Data Science Capstone (Spring 2022)

MAPPING DEPRIVED AREAS IN LOW- AND MIDDLE-INCOME COUNTRIES (LMIC) USING SATELLITE IMAGERY

Prepared by:
Arathi Nair

Academic Advisor:
Dr. Amir Jafari

Abstract

Majority of populations in low- and middle-income countries (LMIC) live in deprived neighborhoods characterized by poverty and substandard living conditions. The policy efforts of local-to-international stakeholders to provide sustainable development requires accurate and scalable methods to map deprived areas across LMIC cities. Therefore, measuring poverty levels for efficient allotment of financial aids and resources becomes an important step to help poverty reduction. Deep learning techniques in remote sensing can be beneficial in understanding underlying physical structure present in geospatial data. However, the data collected by conducting surveys for mapping remote or developing cities not only require huge financial investments but can also be time and labor intensive. To overcome this challenge, we attempt to utilize open-source geospatial data and satellite images that will help us bridge the data gap and provide us adequate date for mapping deprived cost-effectively. In this project, we propose a machine learning framework based on deep neural network using autoencoders that will enable us to identify and classify deprived neighborhoods in Lagos - Nigeria, Accra - Ghana and Nairobi - Kenya. The model will be trained to learn from geographical features extracted from the unlabeled satellite imagery and determine transferability of model's learnings to predict and map deprived areas in other LMIC cities using transfer learning. Furthermore, we aim to build a model deployment pipeline that can be adapted to map poverty at a global scale.

Contents

1. Introduction

Low- and middle-income countries (LMIC) make up a large share of the world's population with a diverse group by size, population, and income level. According to a report from the United Nations (UN), the global population will reach 9.7 billion by 2050, and 68% of population will live in urban areas. The rapid growth in population will undoubtedly result in socioeconomic inequalities in developing countries. In most cases, these cities lack basic infrastructure and substandard of living due to rapid growth of slum-like communities. Urban sustainable development is an increasing concern worldwide, and global institutions like United Nations and World Bank are centering their Sustainable Development Goals to tackle the urbanization challenges in these countries.

Sustainable Development Goal 1 ('End poverty in all its forms everywhere') and Sustainable Development Goal 11 ('Make cities and human settlements inclusive, safe, resilient and sustainable') aims to enable LMIC cities to tackle poverty and better plan for its future population. Decision makers use neighborhood deprivation maps to estimate number of people living in these areas, allocate funds, plan, and evaluate policies to make long term development decisions.

Deprived areas are generally defined as settled urban space that lack physical or social assets which support healthy, safe, productive living. In this paper, we implement an unsupervised deep learning algorithm using Convolutional Neural Network (CNN) and Multilayer Perceptron (MLP) autoencoders that will perform image classification on remote sensing data using Google Earth satellite images. The satellite images are extracted and preprocessed from Google Earth Engine in Sentinel-2 (100m) spatial resolution for three cities: Lagos - Nigeria, Accra - Ghana and Nairobi - Kenya. Geographical data being highly unstructured, the autoencoder will extract important learning features to estimate poverty for these cities. Leveraging deep neural network architecture, we attempt to build a transfer learning framework using pure satellite images for poverty prediction as an alternate approach to the expensive surveys currently implemented.

2. Problem Statement

The proposed machine learning model aims to address three key objectives.
  i. Extraction of cloud free Sentinel-2 Satellite images from Google Earth Engine.
 ii. Implementation of a Convolutional Neural Network (CNN) and a Multilayer Perceptron (MLP) autoencoder for feature selection and deep learning.
iii. Finally, the model is tested on the classify images in the existing labeled training dataset into two classes (0 - built-up area, 1 - deprived).

3. Solution and Methodology

3.1 Sentinel -2 Cloud Free Image Extraction

To integrate satellite images in our machine learning model, sourcing cloud free satellite images from Google Earth Engine was one of the main challenges in our data preparation for creating an accurate analysis of the geographical features of these cities. We have implemented Sentinel Hub S2 cloudless machine learning algorithm to detect and mask the clouds/cloud shadows reflected on the earth's surface. The cloud mask algorithm uses two image collections from Google Earth Engine; Sentinel-2 (S2) surface reflectance (SR) collection is used to selected satellite images from a particular date range and Sentinel-2 cloud probability collection is used to identity clouds and shadows projected by low-reflectance near-infrared (NIR) pixels. The figure below shows the framework of the Sentinel-2 cloud masking algorithm for satellite image extraction.
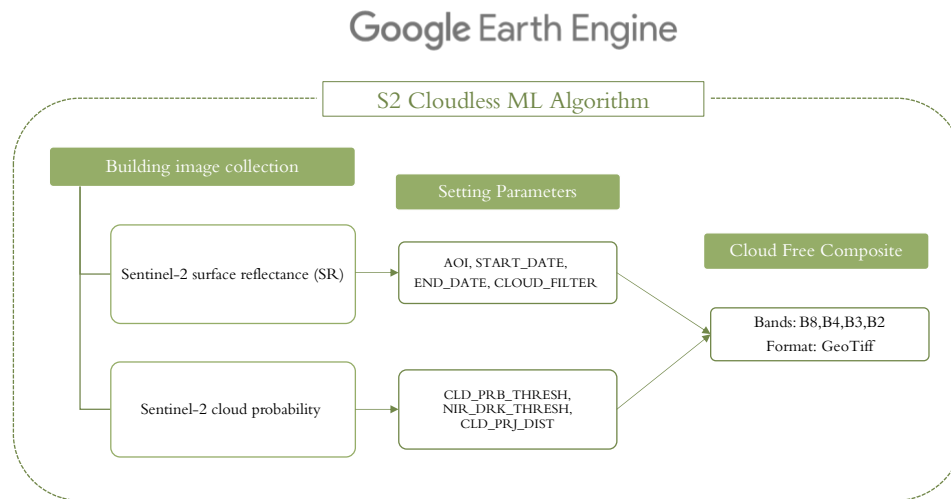
The parameters defined to filter the images from the data collection to determine cloud and cloud shadows were as follows.

Parameter	Type	Description
AOI	ee.Geometry	Area of interest
START_DATE	string	Image collection start date (inclusive)
END_DATE	string	Image collection end date (exclusive)
CLOUD_FILTER	integer	Maximum image cloud cover percent allowed in image collection
CLD_PRB_THRESH	integer	Cloud probability (%); values greater than are considered cloud
NIR_DRK_THRESH	float	Near-infrared reflectance; values less than are considered potential cloud shadow
CLD_PRJ_DIST	float	Maximum distance (km) to search for cloud shadows from cloud edges
BUFFER	integer	Distance (m) to dilate the edge of cloud-identified objects

The Sentinel-2 collection was built by selecting area of interest (AOI) i.e., geographical coordinates to create area boundary and date parameters to capture s2cloudless images for each city. To add cloud probability layer on the derived images, cloud mask components were set to identify water pixels in the image by adding Scene Classification Layer (SCL) band and NIR_DRK_THRESH was fine tuned to determine the projection of cloud shadows in the images. The tables below show the optimal parameters that were set for cloud-shadow masking for all three cities.

- **Lagos**

Surface Reflectance Collection	Cloud Probability Collection
START_DATE = '2021-12-20' END_DATE = '2022-01-23' CLOUD_FILTER = 60	START_DATE = '2021-12-20' END_DATE = '2022-01-23' CLOUD_FILTER = 60
CLD_PRB_THRESH = 65	CLD_PRB_THRESH = 60
NIR_DRK_THRESH = 0.80	NIR_DRK_THRESH = 1
CLD_PRJ_DIST = 2	CLD_PRJ_DIST = 6
BUFFER = 100	BUFFER = 100

- **Accra**

Surface Reflectance Collection	Cloud Probability Collection
START_DATE = '2021-11-14' END_DATE = '2021-12-23' CLOUD_FILTER = 30	START_DATE = '2021-11-14' END_DATE = '2021-12-23' CLOUD_FILTER = 30
CLD_PRB_THRESH = 85	CLD_PRB_THRESH = 80
NIR_DRK_THRESH = 0.50	NIR_DRK_THRESH = 1
CLD_PRJ_DIST = 2	CLD_PRJ_DIST = 2
BUFFER = 65	BUFFER = 90

- **Nairobi**

Surface Reflectance Collection	Cloud Probability Collection
START_DATE = '2022-01-03' END_DATE = '2022-01-14' CLOUD_FILTER = 60	START_DATE = '2022-01-03' END_DATE = '2022-01-14' CLOUD_FILTER = 60
CLD_PRB_THRESH = 50	CLD_PRB_THRESH = 50
NIR_DRK_THRESH = 0.15	NIR_DRK_THRESH = 0.15
CLD_PRJ_DIST = 1	CLD_PRJ_DIST = 3
BUFFER = 50	BUFFER = 70

This cloud detection algorithm detects the likelihood of clouds and shadows present in the given image collection and then mask the cloud pixels to replace them with cloud-free pixels from a different timeframe. Lastly, all the mosaics were merged for obtaining a single cloud-free image. Shown below are the results of the cloud free images that will be used for training our autoencoder models for feature learning.
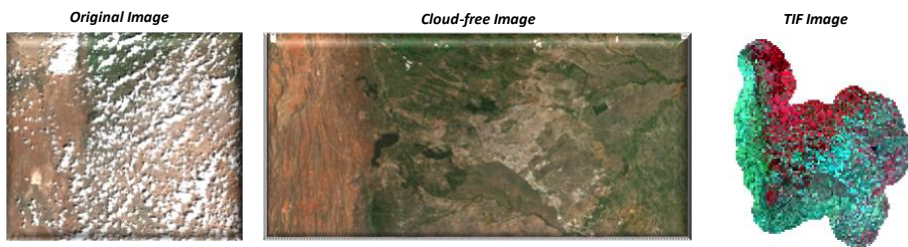
- **Lagos**

Original Image   *Cloud-free Image*   *TIF Image*

- **Accra**

Original Image   *Cloud-free Image*   *TIF Image*

- **Nairobi**

Original Image   *Cloud-free Image*   *TIF Image*

3.2 Image Clipping, Conversion and Normalization

Post extracting the clear satellite images from Google Earth, the TIF images were clipped to (10 x 10 pixels) 100m spatial images. A total of 1030089 images were extracted from all the three cities for training the CNN & MLP autoencoder models. All the clipped images were converted int o PNG format and normalized to fit each pixel values between 0 - 255 to match the RGB standard.

10 x 10 px clipped images - Before normalization   *10 x 10 px clipped images - After normalization*
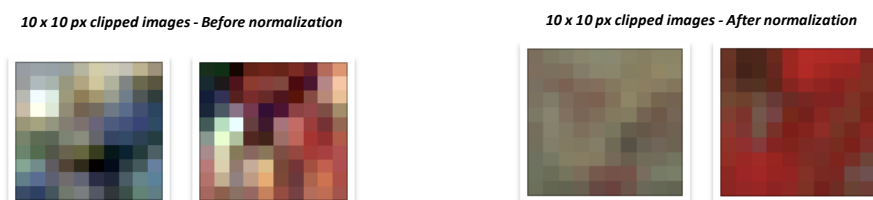
3.3 Training Autoencoder

3.3.1. Convolutional Autoencoder

The main task of the Convolutional (CNN) Autoencoder was to reconstruct the satellite images (10 x 10 pixels) inputted into the model and minimize reconstruction error by learning the optimal filters. A typical autoencoder consist of four parts: encoder, bottleneck, decoder, and reconstruction loss. The encoder reduces the high-dimensional input to the low-level code, which is called the bottleneck. From the bottleneck layer, the decoder attempts to reconstruct the data from the learned encoding. Finally, the reconstruction error (RE) measures how well the decoder performed in creating an output similar to the input. The following block diagram represents the process of image compression and reconstruction in the autoencoder.
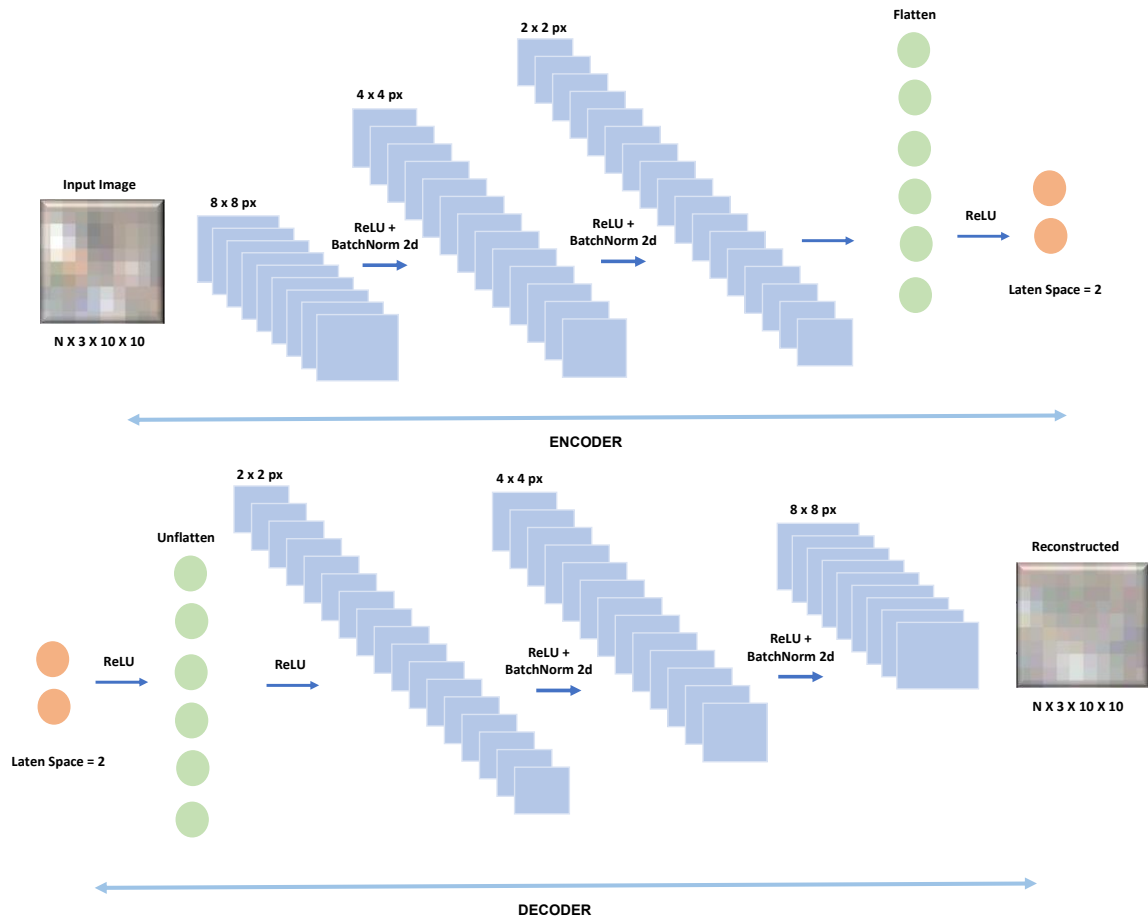
Computation of the spatial dimension of the output size is represented by a function of the input size (W), the receptive field of the convolutional layer (F), the stride applied (S) and the amount of zero padding used (P). The formula used in determining the dimension of the feature maps in convolutional neural network are as follow.

$$\text{Encoder: } (W - F + 2P) / S + 1 \tag{1}$$
$$\text{Decoder: } S * (W - 1) + F - 2P \tag{2}$$

The above CNN autoencoder model was implemented using PyTorch. The satellite images in size 10 x 10 pixels with 3 channels (RGB) were passed into the encoder to compress the high dimensional images to a smallest possible dimension, embedding the content of the original image into the latent space of 2 x 2 pixels. This latent space vector is used by the decoder to reconstruct the compressed image back into 10 x 10 pixels with the lowest possible reconstruction error (RE). To obtain the reconstruction error (RE) between the original input and reconstructed output the model is trained to minimize the loss function mean squared error (MSE). In between each convolutional layer is a rectified linear unit (ReLU) activation function conducting nonlinear transformation except for the last output layer which uses Sigmoid as an activation function.

To enforce robustness of the CNN autoencoder, it was trained as a denoised autoencoder. By adding white noise to the autoencoder we combat the tendency of the regular autoencoder to overfit and forces the model to learn important features from the satellite images. The graph below show the results obtained after training on satellite images for all three cities - Lagos, Accra and Nairobi.

3.3.2. Multilayer Perceptron Autoencoder

As an alternative, a MLP autoencoder was also trained for the image reconstruction using linear neural networks. MLP autoencoder is represented as a feed forward neural network that consists of three layers nodes - an input, a hidden and an output layer. Each layer in a MLP is fully connected with the next layer. In the hidden layers, each node is operated with a leaky rectified linear unit (LReLU) activation function that avoids vanishing gradient problem introduced by sigmoid and dying neuron problem caused by using ReLU. This enables the networks to distinguish and separate useful patterns from the satellite images. The model is trained to calculate the reconstruction error (RE) between the predicted output and the expected output values by minimizing the loss function MSE. Further to build a robust model, we have also added noise to the input image forcing the model to separate the important signals from high-dimensional data. The image below shows the final MLP autoencoder model which was implemented in PyTorch. Like the CNN autoencoder, the satellite images were reduced in dimension size from 10 x 10 px to the smallest dimension with a latent space of 2 x 2 pixels.
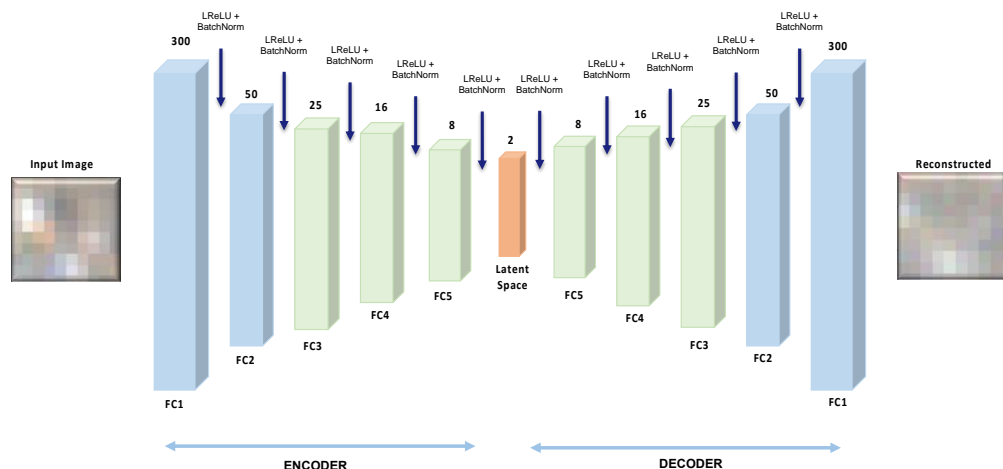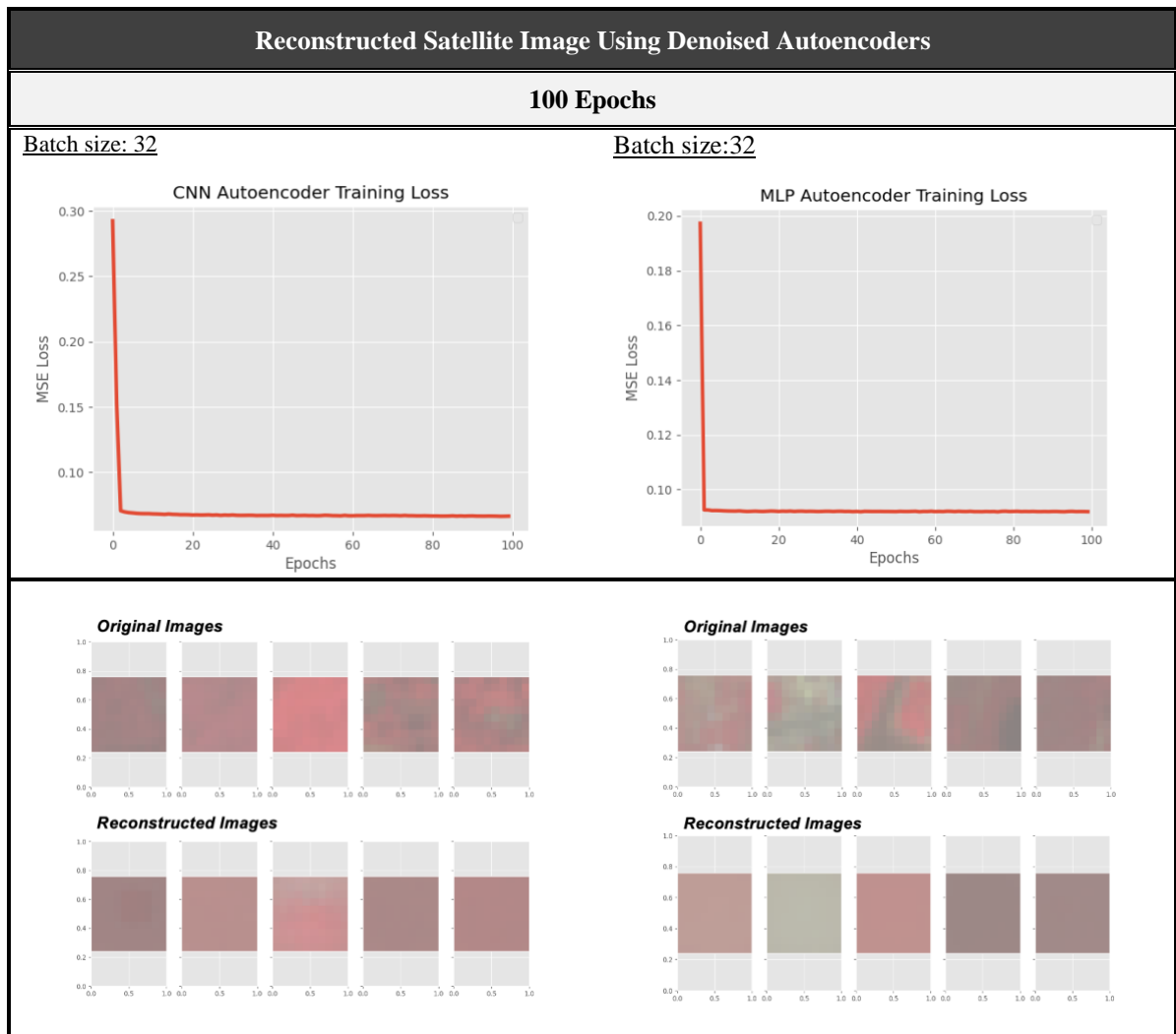
4. Results

4.1 Training Autoencoder

The graphs below show the results obtained after training the CNN & MLP autoencoders on 1030089 images in size 10 x 10 pixels for over 100 and 200 epochs. Both models were trained in batch sizes 10 and 32 using ADAM method for stochastic optimization and learning rate of 0.0001.
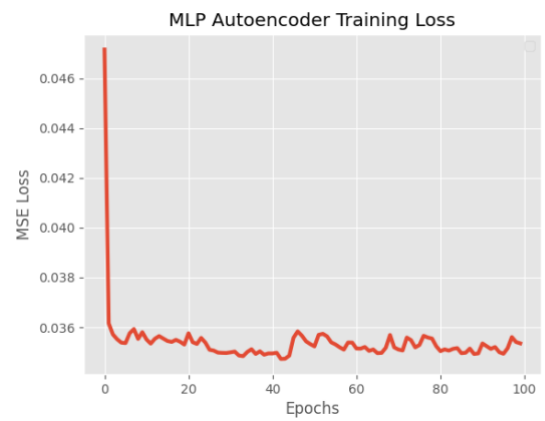
In the training loss for denoised CNN autoencoder with batch size 32, we notice sharp fall from 0.3 to 0.06 in MSE reconstruction error after running for 100 and 200 epochs. Similarly, for the denoised MLP autoencoder, we notice the reconstruction loss drops from 0.2 to 0.09 after running over 100 and 200 epochs. Whereas the MSE for batch size 10 the training loss for the denoised CNN & MLP autoencoders starts low at 0.05 and falls further to 0.02 and 0.03 respectively. Additionally, after comparing the original images with the reconstructed images for both the models we found the results produced by the autoencoder models were satisfactory.

Batch size: 10
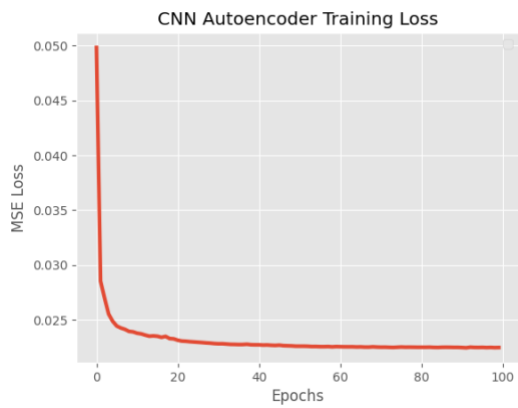
CNN Autoencoder Training Loss

MLP Autoencoder Training Loss

Original Images

Reconstructed Images

Original Images

Reconstructed Images

200 Epochs

Batch size: 32

Batch size: 32

CNN Autoencoder Training Loss

MLP Autoencoder Training Loss

Original Images

Reconstructed Images
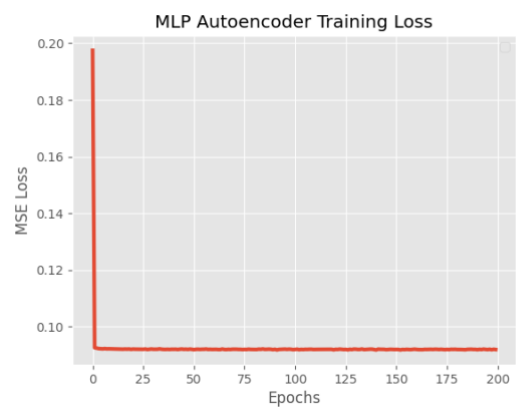
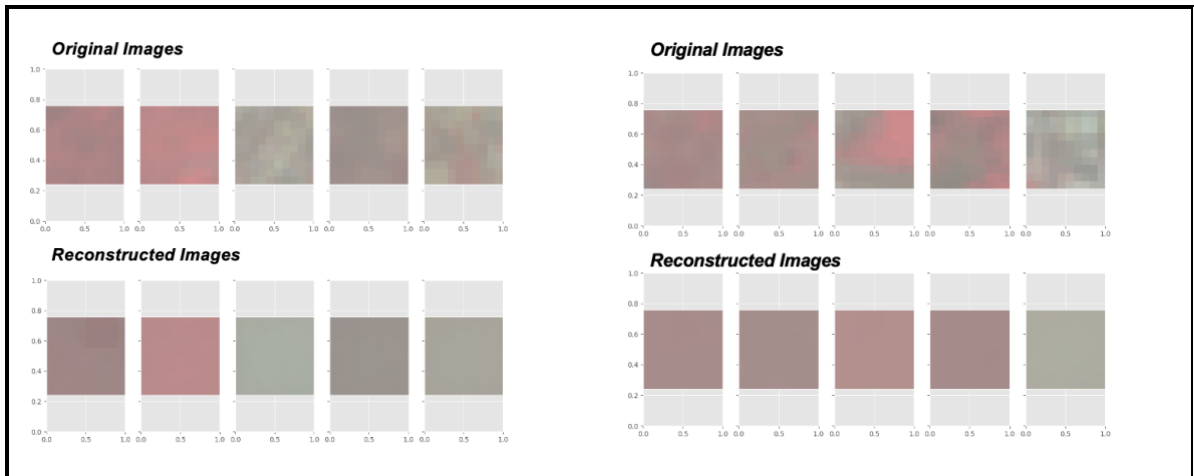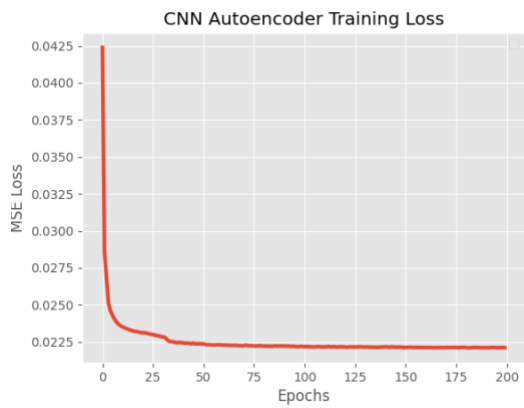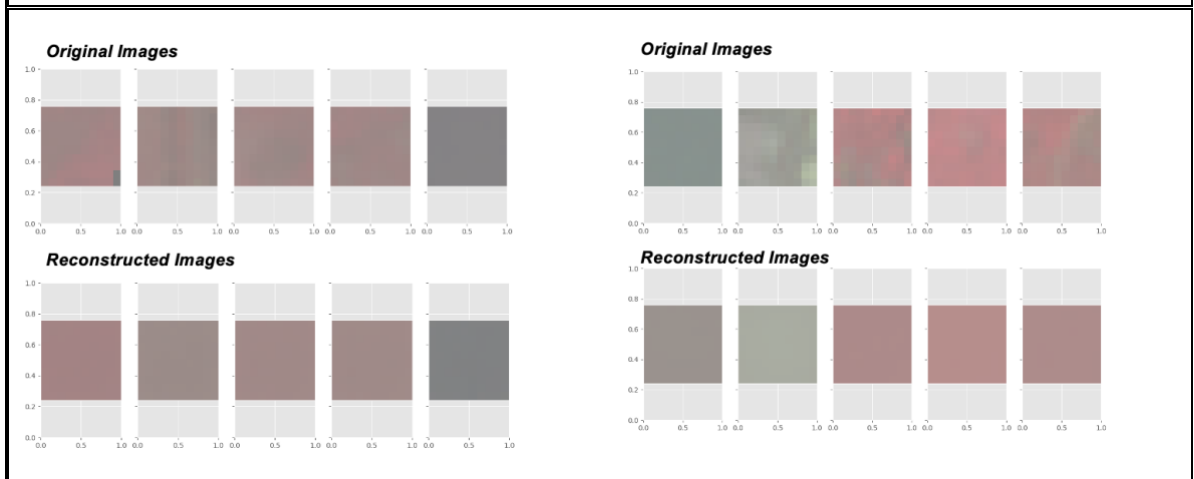**Original Images**

Reconstructed Images
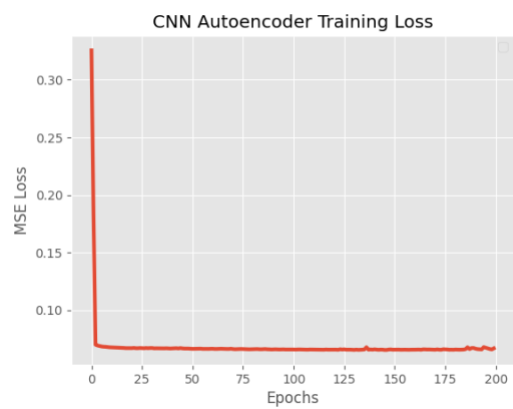
Batch size: 10

Batch size: 10

CNN Autoencoder Training Loss

MLP Autoencoder Training Loss

Original Images

Reconstructed Images

Original Images

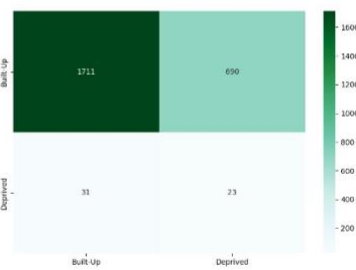**Reconstructed Images**
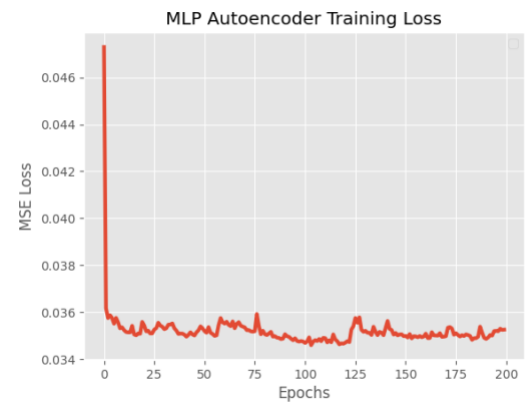
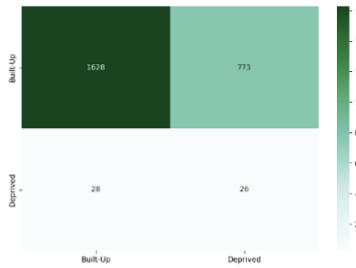## 4.2 Image Classification on Training Dataset
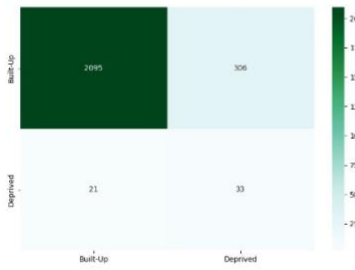
After training the CNN & MLP autoencoders, the saved encoder model weights were used to classify and predict deprived areas on the Lagos and Accra training dataset. For this analysis images with labels build-up and deprived areas were considered as label - 2 pertains to areas related to vegetation and water were excluded from this classification. The encoder models are used as a classifier along with the saved weights from pretrained autoencoders to attempt segregate the images into two categories i.e., 0 - built-up areas, 1 - deprived areas based on the learned geographical features. The training data was split into training and validation data to train the encoder model and then the accuracy of the models were determined by evaluating it on the test data.

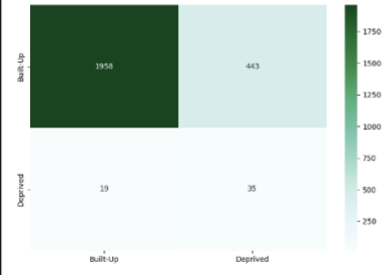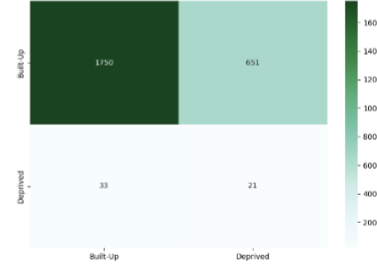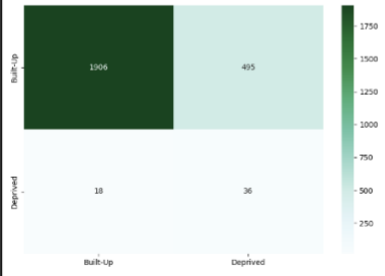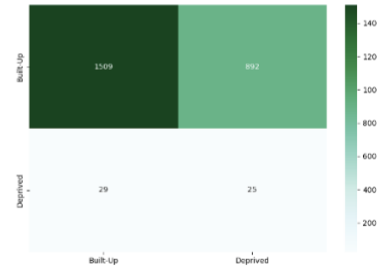## 4.2.1 Classification for Lagos Training Dataset

Presented below are the results obtained after testing the autoencoder models on the labeled training data set for Lagos. The dataset consists a total of 18674 images in the train set with 9854 images for built-up area (class - 0) and 8820 images deprived area (class - 1) for training the image classifier. Whereas the test set consists of a total of 2455 images with built-up area (class - 0) containing 2401 images and deprived area (class - 1) containing 54 images for testing the models. For improving performance, the models were also trained using ReduceLROnPlateau scheduler to reduce the learning rate in case model learning stagnates and early stopping to avoid over fitting.

To being with, we test the classification of deprived areas using per-trained CNN & MLP autoencoder models run for 100 epochs and 200 epochs using both batch sizes 32 and 10. All the models were compared based on the F1 score achieved for the deprived area (class - 1) and model's ability to classify the images in the test data by plotting the confusion matrix.
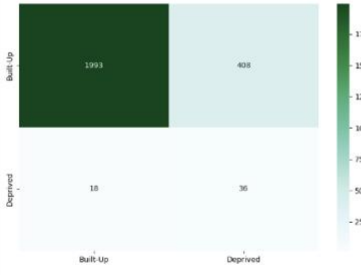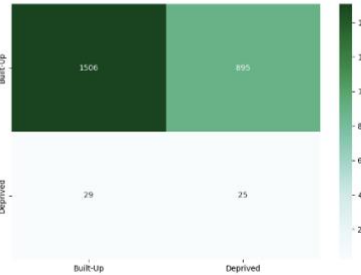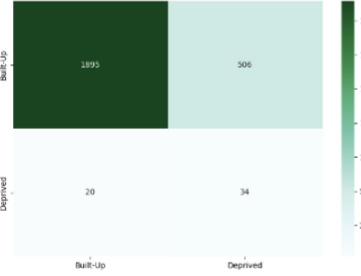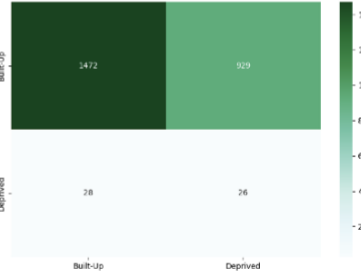
- Autoencoder model trained with batch size 32 over 100 epochs

W/O Scheduler	CNN		MLP	
	Build-up (Class – 0)	Deprived (Class – 1)	Build-up (Class – 0)	Deprived (Class – 1)
F1 Score	0.93	0.17	0.83	0.06
Precision	0.99	0.10	0.98	0.03
Recall	0.87	0.61	0.71	0.43
Accuracy	0.87		0.71	
Confusion Matrix				
Macro Avg F1	0.55		0.44	
Weighted Avg F1	0.91		0.81	
With Scheduler	CNN		MLP	
	Build-up (Class – 0)	Deprived (Class – 1)	Build-up (Class – 0)	Deprived (Class – 1)
F1 Score	0.88	0.13	0.80	0.06
Precision	0.99	0.07	0.98	0.03
Recall	0.80	0.72	0.68	0.48
Accuracy	0.79		0.67	
Confusion Matrix				
Macro Avg F1	0.51		0.43	
Weighted Avg F1	0.87		0.79	

- Autoencoder model trained with batch size 10 over 100 epochs
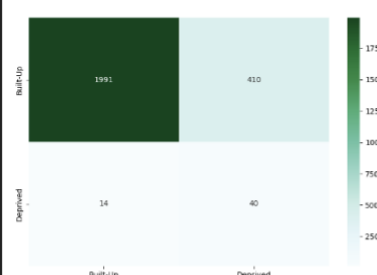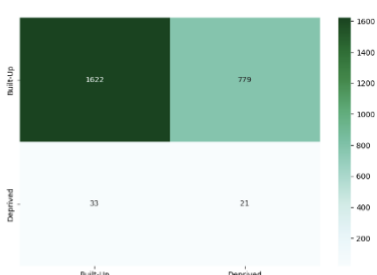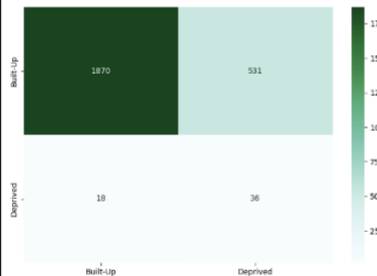
W/O Scheduler	CNN		MLP	
	Build-up (Class – 0)	Deprived (Class – 1)	Build-up (Class – 0)	Deprived (Class – 1)
F1 Score	0.89	0.13	0.84	0.06
Precision	0.99	0.07	0.98	0.03
Recall	0.82	0.65	0.73	0.39
Accuracy	0.81		0.72	
Confusion Matrix				
Macro Avg F1	0.51		0.45	
Weighted Avg F1	0.88		0.82	

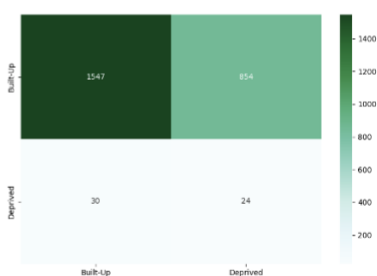| With Scheduler	CNN		MLP	
	Build-up (Class – 0)	Deprived (Class – 1)	Build-up (Class – 0)	Deprived (Class – 1)
F1 Score	0.88	0.12	0.77	0.05
Precision	0.99	0.07	0.98	0.03
Recall	0.79	0.67	0.63	0.46
Accuracy	0.79		0.62	
Confusion Matrix				
Macro Avg F1	0.50		0.41	
Weighted Avg F1	0.86		0.75	

- Autoencoder model trained with batch size 32 over 200 epochs

W/O Scheduler	CNN		MLP	
	Build-up (Class – 0)	Deprived (Class – 1)	Build-up (Class – 0)	Deprived (Class – 1)
F1 Score	0.90	0.14	0.77	0.05
Precision	0.99	0.08	0.98	0.03
Recall	0.83	0.67	0.63	0.46
Accuracy	0.83		0.62	
Confusion Matrix				
Macro Avg F1	0.52		0.41	
Weighted Avg F1	0.89		0.75	
With Scheduler	CNN		MLP	
	Build-up (Class – 0)	Deprived (Class – 1)	Build-up (Class – 0)	Deprived (Class – 1)
F1 Score	0.88	0.11	0.78	0.05
Precision	0.99	0.06	0.98	0.03
Recall	0.79	0.61	0.61	0.48
Accuracy	0.79		0.64	
Confusion Matrix				
Macro Avg F1	0.50		0.41	
Weighted Avg F1	0.86		0.76	

- Autoencoder model trained with batch size 10 over 200 epochs
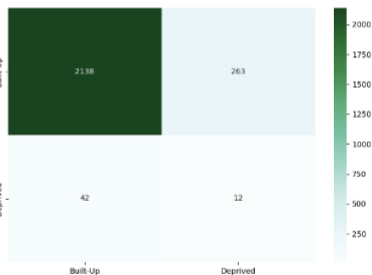
W/O Scheduler	CNN		MLP	
	Build-up (Class – 0)	Deprived (Class – 1)	Build-up (Class – 0)	Deprived (Class – 1)
F1 Score	0.90	0.16	0.80	0.05
Precision	0.99	0.09	0.98	0.39
Recall	0.83	0.74	0.68	0.39
Accuracy	0.83		0.67	
Confusion Matrix				
Macro Avg F1	0.53		0.42	
Weighted Avg F1	0.89		0.78	
With Scheduler	CNN		MLP	
	Build-up (Class – 0)	Deprived (Class – 1)	Build-up (Class – 0)	Deprived (Class – 1)
F1 Score	0.87	0.12	0.78	0.05
Precision	0.99	0.06	0.98	0.03
Recall	0.78	0.67	0.64	0.44
Accuracy	0.78		0.64	
Confusion Matrix				
Macro Avg F1	0.49		0.41	
Weighted Avg F1	0.86		0.76	

From the above models, the CNN autoencoder model trained with batch size 32 over 100 epochs gave the best results with a F1 score of 93% for built-up area (class -1) and 17% for deprived area (class - 0). On the test set, the model was able to detect 2095 out of 2401 built-up area images and 33 out of 54 deprived images accurately. Since the data for these two classes were imbalanced in the existing training dataset, additional images were added to the training dataset using data augmentation techniques such as image rolling, image shifting and oversampling.
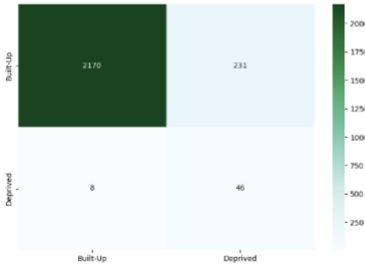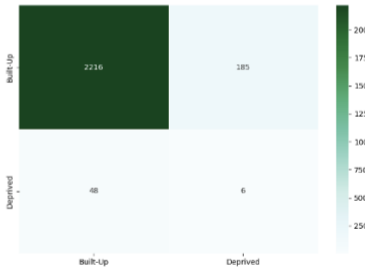
The results after including additional data to train the classifier saw a significant increase in the F1 scores of the deprived class. The CNN autoencoder model trained on 10 batch size over 100 epochs achieved the highest F1 of 28%. On the test set, the model was able to detect 43 out of 54 images accurately.

The below table shows the results obtained using pretrained CNN & MLP autoencoder models with data augmentation.

- Autoencoder model trained with batch size 32 over 100 epochs with data augmentation

W/O Scheduler	CNN		MLP	
	Build-up (Class – 0)	Deprived (Class – 1)	Build-up (Class – 0)	Deprived (Class – 1)
F1 Score	0.94	0.25	0.93	0.07
Precision	0.99	0.15	0.99	0.04
Recall	0.90	0.80	0.89	0.22
Accuracy	0.89		0.88	
Confusion Matrix				
Macro Avg F1	0.63		0.50	
Weighted Avg F1	0.78		0.91	

- Autoencoder model trained with batch size 10 over 100 epochs with data augmentation

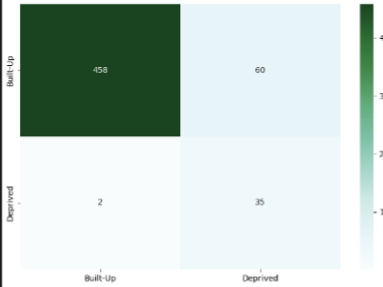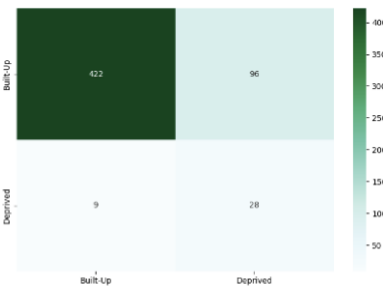| W/O Scheduler	CNN		MLP	
	Build-up (Class – 0)	Deprived (Class – 1)	Build-up (Class – 0)	Deprived (Class – 1)
F1 Score	0.95	0.28	0.95	0.05
Precision	1.00	0.17	0.98	0.03
Recall	0.90	0.85	0.92	0.11
Accuracy	0.90		0.91	
Confusion Matrix				
Macro Avg F1	0.61		0.50	
Weighted Avg F1	0.93		0.93	

Furthermore, the result obtained by applying pretrained autoencoders were compared with the classification results obtained by CNN & MLP models implemented without pretrained autoencoder. The below table shows the model performances after including data augmentation. The results using both the methods (i.e., with and without pretrained autoencoder) produced close results with a minor improved performance by the pertained CNN autoencoder with a F1 score of 28%. However, the performance of the MLP model without pretrained autoencoder provided a better F1 score of 18 %.

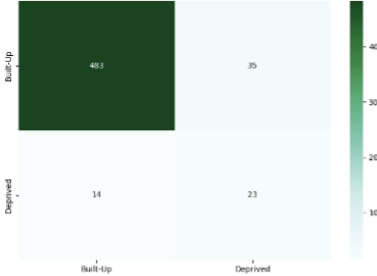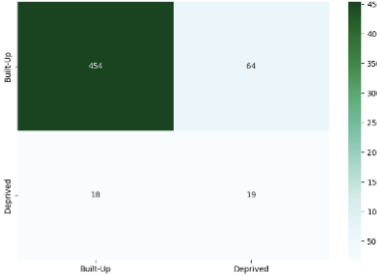| With Pretrained Autoencoder	CNN	MLP	Without Pretrained Autoencoder	CNN	MLP
	Deprived (Class – 1)	Deprived (Class – 1)		Deprived (Class – 1)	Deprived (Class – 1)
F1 Score	0.28	0.07	F1 Score	0.26	0.18
Precision	0.17	0.04	Precision	0.15	0.10
Recall	0.85	0.22	Recall	0.87	0.65
Macro Avg F1	0.61	0.50	Macro Avg F1	0.58	0.55

4.2.2 Classification for Accra Training Dataset

Similarly, the pretrained CNN & MLP autoencoder models were tested on classifying the deprived class in the Accra training dataset. As shown in the table below, the CNN model trained using batch size 32 over 100 epochs achieved the highest F1 score with 53% for detecting deprived class followed by MLP model trained using batch size 32 over 100 epochs with a F1 score of 35% for deprived class. On the test set, CNN model accurately detected 35 out of 37 images as deprived and MLP model accurately detect 28 out of 37 deprived images.

- Autoencoder model trained with batch size 32 over 100 epochs

W/O Scheduler	CNN		MLP	
	Build-up (Class – 0)	Deprived (Class – 1)	Build-up (Class – 0)	Deprived (Class – 1)
F1 Score	0.94	0.53	0.89	0.35
Precision	1.00	0.37	0.98	0.23
Recall	0.88	0.95	0.81	0.76
Accuracy	0.89		0.81	
Confusion Matrix				
Macro Avg F1	0.73		0.62	
Weighted Avg F1	0.91		0.85	

- Autoencoder model trained with batch size 10 over 100 epochs

W/O Scheduler	CNN		MLP	
	Build-up (Class – 0)	Deprived (Class – 1)	Build-up (Class – 0)	Deprived (Class – 1)
F1 Score	0.95	0.48	0.92	0.32
Precision	0.97	0.40	0.94	0.23
Recall	0.93	0.62	0.88	0.51
Accuracy	0.91		0.85	
Confusion Matrix				
Macro Avg F1	0.72		0.62	
Weighted Avg F1	0.92		0.88	

5. Conclusion

In this report, we experimented with the usage of cloud free satellite images from Google Earth Engine to create a customized autoencoder model for distinguishing deprived areas in LMIC cities like Lagos and Accra. The experimentation results for pretrained CNN autoencoder provided better results compared to the pretrained MLP autoencoder on the labeled training dataset. Although, the proposed models produced satisfactory results for predicting built-up areas; the accuracy for predicting deprived areas were low due to under representation of the deprived class in the training dataset. Additionally, it was also observed that adding more data for training aided in improving the performance of the models. This indicates there is a potential to increase model prediction accuracy by training the models on larger dataset. The challenges of extracting 100% cloud free satellite images and insufficient training data for deprived class needs to be addressed to develop a high performing transfer learning model and requires further improvements.

6. References

[1] Sentinel-2 Cloud Masking with s2cloudless - https://developers.google.com/earth-engine/tutorials/community/sentinel-2-s2cloudless

[2] Evaluating the Ability to Use Contextual Features Derived from Multi-Scale Satellite Imagery to Map Spatial Patterns of Urban Attributes and Population Distributions - https://www.mdpi.com/2072-4292/13/19/3962

[3] The Role of Earth Observation in an Integrated Deprived Area Mapping "System" for Low-to-Middle Income Countries - https://www.mdpi.com/2072-4292/12/6/982

[4] Deep Internal Learning for Inpainting of Cloud-Affected Regions in Satellite Imagery - https://www.mdpi.com/2072-4292/14/6/1342

[5] A Convolutional Autoencoder Topology for Classification in High-Dimensional Noisy Image Datasets - https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8622369/

[6] Deep Transfer Learning for Land Use and Land Cover Classification: A Comparative Study - https://www.mdpi.com/1424-8220/21/23/8083

[7] Transfer Learning from Deep Features for Remote Sensing and Poverty Mapping - https://www.researchgate.net/publication/282403466_Transfer_Learning_from_Deep_Features_for_Remote_Sensing_and_Poverty_Mapping

[8] Improving Image Autoencoder Embeddings with Perceptual Loss - https://www.researchgate.net/publication/338547072_Improving_Image_Autoencoder_Embeddings_with_Perceptual_Loss

[9] Extracting and Composing Robust Features with Denoising Autoencoders - https://www.researchgate.net/publication/221346269_Extracting_and_composing_robust_features_with_denoising_autoencoders

[10] Super Resolution for Noisy Images Using Convolutional Neural Networks

https://www.spiedigitallibrary.org/conference-proceedings-of-spie/10677/1067710/Super-resolution-for-noisy-images-via-deep-convolutional-neural-network/10.1117/12.2297664.full?SSO=1