}
0.837314	{'activation': 'relu', 'dropout': 0.1, 'epochs': 30, 'lr': 0.0003, 'n_neurons': (50, 300, 300, 200, 100)}
0.832851	{'activation': 'relu', 'dropout': 0.1, 'epochs': 30, 'lr': 0.0003, 'n_neurons': (50, 200, 200, 100, 100)}
0.831994	{'activation': 'relu', 'dropout': 0.2, 'epochs': 30, 'lr': 0.0002, 'n_neurons': (50, 300, 300, 200, 100)}
0.830989	{'activation': 'relu', 'dropout': 0.1, 'epochs': 30, 'lr': 0.0002, 'n_neurons': (50, 300, 300, 200, 100)}
0.830609	{'activation': 'relu', 'dropout': 0.1, 'epochs': 30, 'lr': 0.0005, 'n_neurons': (50, 200, 200, 100, 100)}
0.825973	{'activation': 'relu', 'dropout': 0.2, 'epochs': 30, 'lr': 0.0003, 'n_neurons': (50, 200, 200, 100, 100)}
0.818087	{'activation': 'relu', 'dropout': 0.1, 'epochs': 30, 'lr': 0.0002, 'n_neurons': (50, 200, 200, 100, 100)}
0.811465	{'activation': 'relu', 'dropout': 0.2, 'epochs': 30, 'lr': 0.0002, 'n_neurons': (50, 200, 200, 100, 100)}

The following results are based on testing the best two models obtained from the grid search on the testing data.

Parameters	Accuracy (%)	Precision (deprived)	Recall (deprived)	F1 score (deprived)
Model -1: {'activation': 'relu', 'dropout': 0.2, 'epochs': 40, 'lr': 0.0005, 'n_neurons': (50, 300, 512, 512, 200)}	88.39	0.124	0.70	0.21
Model -2: {'activation': 'relu', 'dropout': 0.2, 'epochs': 40, 'lr': 0.0005, 'n_neurons': (50, 300, 300, 200, 100)}	85	0.1	0.72	0.18

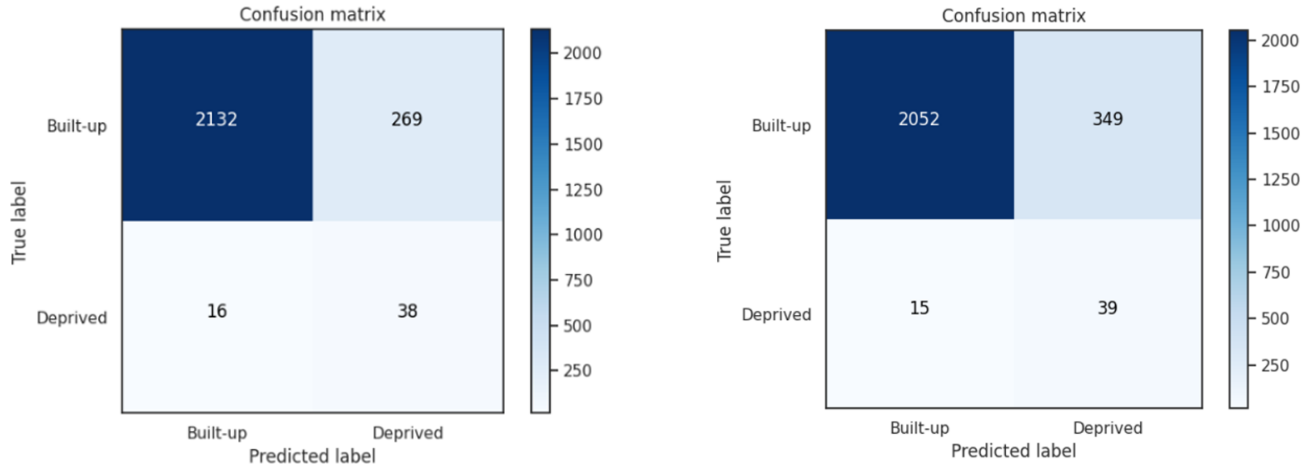


Figure 17: Confusion matrix Grid search Model -1 and Model -2

9.6 Google Open Building Results

Using Google Open Building images on the same CNN structure was providing acceptable results but not significantly higher than satellite raw image classification

Model -1	Using Google Open Building images on same CNN architecture used for satellite images
Model -2	Using pre-trained VGG16 with last 5 layers enabled for training

The following are the results observed in the training:

	Testing Performance						
Model	Test Acc	Precision "1"	Recall "1"	f1 score "1"	MacroAvg Precision	MacroAvg Recall	MacroAvg F1 score
VGG16	0.92	0.22	0.70	0.33	0.60	0.82	0.65
Standalone	0.96	0.20	0.74	0.31	0.59	0.83	0.63

9.7 Ensemble Model Results

Using a hard voting algorithms, the top seven performing models:

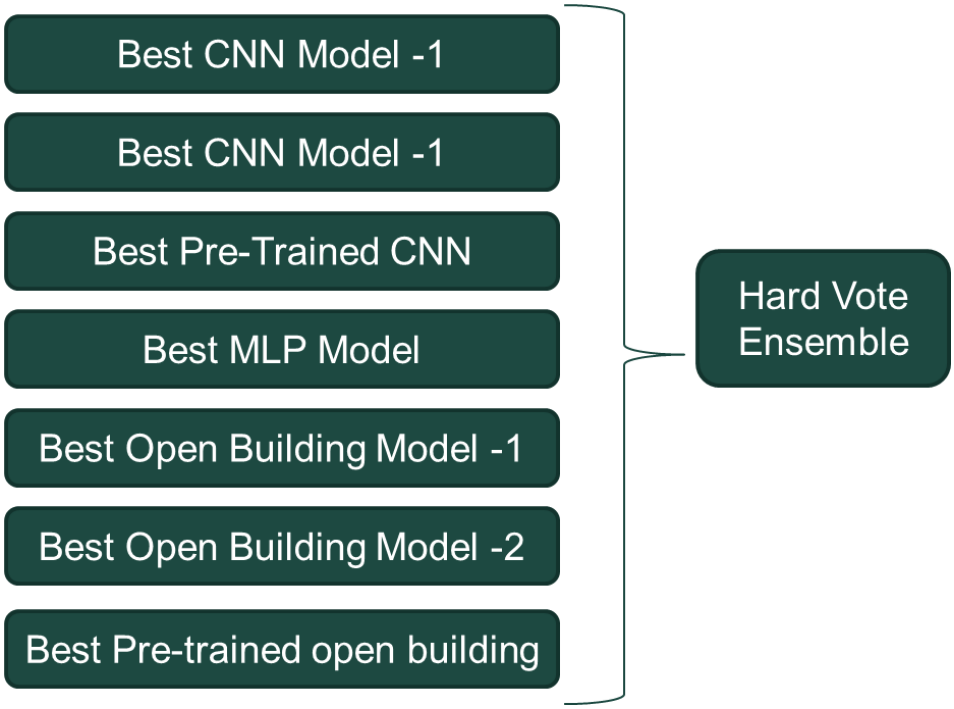


Figure 18: Hard voting ensemble

Test data was processed to ensure that same labels and data are used across the different models (i.e. satellite image for label 1 matches the open building data for the same label).

The following were the best results achieved after trying different models:

Model	Test Acc	Precision "1"	Recall " 1"	f1 score "1"	MacroAvg Precision	MacroAvg Recall	MacroAvg F1 score
VGG16	0.97	0.45	0.80	0.58	0.72	0.89	0.78

9.8 Hybrid Model Results

Parameters	Accuracy (%)	Precision (deprived)	Recall (deprived)	F1 score (deprived)
Model -1: {'batch_size : 64', epochs': 37, 'optimizer':'adamax'}	84	0.12	0.85	0.21

Raw-CNN:{‘Conv2D’ : (100, 120, 150) , 'activation': 'relu’, ‘Dense_n_neurons: (100)}’ OB_CNN: { ‘Conv2D’ : (100, 150, 200,200) , 'activation': 'relu’, ‘Dense_n_neurons: (100)}’ MLP : ('n_neurons': (20,15,10), 'activation': 'relu’} MLP_Concatenated : ('n_neurons': (210,50,25, 10), 'activation': 'relu’}				
Model -2: {'batch_size : 64', epochs': 17, ‘optimizer’:’adam’, ‘lr’ :’0.001’} Raw-CNN:{‘Conv2D’ : (100, 150, 200) , 'activation': 'relu’, ‘Dense_n_neurons: (100)}’ OB_CNN: { ‘Conv2D’ : (100, 128, 256,256, 512) , 'activation': 'relu’, ‘Dense_n_neurons: (150)}’ MLP : ('n_neurons': (30,20,10), 'activation': 'relu’, ‘dropout’: ‘0.2’} MLP_Concatenated : ('n_neurons': (260,100,50, 20), 'activation': 'relu’, ‘dropout’: ‘0.2’}	85	0.12	0.79	0.22
Model -3: {'batch_size : 64', epochs': 25, ‘optimizer’:’adam’, ‘lr’ :’0.001’} Raw-CNN:{‘Conv2D’ : (100, 150, 200) , 'activation': 'relu’, ‘Dense_n_neurons: (100, 50)}’ OB_CNN: { ‘Conv2D’ : (100, 128, 128,256, 256, 512) , 'activation': 'relu’, ‘Dense_n_neurons: (100,50)}’ MLP : ('n_neurons': (30,20,10), 'activation': 'relu’, ‘dropout’: ‘0.2’} MLP_Concatenated : ('n_neurons': (110,50,25, 10), 'activation': 'relu’, ‘dropout’: ‘0.2’}	88	0.14	0.74	0.24

10 Discussion

The core challenge with this project and other similar remote sensing researches is the following constraints:

- The timely and expensive process to label areas based on agreed definition and framework
- Availability of very High-resolution images without licensing or other royalties
- The computational cost of Deep Learning especially when adding more labels and high resolution images
- Handling the mapping ethically without genarlization absolute labeling

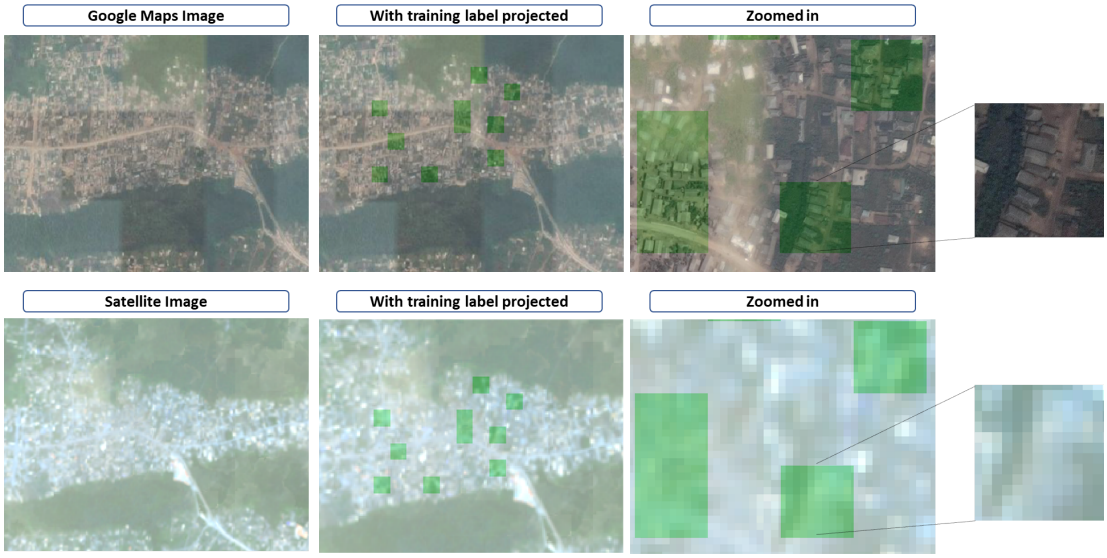


Figure 14: Difference between Very High-Resolution and the images used in this report

Keeping the above constraints in mind and evaluating the best tradeoff between the different approaches (i.e. losing details in low resolution images vs having faster computation and royalty free assets), the formula that is believed to work is to utilize the models achieved in this project and enhance the overall performance via other techniques.

An example of such tradeoff followed in this project was to avoid using the “non-built-up” class all together in the Deep Learning models and re-frame the problem as a binary classification and rely on other techniques (i.e. OpenStreetMap keys to identify such areas).

11 Conclusion

With cloud-free images available from open sources, this study initiated a framework to leverage the best modeling techniques and combinations to reach an acceptable performance on a 10x10 m2 area mapping. To reach acceptable results, several Deep Learning training techniques were examined to reach an optimum ensemble model ready to integrate with other datasets and models to enhance the classification decision.

Image sources	Sentinel -2
Total deprived labeled images	Average of 270 labeled areas for three major cities in Africa
Image size for classification	10x10 pixels covering 100 square meters
Best Performance reached	Macro AVG F1 score of 0.78

12 References and Bibliography

[1] DEEP LEARNING: FROM REMOTELY SENSED DATA TO GEO-SPATIAL SEMANTIC INFORMATION: https://grssturkey.org/assets/files/uygu2021/claudio_persello.pdf

[2] https://ideamapsnetwork.org/slum-modeller-needs.

[3] Machine Learning Approaches for Slum Detection Using Very High Resolution Satellite Images, https://nkaza.github.io/files/pdfs/slumdetection.pdf

[4] Identifying a Slums' Degree of Deprivation from VHR Images Using Convolutional Neural Networks,https://www.mdpi.com/2072-4292/11/11/1282

[5] https://bdtechtalks.com/2020/01/06/convolutional-neural-networks-cnn-convnets/

[6] CONTINENTAL-SCALE BUILDING DETECTION FROM HIGH RESOLUTION SATELLITE IMAGERY, https://arxiv.org/pdf/2107.12283.pdf

[7] https://sites.research.google/open-buildings/

Elena Ranguelova, Berend Weel, Debraj Roy, Monika Kuffer, Karin Pfeffer & Michael Lees (2019) Image based classification of slums, built-up and non-built-up areas in Kalyan and Bangalore, India, European Journal of Remote Sensing, 52:sup1, 40-61, DOI: 10.1080/22797254.2018.1535838

Michael Wurm, Thomas Stark, Xiao Xiang Zhu, Matthias Weigand, Hannes Taubenböck,Semantic segmentation of slums in satellite images using transfer learning on fully convolutional neural networks,ISPRS Journal of Photogrammetry and Remote Sensing, Volume 150, 2019, ISSN 0924-2716, https://doi.org/10.1016/j.isprsjprs.2019.02.006

Boris Babenko, Jonathan Hersh, David Newhouse, Anusha Ramakrishnan, Tom Swartz, Poverty Mapping Using Convolutional Neural Networks Trained on High and Medium Resolution Satellite Images, With an Application in Mexico, https://arxiv.org/pdf/1711.06323.pdf

Christina Corbane, Vasileios Syrris, Filip Sabo, Panagiotis Politis, Michele Melchiorri, Martino Pesaresi, Pierre Soille & Thomas Kemper , Convolutional neural networks for global human settlements mapping from Sentinel-2 satellite imagery, https://link.springer.com/article/10.1007/s00521-020-05449-7

Nicholus Mboga, Claudio Persello, John Ray Bergado and Alfred Stein, Detection of Informal Settlements from VHR Images Using Convolutional Neural Networks

Aaron E. Maxwell , Timothy A. Warner and Luis Andrés Guillén: Accuracy Assessment in Convolutional Neural Network-Based Deep Learning Remote Sensing Studies—Part 1: Literature Review