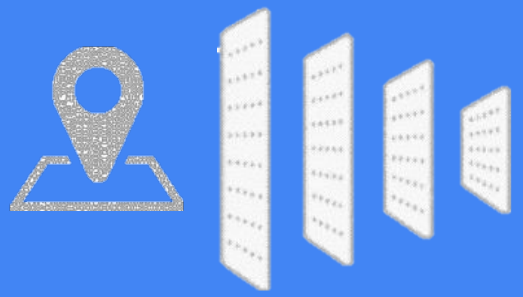
Using Deep Learning to Map Deprived Areas

Project Team:

Abdulaziz Gebril
Mina Hanna
Mojahid Osman
Arathi Nair

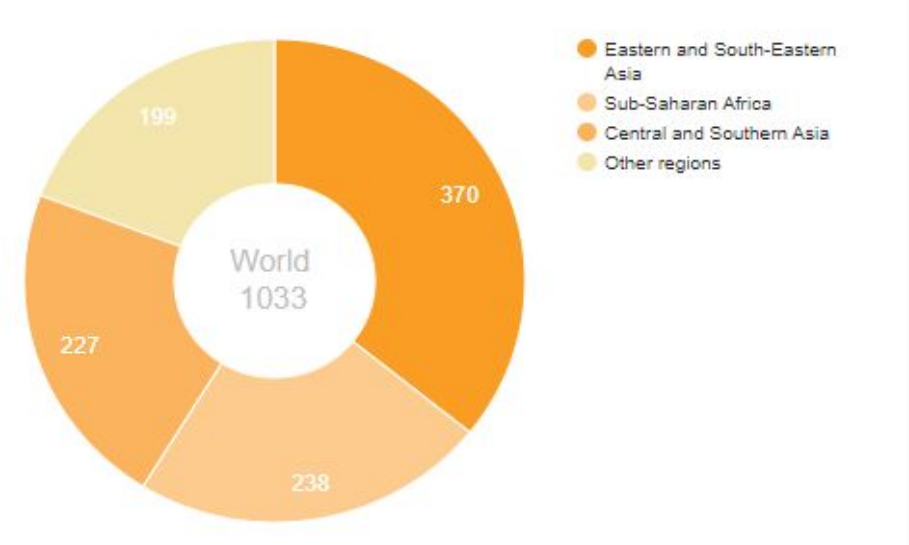
Supervised by:

Prof. Amir Jafari



The Problem

- According to the UN,[1] more than 1 billion people are living in deprived areas
- Policy makers, government and global organizations are seeking detailed identification of deprived areas to enhance assignment of resources and track progress of development projects



This project aims to process geospatial data and utilize various Computer Vision and Machine Learning techniques to help identifying deprived areas while using open source data and accessible satellite images

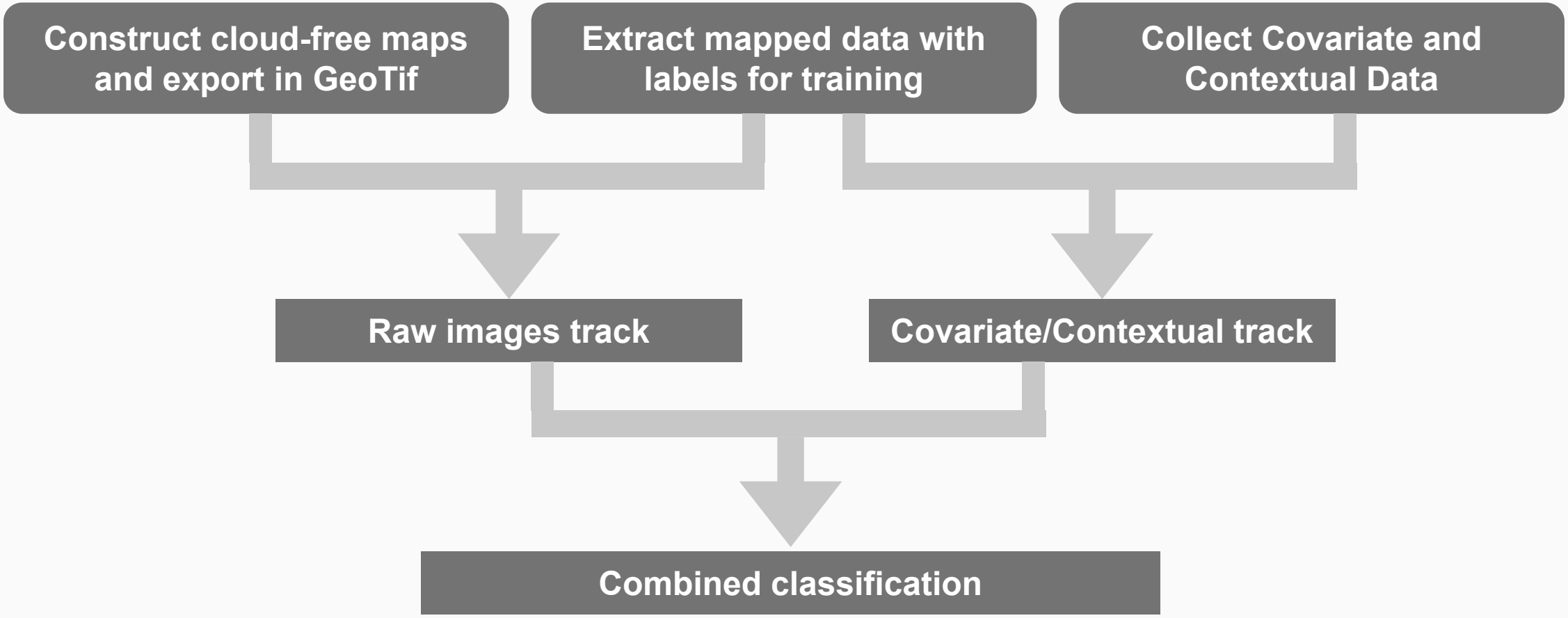
*https://unstats.un.org/sdgs/report/2019/goal-11/

Related Work

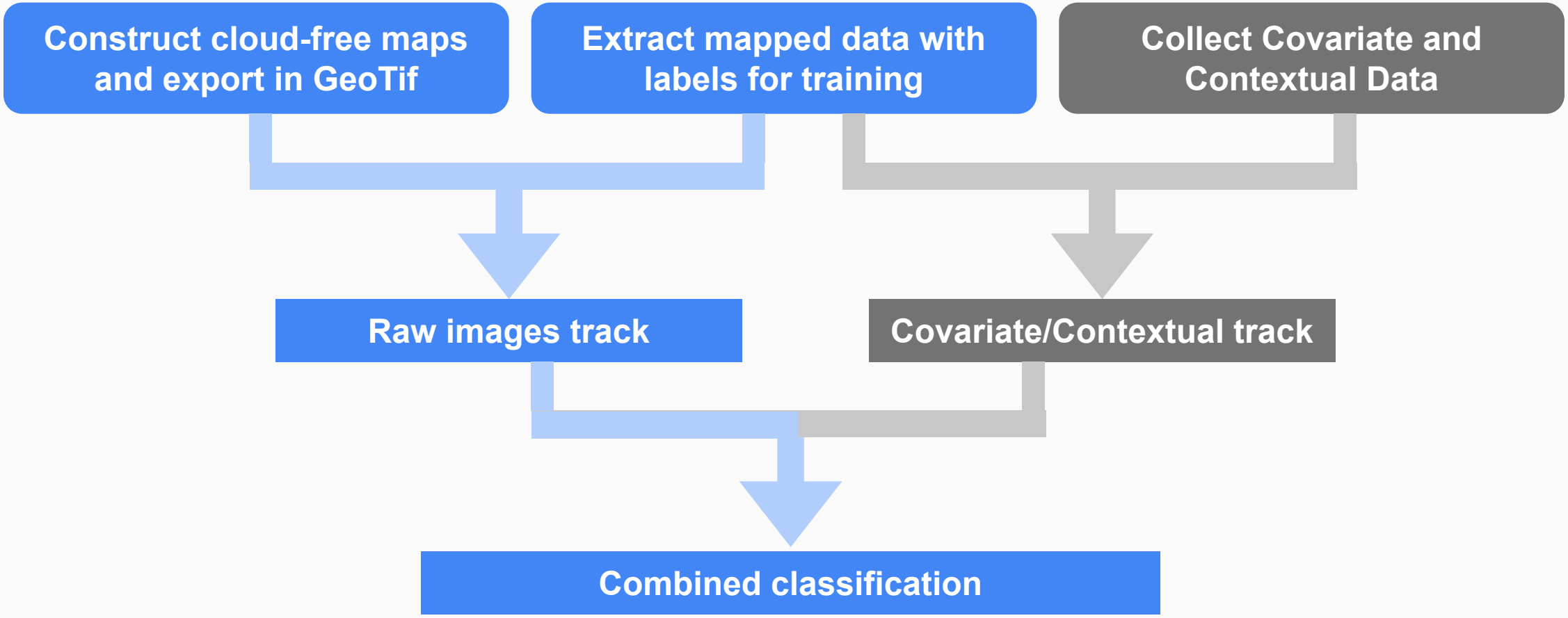
- (Krishna et al., 2018) used Very High-Resolution Satellite images and applied CNN model to have a multi-class and Hierarchical class classifier able to classify informal areas with detailed labels like single/multi-story with overall accuracy reaching 0.71 and F-1 score of 0.70 for the multi-class classification.
- (Monika et al., 2019) 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.
- (Michael et al., 2019) used transfer learning from a model trained on very high resolution imagery from QuickBird to Sentinel-2 and TerraSAR-X data. It was noticed that a significant increase for Sentinel-2 applying transfer learning can be observed (from 38 to 55% and from 79 to 85% for PPV and sensitivity, respectively).

Our approach is distinguished in utilizing low resolution satellite imagery, much smaller grid size and novel data license free datasources

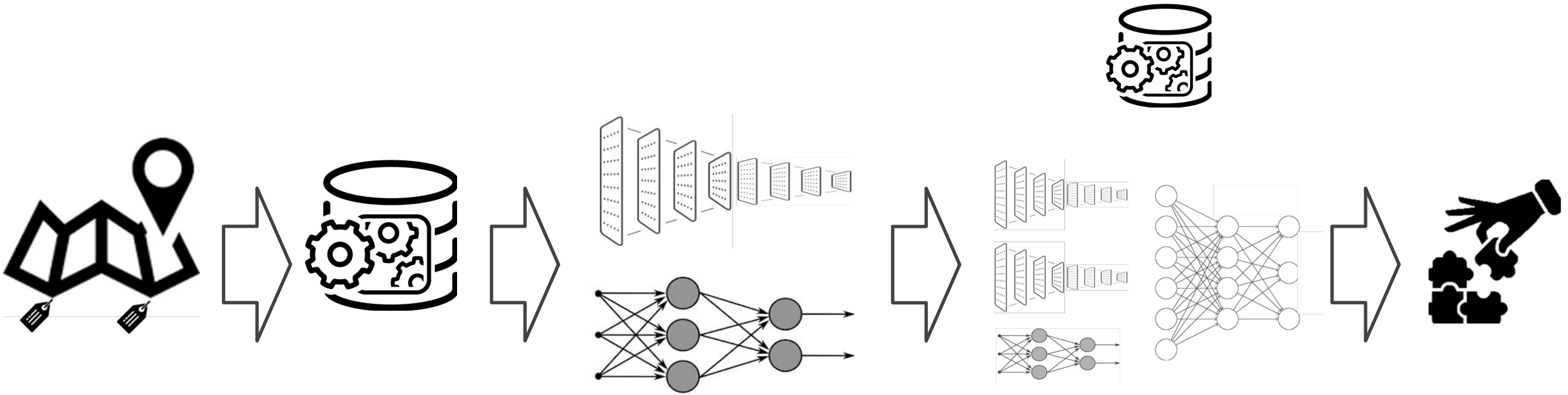
The project big picture



Focus area for this presentation

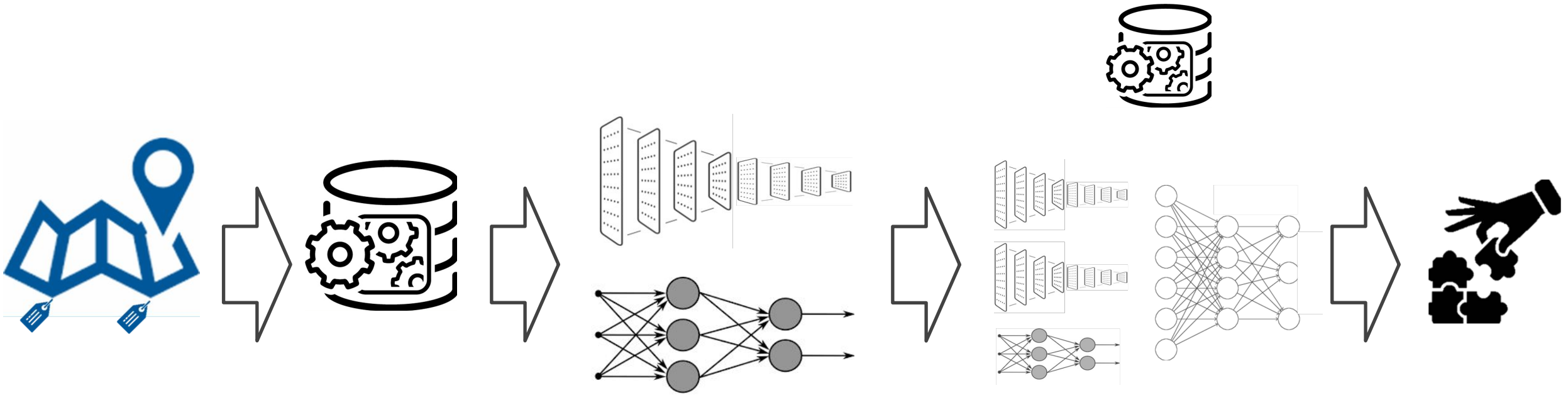


Agenda



The Data **Processing** **Modeling | Results 1** **Modeling | Results 2** **Conclusion**

Agenda

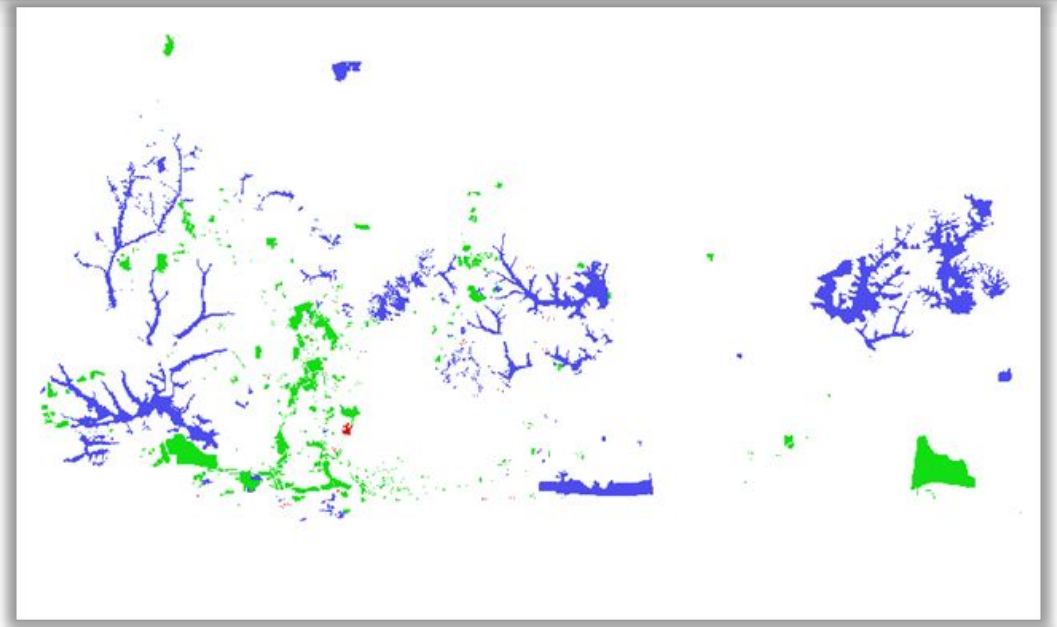


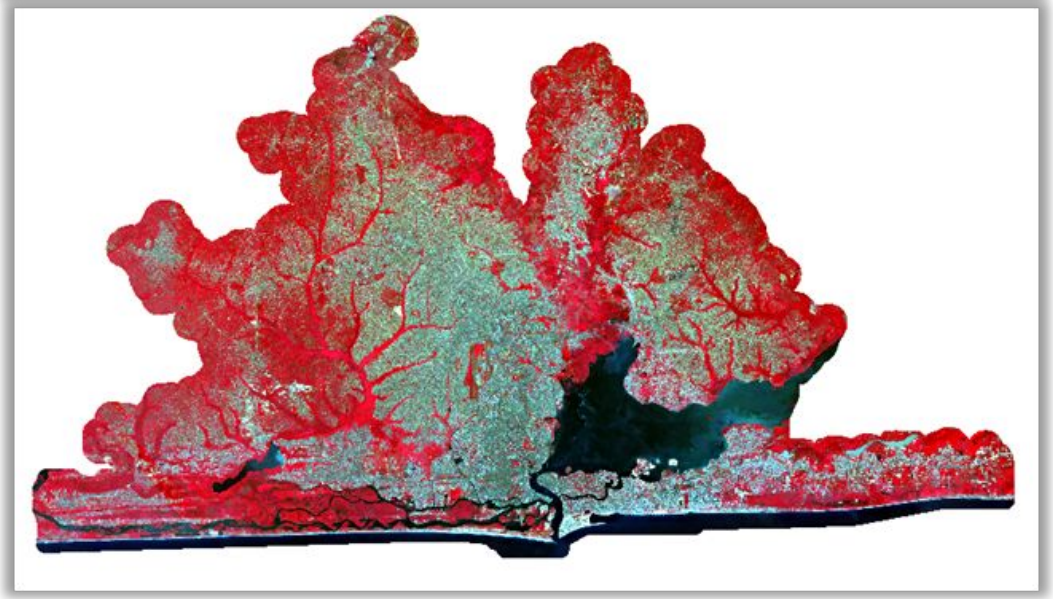
The Data | **Processing** | **Modeling | Results 1** | **Modeling | Results 2** | **Conclusion**

The Data

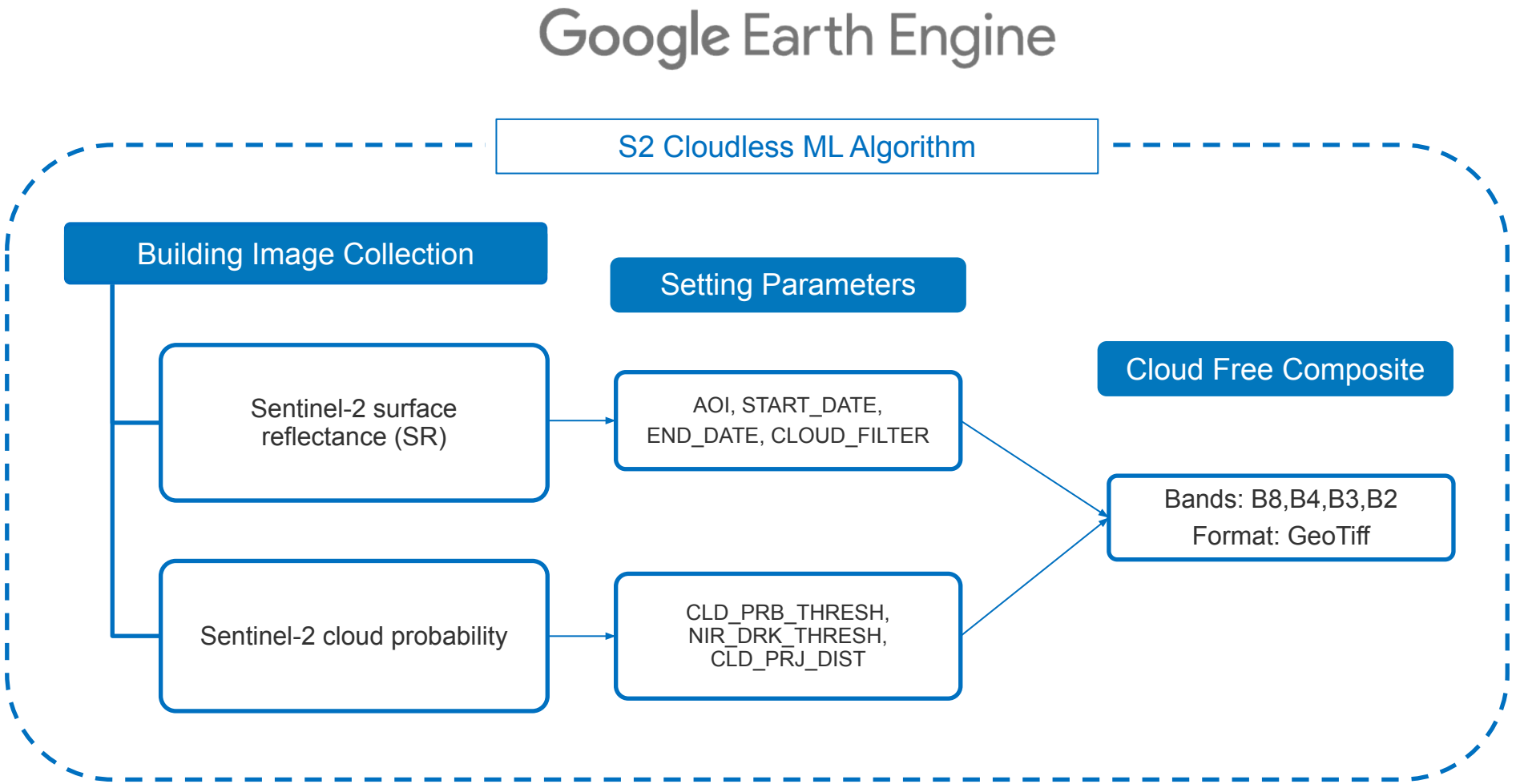
All labled data was collected with joint efforts led by Idea Maps Network. IdeaMaps network mapped 100 m2 areas to the following three labels (across multiple cities in Africa):
1- Built-up areas with label 0
2- Deprived area with label 1
3 - Non-built-up area with label 2

The data is the two following GeoTiff files:
1- Map image extracted from Google earth engine (which is another track of our project)
2- Labeled Tiff image contains labeled areas





Extracting Cloud Free Sentinel-2 Images Using Google Earth Engine Python API



Results - Lagos

Original Image

Cloud-free Image

TIF Image

Define collection filter and cloud mask parameters

```
AOI = ee.Geometry.MultiPolygon(shapefile['features'][0]["geometry"]['coordinates'])
START_DATE = '2021-12-20'
END_DATE = '2022-01-23'
CLOUD_FILTER = 60
CLD_PRB_THRESH = 65
NIR_DRK_THRESH = 0.80
CLD_PRJ_DIST = 2
BUFFER = 100
```

Apply cloud and cloud shadow mask

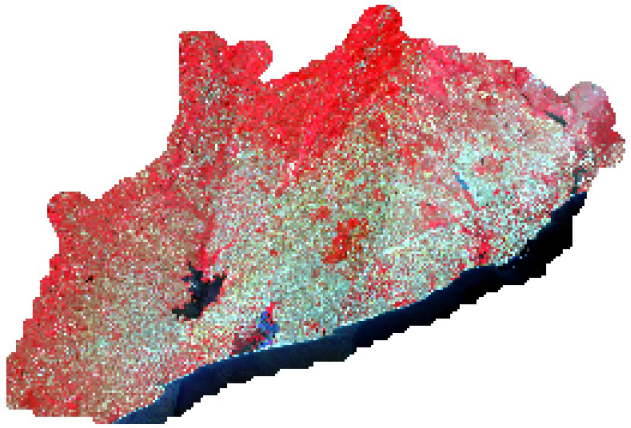
```
AOI = ee.Geometry.MultiPolygon(shapefile['features'][0]["geometry"]['coordinates'])
START_DATE = '2021-12-20'
END_DATE = '2022-01-23'
CLOUD_FILTER = 60
CLD_PRB_THRESH = 60
NIR_DRK_THRESH = 1
CLD_PRJ_DIST = 6
BUFFER = 100
```

Results - Accra

Original Image

Cloud-free Image

TIF Image



Define collection filter and cloud mask parameters

```
AOI = ee.Geometry.MultiPolygon(shapefile['features'][0]["geometry"]['coordinates'])
START_DATE = '2021-11-14’
END_DATE = '2021-12-23’
CLOUD_FILTER = 30
CLD_PRB_THRESH = 85
NIR_DRK_THRESH = 0.50
CLD_PRJ_DIST = 2
BUFFER = 65
```

Apply cloud and cloud shadow mask parameters

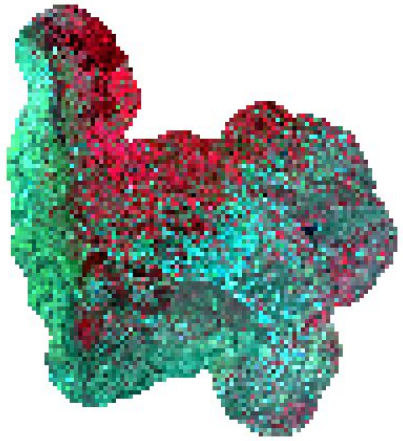
```
AOI = ee.Geometry.MultiPolygon(shapefile['features'][0]["geometry"]['coordinates'])
START_DATE = '2021-11-14’
END_DATE = '2021-12-23’
CLOUD_FILTER = 30
CLD_PRB_THRESH = 80
NIR_DRK_THRESH = 1
CLD_PRJ_DIST = 2
BUFFER = 90
```

Results - Nairobi

Original Image

Cloud-free Image

TIF Image



Define collection filter and cloud mask parameters

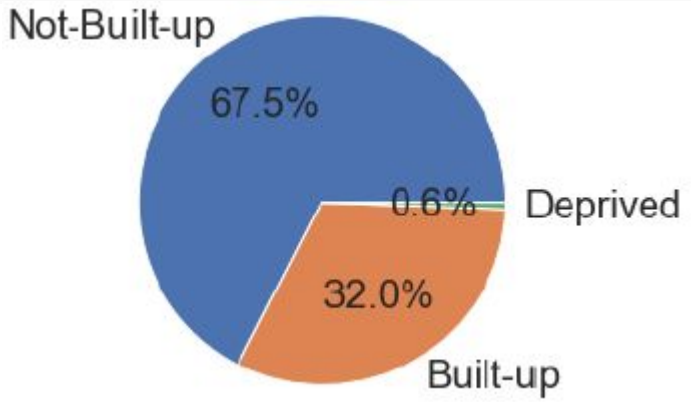
```
AOI = ee.Geometry.MultiPolygon(shapefile['features'][0]["geometry"]['coordinates'])
START_DATE = '2022-01-03’
END_DATE = '2022-01-14’
CLOUD_FILTER = 60
CLD_PRB_THRESH = 50
NIR_DRK_THRESH = 0.15
CLD_PRJ_DIST = 1
BUFFER = 50
```

Apply cloud and cloud shadow mask parameters

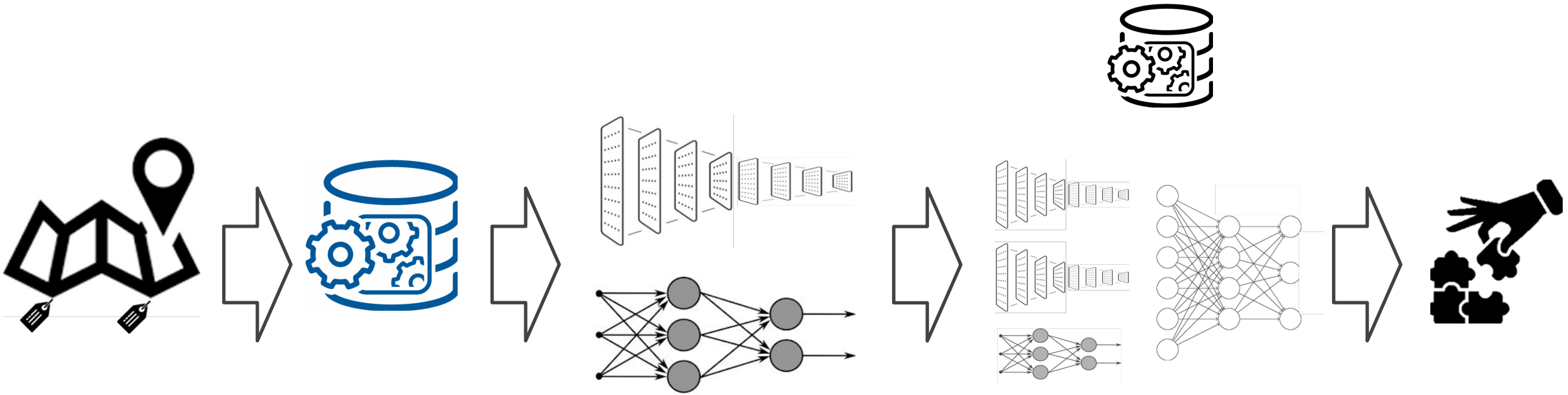
```
AOI = ee.Geometry.MultiPolygon(shapefile['features'][0]["geometry"]['coordinates'])
START_DATE = '2022-01-03’
END_DATE = '2022-01-14’
CLOUD_FILTER = 60
CLD_PRB_THRESH = 50
NIR_DRK_THRESH = 0.15
CLD_PRJ_DIST = 3
BUFFER = 70
```

Key Findings about the data

- Data is highly imbalanced (269 deprived labels out of 47K labeled areas)
- Sentinel-2 satellite images are low resolution when compared to other high resolutions images (i.e. google maps)
- GeoTiff files needs to be converted to standard image format for Deep Learning
- Non-built-up labels are areas with no buildings and typically representing forests, lakes and other non-built-up categories

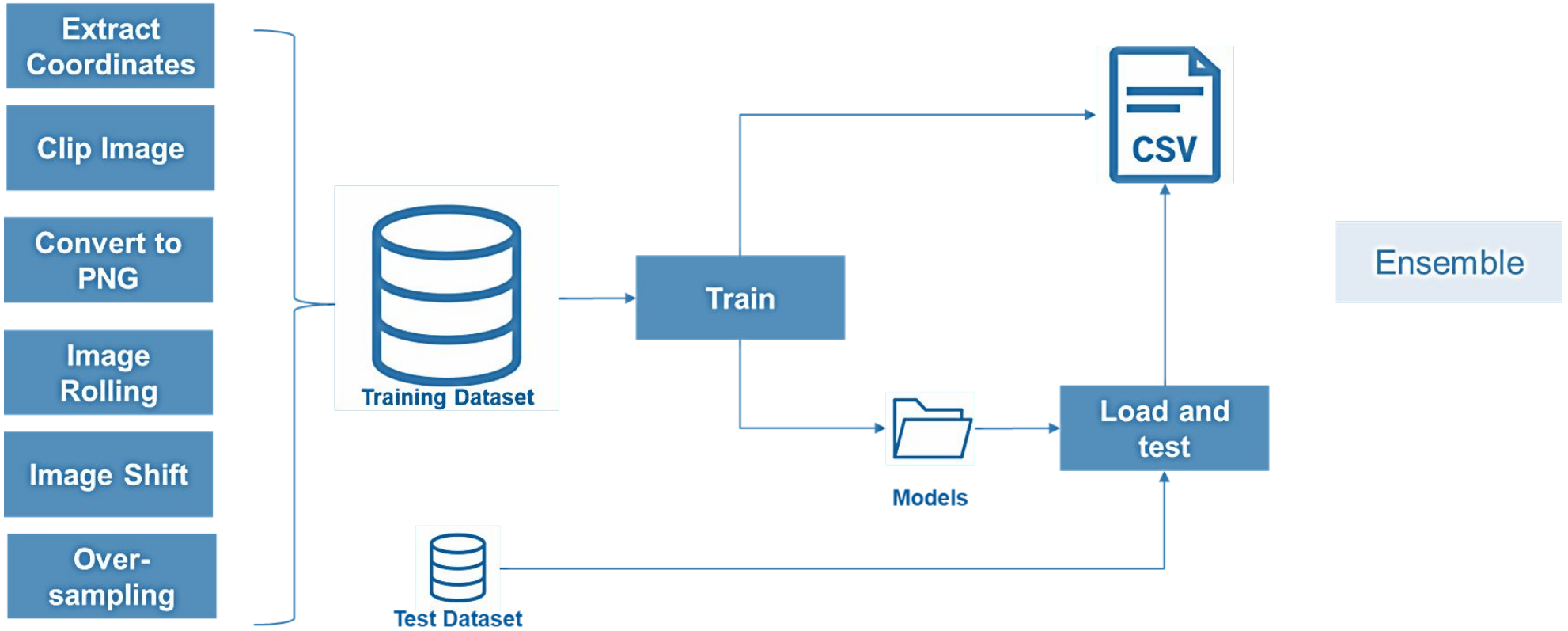


Agenda



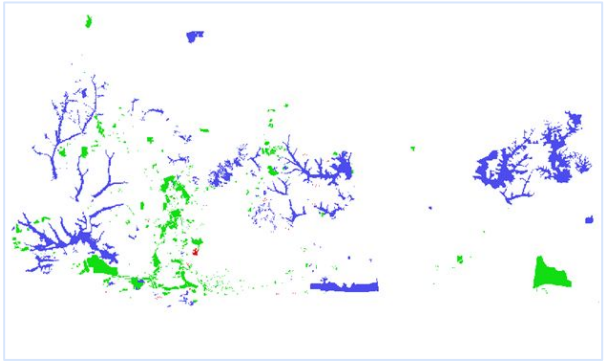
The Data | Processing | Modeling | Results 1 | Modeling | Results 2 | Conclusion

The Big Picture



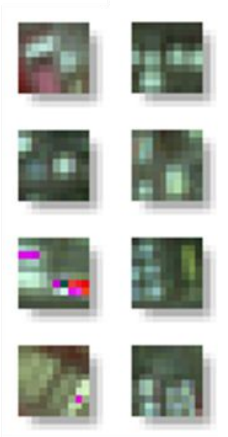
Data Processing Steps

1 **Extract label coordinates**



	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

2 **Clip images**

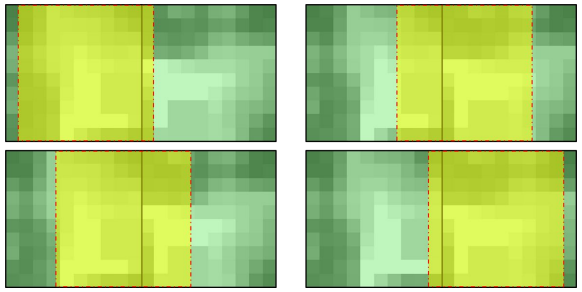


Labeled

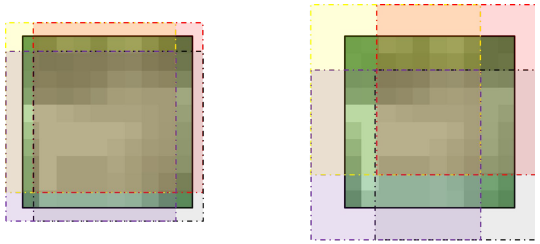
Data Processing Steps

3 Image Augmentation

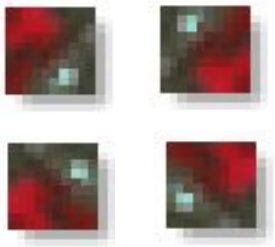
Roll through adjacent images



Shift image by one or two pixels



Rotate all images three times



4 Conversion to PNG

- Using NIR band
- Normalize pixel values
- Follow naming convention

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

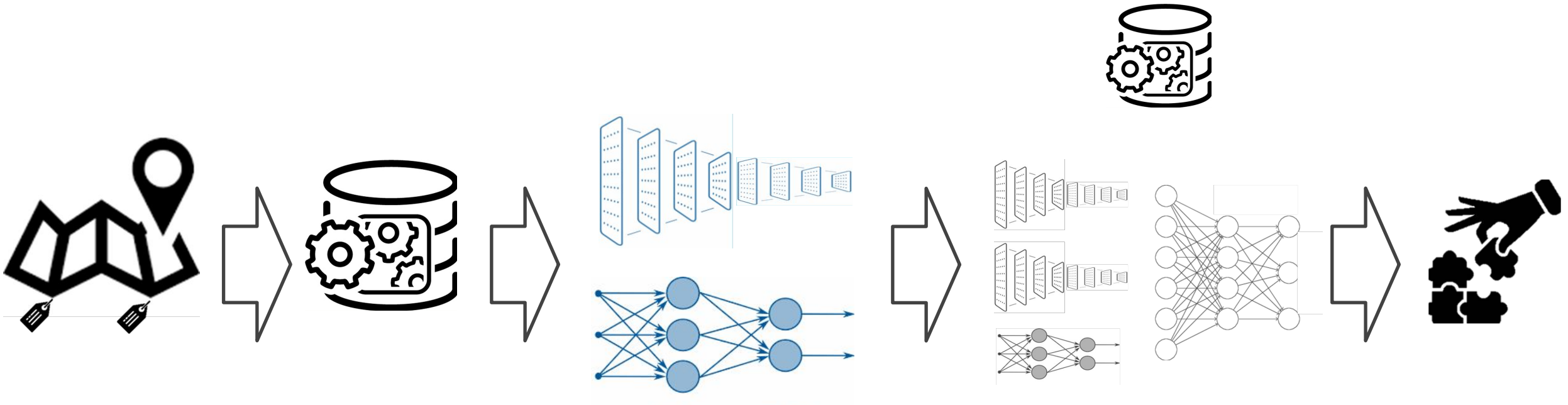
Key Takeaways from Data Pre-processing

- Generation of more than 8000 labeled images from minority class (~215 images)
- Using of GDAL and Rasterio libraries to process geo-tiff files
- Using QGIS to validate data and overlay different images
- Data was split using a stratified random sampling to maintain ratios

Rasterio

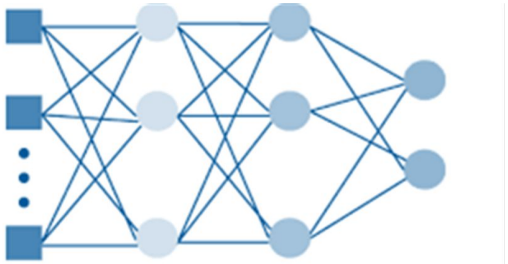
QGIS

Agenda

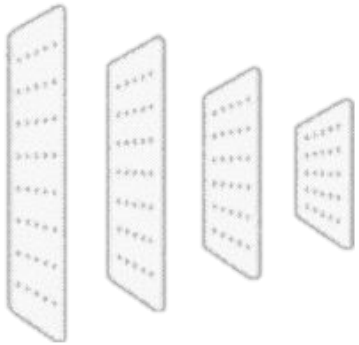


The Data | Processing | Modeling | Results 1 | Modeling | Results 2 | Conclusion

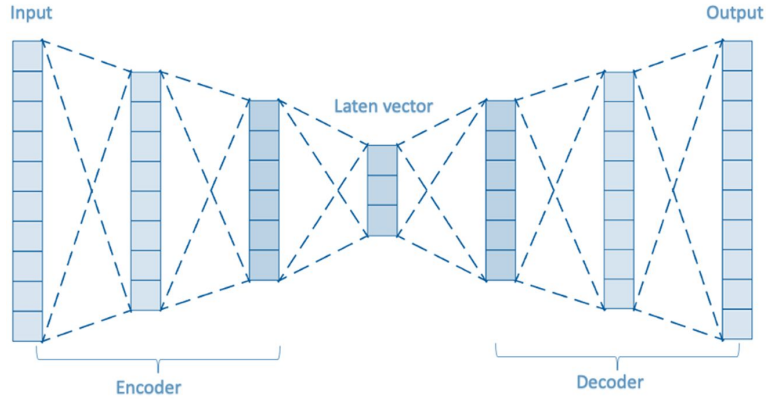
The big picture



MLP

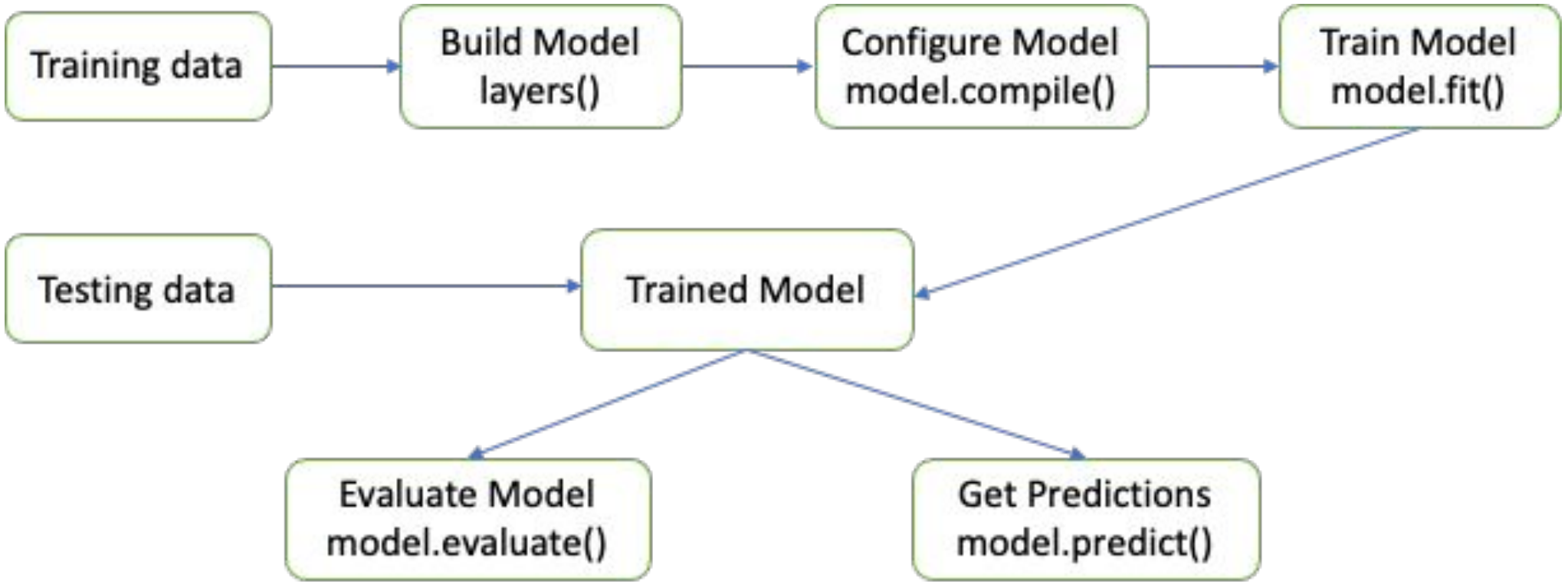


CNN

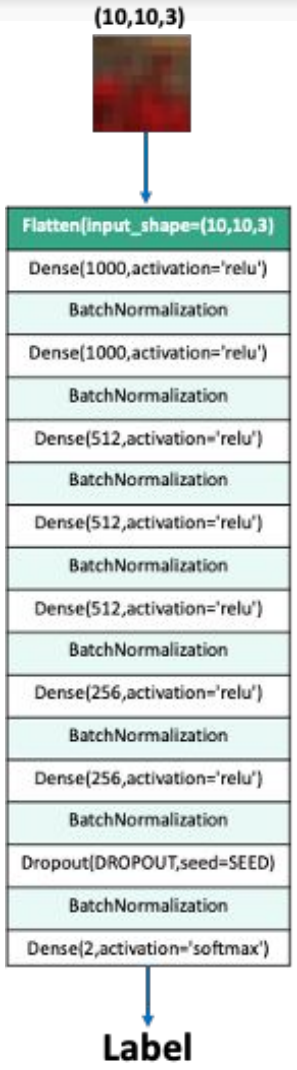


AutoEncoder

Modeling


Training data
Build Model
layers()
Configure Model
model.compile()
Train Model
model.fit()
Testing data
Trained Model
Evaluate Model
model.evaluate()
Get Predictions
model.predict()

Deep Neural Network – MLP

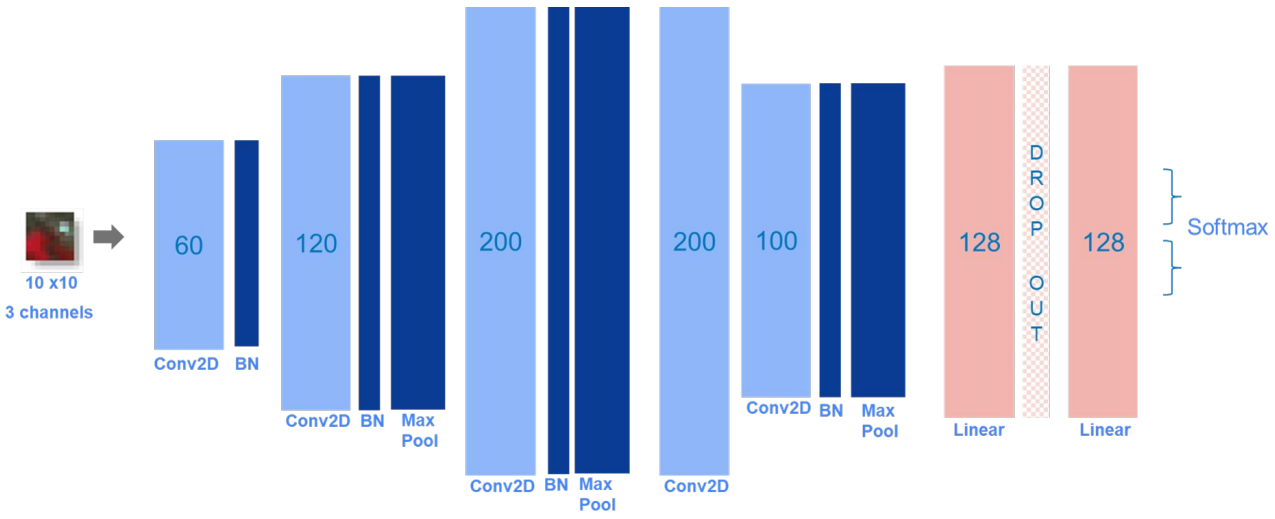


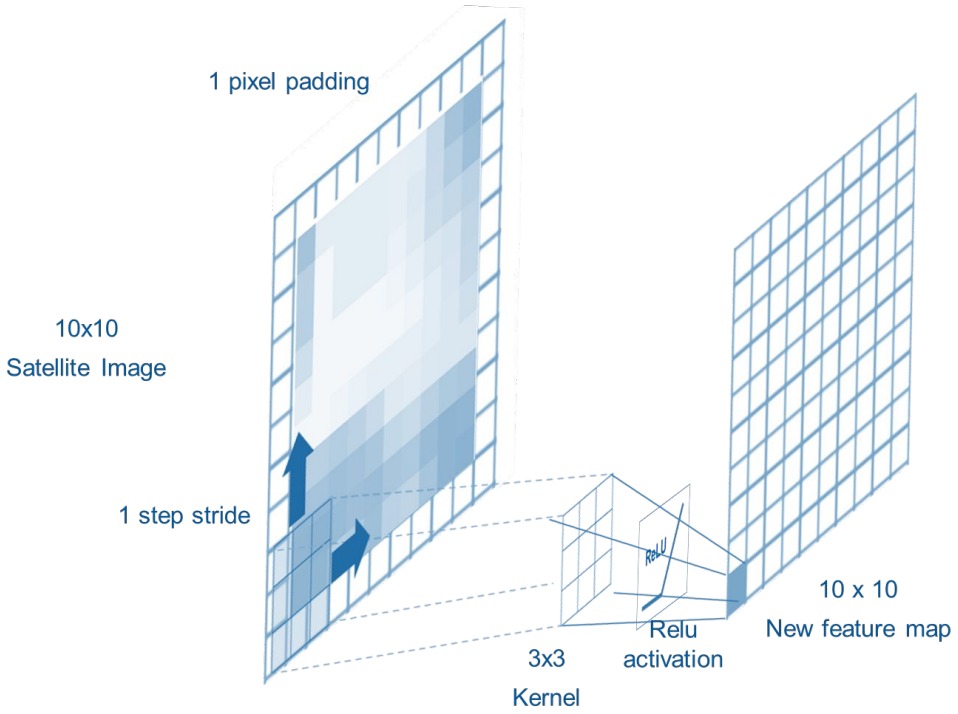
	MLP Models	Optimizer	Precision "1"	Recall "1"	f1 score "1"	MacroAvg Precision	MacroAvg Recall	MacroAvg f1 score
M01	2 Labels | Original images	adama	0.04	0.26	0.06	0.51	0.55	0.49
M03	2 Labels | cloud free image + augm.(rotaion,rolling and shifting)	RMSprop	0.06	0.46	0.11	0.52	0.65	0.51
M04	2 Labels | cloud free image + augm.(rotaion,rolling and shifting)	adagrad	0.07	0.59	0.12	0.53	0.70	0.51
M05	2 Labels | cloud free image + augm.(rotaion,rolling and shifting)	adam	0.08	0.56	0.14	0.54	0.71	0.53

Confusion Matrix - MLP		
	Built-up	Deprived
Built-up	2071	330
Deprived	24	30

- All models were trained for 200 epochs with early stopping

Deep Neural Network – CNN





- Elimination of the labels representing the non-built-up areas which can be achieved via other methods
- Using Kernel size of 3x3 while keeping padding set to ‘same’ to keep the 10x10 image size across the layers
- We used 100 epochs with early stopping. Key Metrics for the models were F-1 score
- Using Grid Search to reach optimum hyper parameters (lr, dropout, neurons, optimizer)

Deep Neural Network – CNN Highlights

	CNN Models	Test Acc	Precision "1"	Recall "1"	f1 score "1"	MacroAvg Precision	MacroAvg Recall	MacroAvg F1 score
M01	3 Labels |original image + class weights	0.63	0.02	0.78	0.03	0.55	0.64	0.46
M02	2 Labels |original image + class weights	0.69	0.03	0.57	0.06	0.51	0.63	0.44
M03	2 Labels |cloud free image + class weights	0.72	0.04	0.72	0.08	0.52	0.72	0.46
M04	2 Labels |cloud free image + class weights + augmt. (rotation)	0.84	0.15	0.87	0.26	0.57	0.85	0.58
M05	2 Labels |cloud free image + class weights + augmt. (rotation) VGG16	0.85	0.15	0.76	0.25	0.57	0.81	0.58
M06	2 Labels |cloud free image + class weights + augmt. (rotation) - blank images	0.92	0.27	0.63	0.37	0.63	0.78	0.67
M07	2 Labels |cloud free image + class weights + augmt. (rotation) - blank images VGG16	0.87	0.18	0.67	0.29	0.58	0.77	0.61
M08	2 Labels |cloud free image + class weights + augmt. (rotation) - blank images + dilation	0.88	0.21	0.70	0.32	0.60	0.80	0.63
M09	2 Labels | cloud free image + augmt.(rotation, rolling and shifting)	0.96	0.32	0.63	0.43	0.66	0.80	0.70
M10	Hard Voting CNN Ensemble	0.97	0.41	0.65	0.50	0.70	0.81	0.74

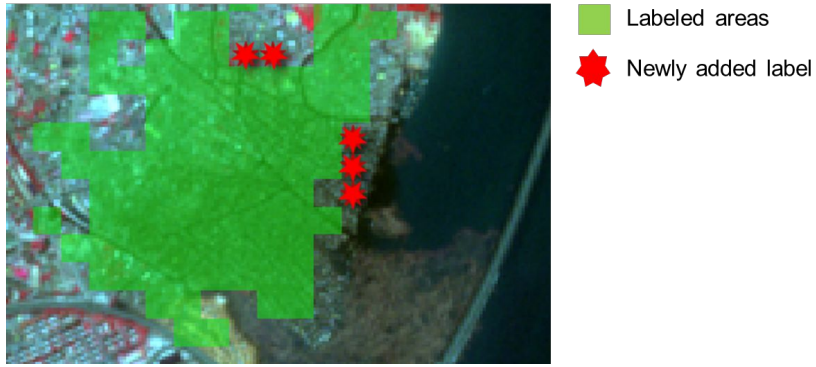
Confusion Matrix		
	Built-up	Deprived
Built-up	1939	50
Deprived	19	35

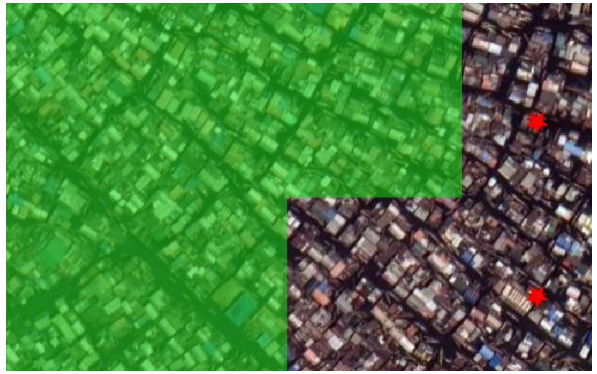
Key Takeaways from CNN and MLP modeling

- Cloud free images with NIR band provided better results
- Small image size (10x10) could be the reason behind low performance of pre-trained models and dilation
- Class weights were used to penalize minority class misclassification but did not add significant improvements
- Augmentation was a key factor for improving model performance
- Hard-voting ensemble was used to combine the best performing models

Model Experiment

- Since the number of images kept aside for testing the model was relatively small (54 images based on the random stratified sampling), 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 model metrics were consistent. To further validate the numbers, the following experiment was conducted:
  - Annotation of new labels using manual reviews on high resolution google maps and street views
  - Split the expanded dataset which will have more test images
  - Run the model with more data points
  - Validate the KPIs



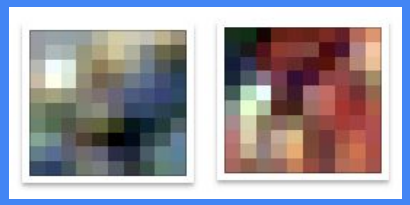


Constructing MLP Autoencoder Using Google Earth Engine Images

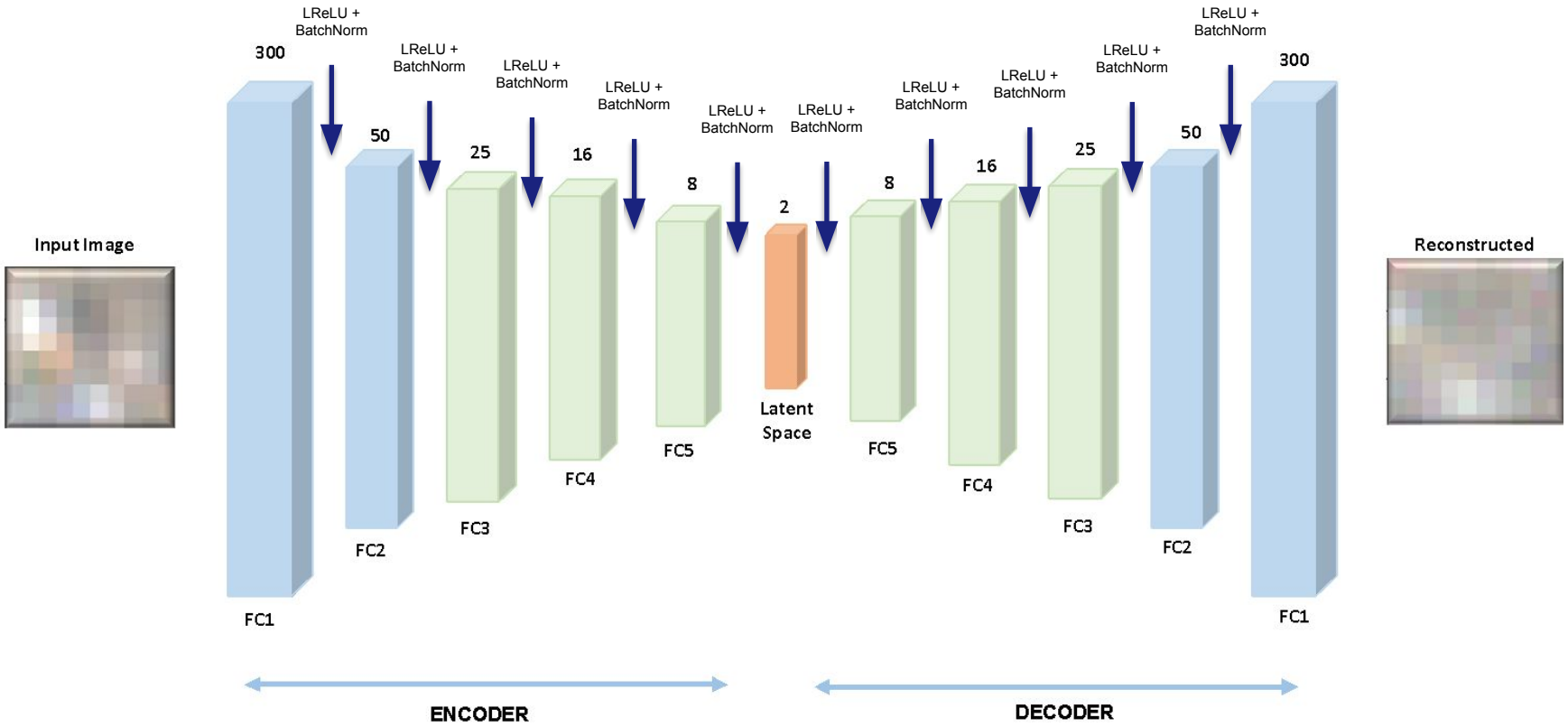
Google Earth Images:
- **Lagos**
- **Accra**
- **Nairobi**

Total images : 1030089
Image size : 10 x 10 px

Before normalization

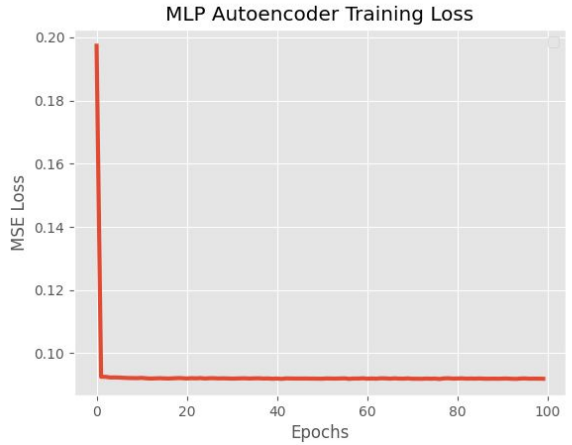


After normalization

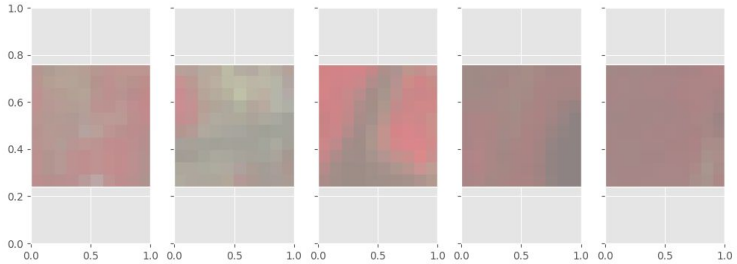


Results – MLP Autoencoder

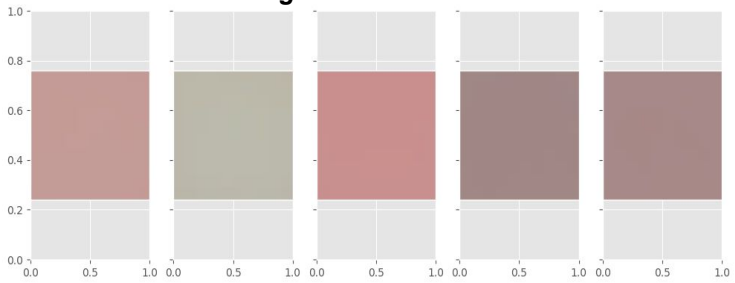
Batch size : 32
Epochs : 100
LR : 1e-4
Noise : 0.2
MSE : 0.091



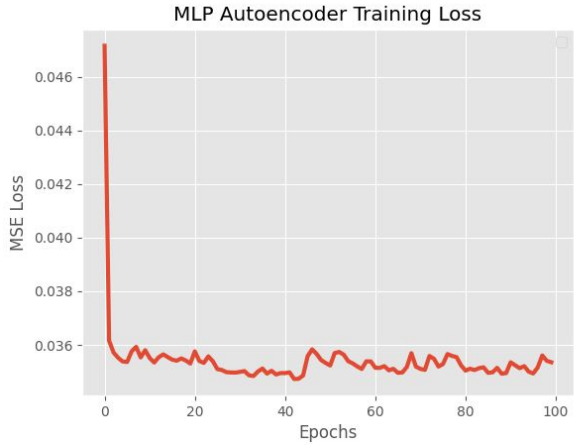
Original Images



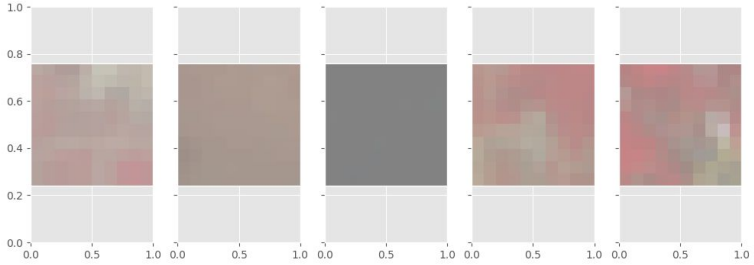
Reconstructed Images



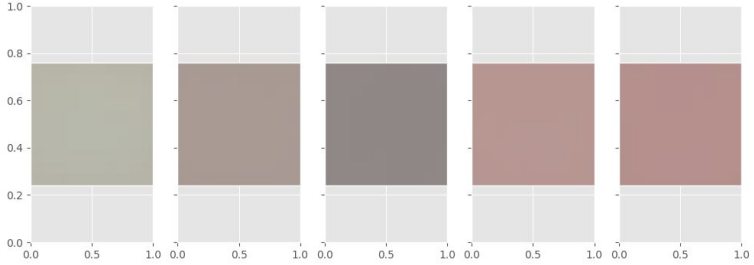
Batch size : 10
Epochs : 100
LR : 1e-4
Noise : 0.2
MSE : 0.035



Original Images

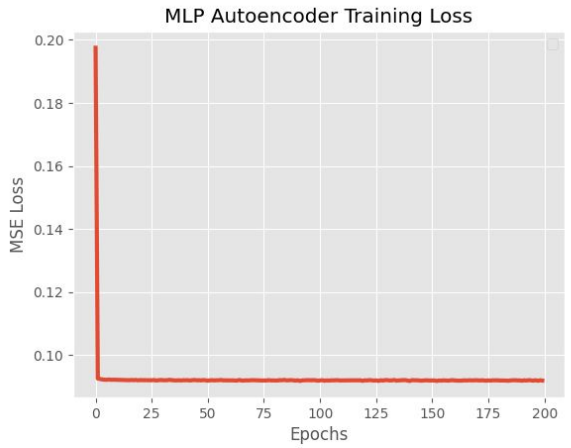


Reconstructed Images

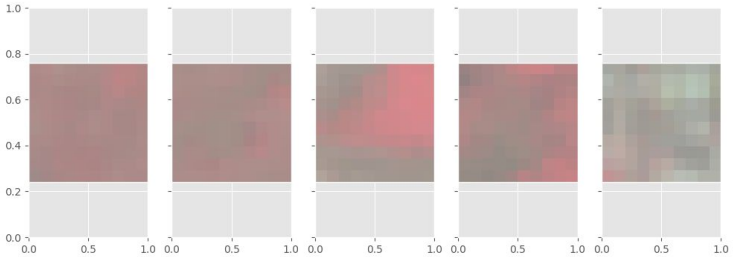


Results – MLP Autoencoder

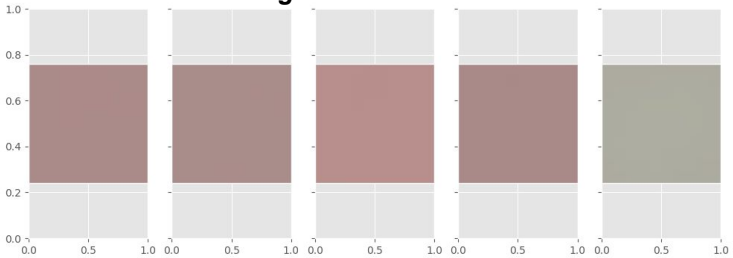
Batch size : 32
Epochs : 200
LR : 1e-4
Noise : 0.2
MSE : 0.091



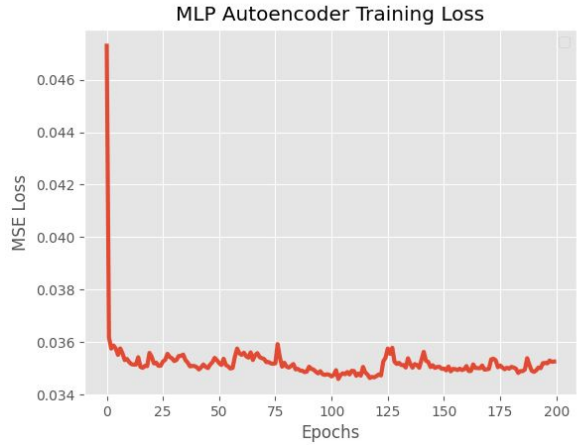
Original Images



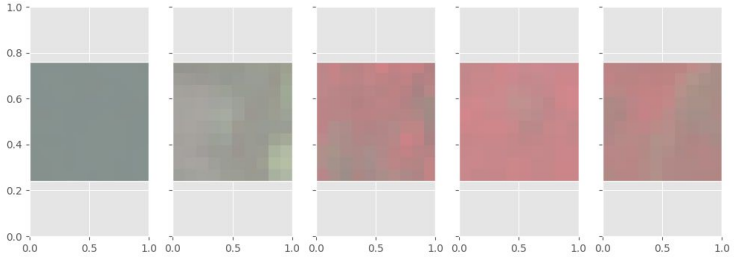
Reconstructed Images



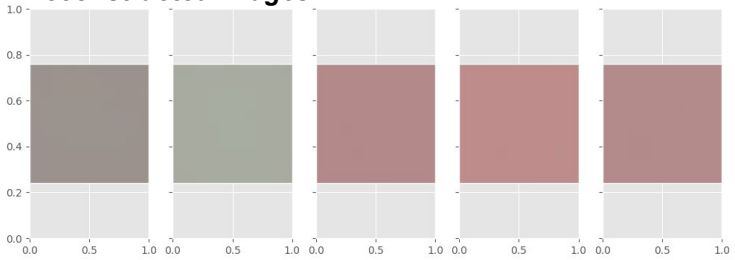
Batch size : 10
Epochs : 200
LR : 1e-4
Noise : 0.2
MSE : 0.035



Original Images



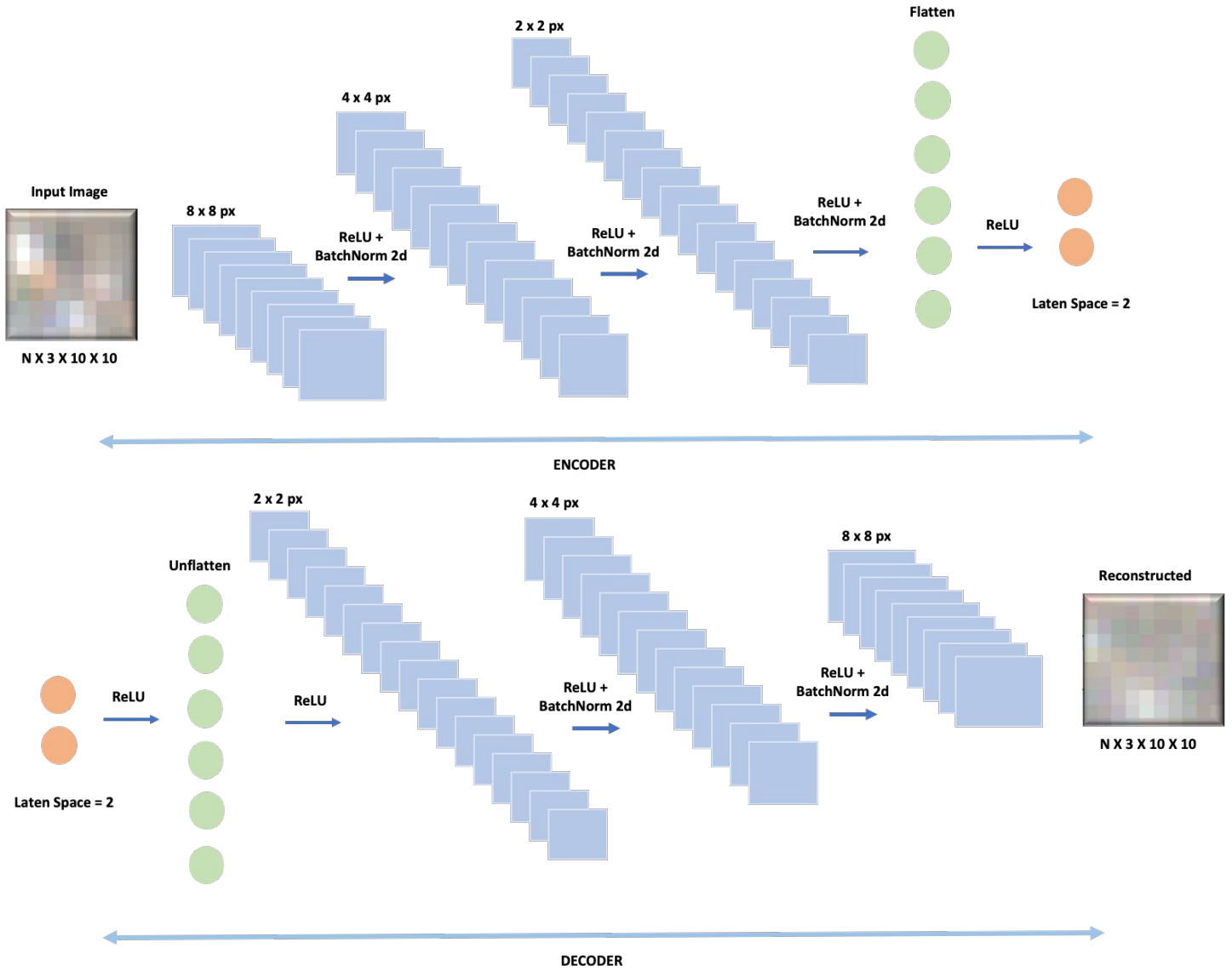
Reconstructed Images



Constructing CNN Autoencoder Using Google Earth Images

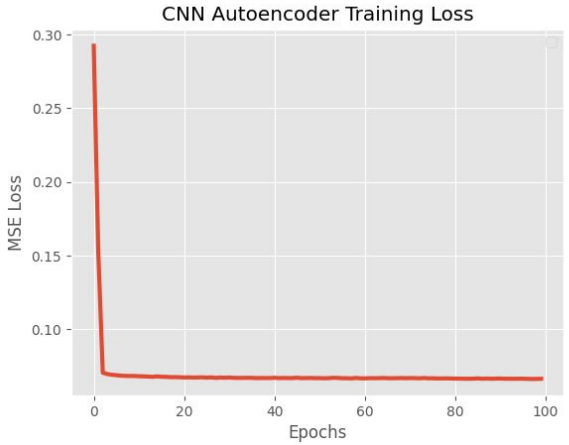
Google Earth Images:
- Lagos
- Accra
- Nairobi

Total images : 1,030,089
Image size : 10 x 10 px

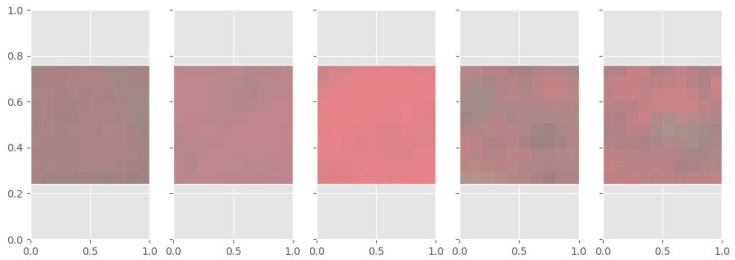


Results – CNN Autoencoder

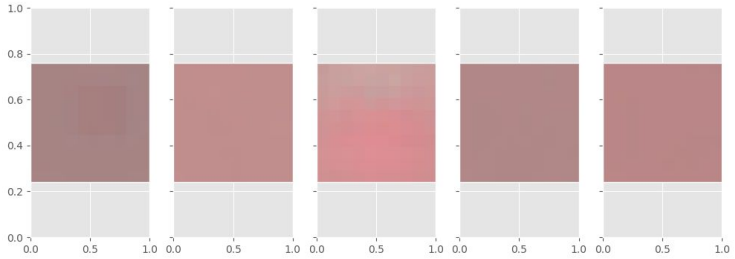
Batch size : 32
Epochs : 100
LR : 1e-4
Noise : 0.2
MSE : 0.066



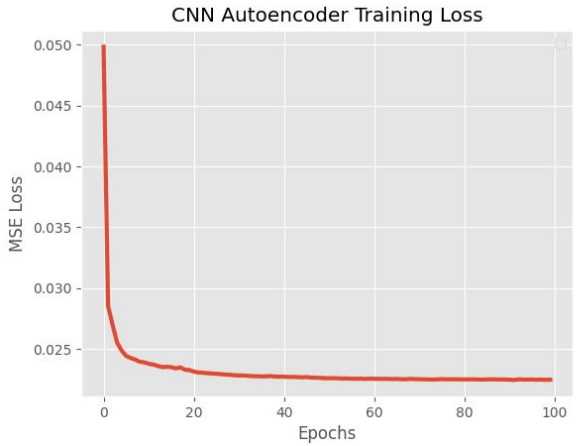
Original Images



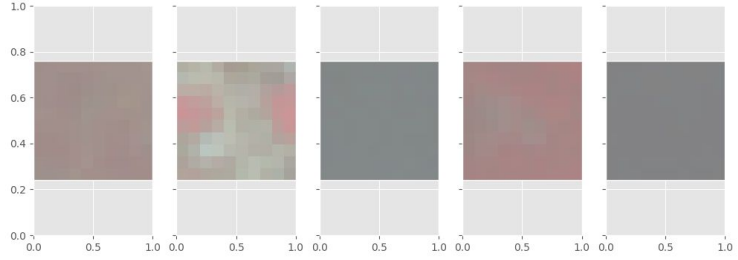
Reconstructed Images



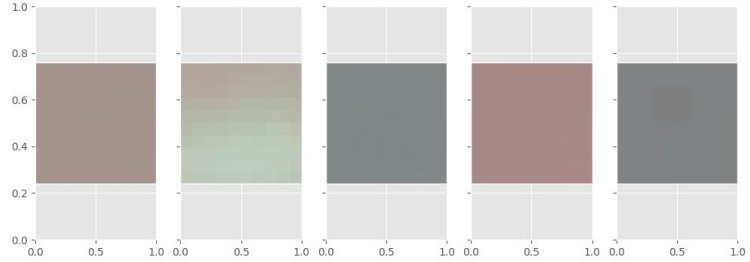
Batch size : 10
Epochs : 100
LR : 1e-4
Noise : 0.2
MSE : 0.022



Original Images

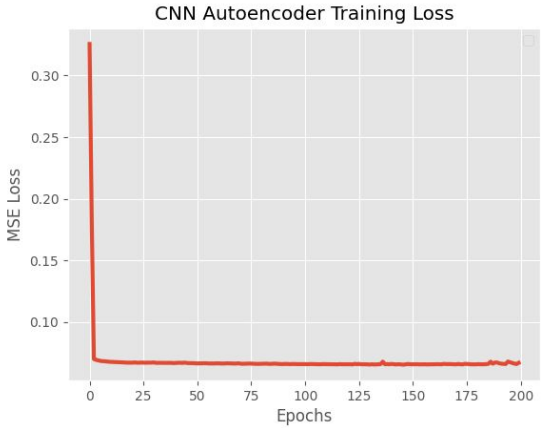


Reconstructed Images

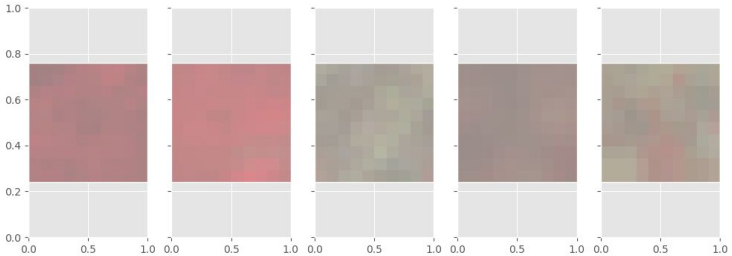


Results – CNN Autoencoder

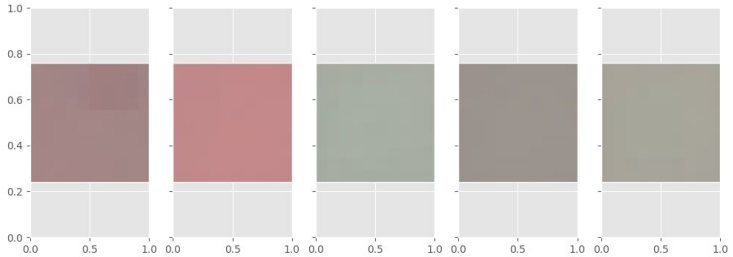
Batch size : 32
Epochs : 200
LR : 1e-4
Noise : 0.2
MSE : 0.067



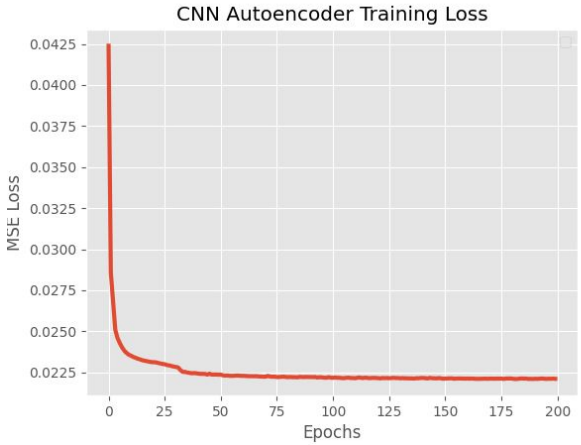
Original Images



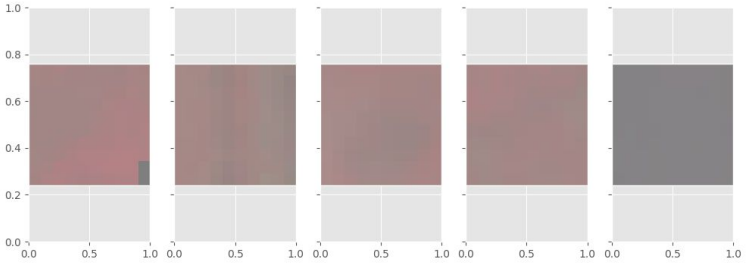
Reconstructed Images



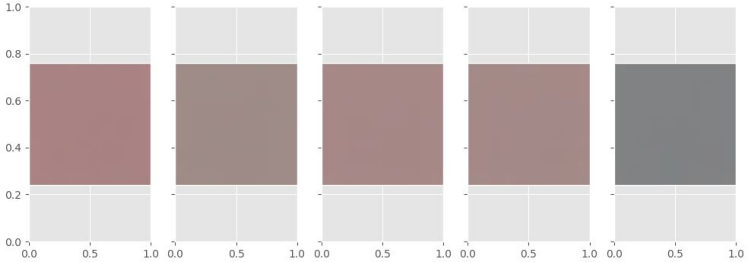
Batch size : 10
Epochs : 200
LR : 1e-4
Noise : 0.2
MSE : 0.022



Original Images

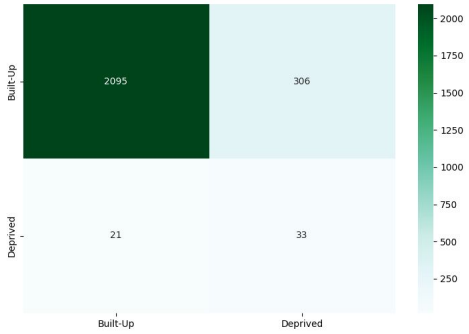
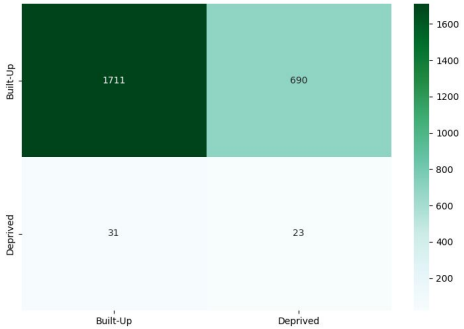


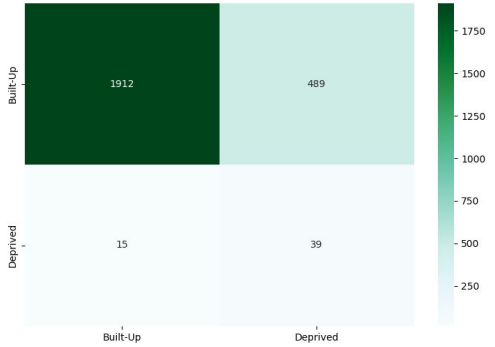
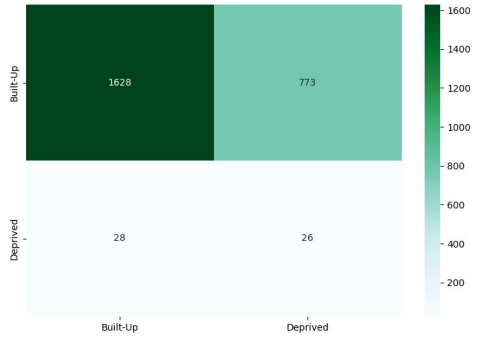
Reconstructed Images



Classifying Training Dataset - Lagos

Autoencoder model trained with batch size 32 over 100 epochs

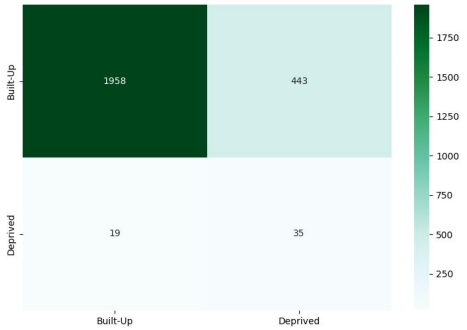
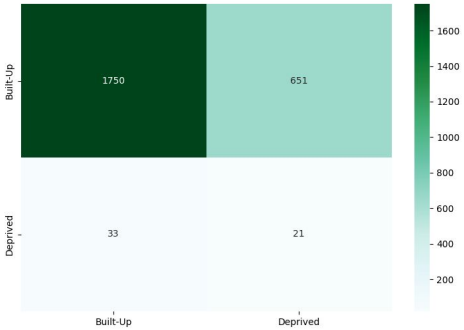
W/O Scheduler	CNN		MLP	
	Build-up (Class – 0)	***Deprived (Class – 1)***	***Build-up (Class – 0)***	***Deprived (Class – 1)***
F1 Score	0.93	0.17	0.83	0.06
Precision	0.99	0.10	0.98	0.03
Recall	0.87	0.61	0.71	0.43
Accuracy	0.87		0.71	
Confusion Matrix				
Macro Avg F1	0.55		0.44	
Weighted Avg F1	0.91		0.81	

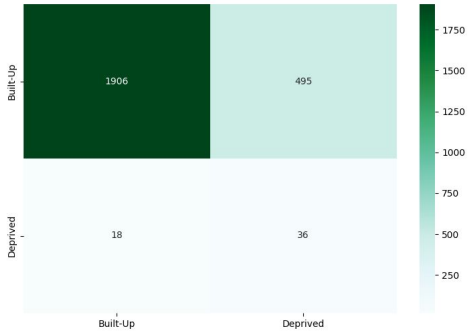
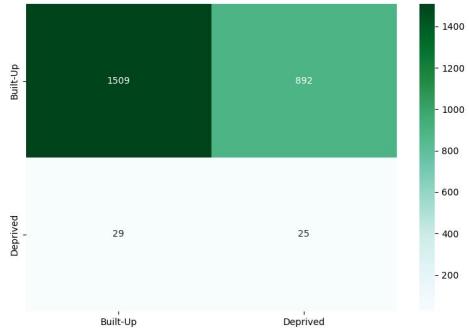
With Scheduler	CNN		MLP	
	Build-up (Class – 0)	***Deprived (Class – 1)***	***Build-up (Class – 0)***	***Deprived (Class – 1)***
F1 Score	0.88	0.13	0.80	0.06
Precision	0.99	0.07	0.98	0.03
Recall	0.80	0.72	0.68	0.48
Accuracy	0.79		0.67	
Confusion Matrix				
Macro Avg F1	0.51		0.43	
Weighted Avg F1	0.87		0.79	

Batch size : 32 | Epochs : 100 | LR : 1 e-4 | Optimizer : Adam

Classifying Training Dataset - Lagos

Autoencoder model trained with batch size 10 over 100 epochs

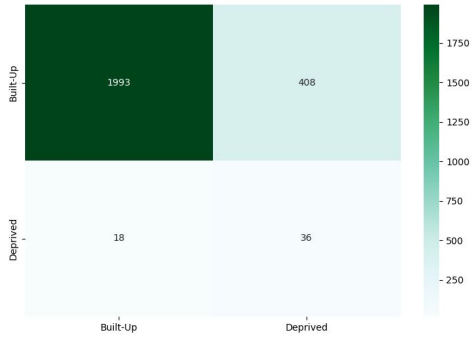
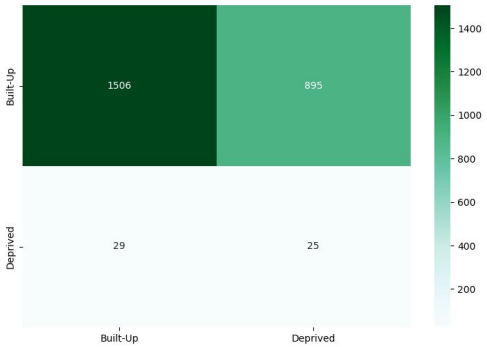
W/O Scheduler	CNN		MLP	
	Build-up (Class – 0)	***Deprived (Class – 1)***	***Build-up (Class – 0)***	***Deprived (Class – 1)***
F1 Score	0.89	0.13	0.84	0.06
Precision	0.99	0.07	0.98	0.03
Recall	0.82	0.65	0.73	0.39
Accuracy	0.81		0.72	
Confusion Matrix				
Macro Avg F1	0.51		0.45	
Weighted Avg F1	0.88		0.82	

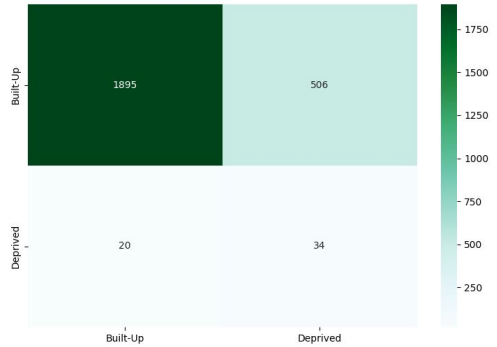
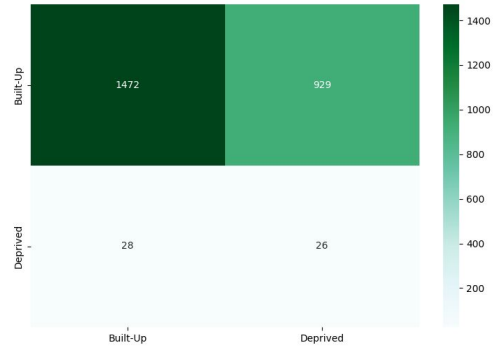
With Scheduler	CNN		MLP	
	Build-up (Class – 0)	***Deprived (Class – 1)***	***Build-up (Class – 0)***	***Deprived (Class – 1)***
F1 Score	0.88	0.12	0.77	0.05
Precision	0.99	0.07	0.98	0.03
Recall	0.79	0.67	0.63	0.46
Accuracy	0.79		0.62	
Confusion Matrix				
Macro Avg F1	0.50		0.41	
Weighted Avg F1	0.86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

Batch size : 32 | Epochs : 100 | LR : 1 e-4 | Optimizer : Adam

Classifying Training Dataset - Lagos

Autoencoder model trained with batch size 32 over 200 epochs

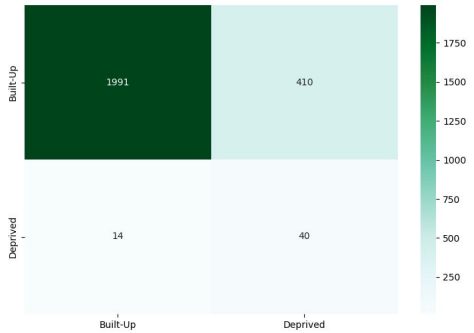
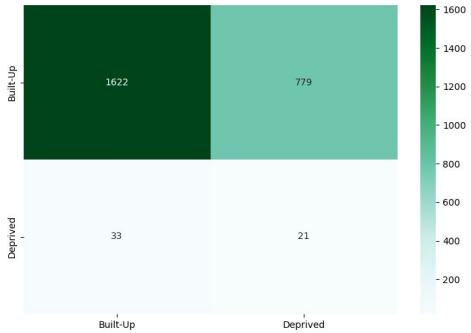
W/O Scheduler	CNN		MLP	
	Build-up (Class – 0)	***Deprived (Class – 1)***	***Build-up (Class – 0)***	***Deprived (Class – 1)***
F1 Score	0.90	0.14	0.77	0.05
Precision	0.99	0.08	0.98	0.03
Recall	0.83	0.67	0.63	0.46
Accuracy	0.83		0.62	
Confusion Matrix				
Macro Avg F1	0.52		0.41	
Weighted Avg F1	0.89		0.75	

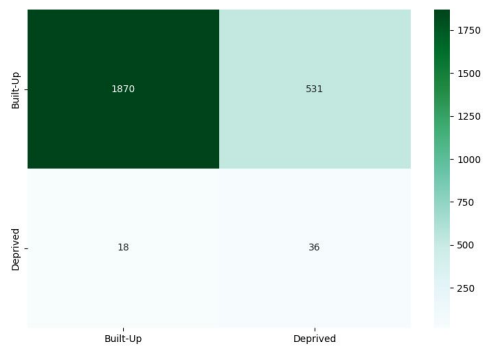
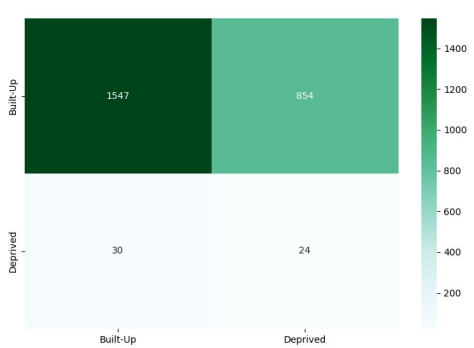
With Scheduler	CNN		MLP	
	Build-up (Class – 0)	***Deprived (Class – 1)***	***Build-up (Class – 0)***	***Deprived (Class – 1)***
F1 Score	0.88	0.11	0.78	0.05
Precision	0.99	0.06	0.98	0.03
Recall	0.79	0.61	0.61	0.48
Accuracy	0.79		0.64	
Confusion Matrix				
Macro Avg F1	0.50		0.41	
Weighted Avg F1	0.86		0.76	

Batch size : 32 | Epochs : 100 | LR : 1 e-4 | Optimizer : Adam

Classifying Training Dataset - Lagos

Autoencoder model trained with batch size 10 over 200 epochs

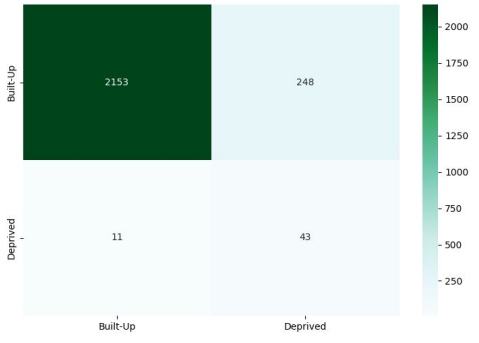
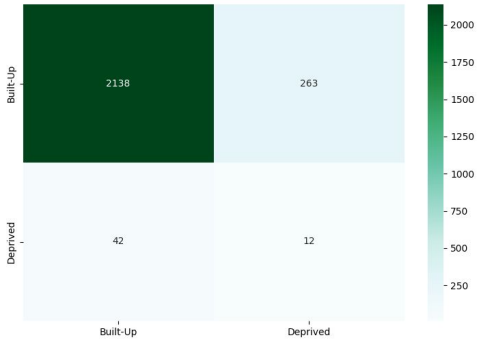
W/O Scheduler	CNN		MLP	
	Build-up (Class – 0)	***Deprived (Class – 1)***	***Build-up (Class – 0)***	***Deprived (Class – 1)***
F1 Score	0.90	0.16	0.80	0.05
Precision	0.99	0.09	0.98	0.39
Recall	0.83	0.74	0.68	0.39
Accuracy	0.83		0.67	
Confusion Matrix				
Macro Avg F1	0.53		0.42	
Weighted Avg F1	0.89		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	

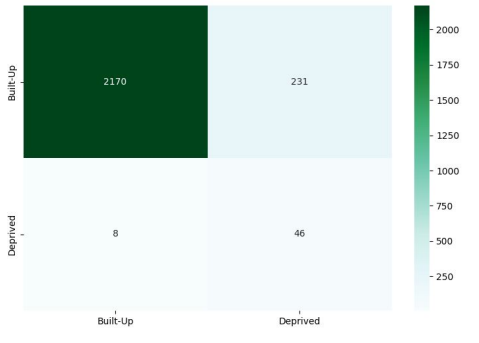
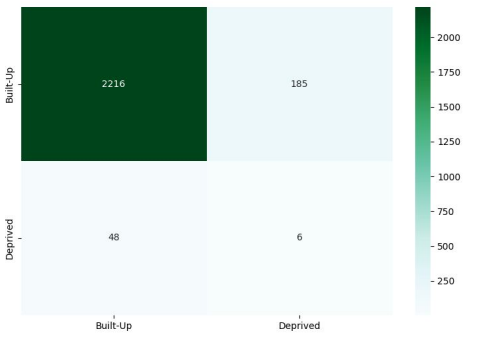
Batch size : 32 | Epochs : 100 | LR : 1 e-4 | Optimizer : Adam

Classifying Training Dataset With Data Augmentation - Lagos

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	0.25	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

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	

Batch size : 32 | Epochs : 100 | LR : 1 e-4 | Optimizer : Adam

MLP Performance Using Autoencoder Generated Images

	MLP - Augmented images using autoencoder	Optimizer	Precision "1"	Recall "1"	f1 score "1"	MacroAvg Precision	MacroAvg Recall	MacroAvg f1 score
M03	2 Labels | cloud free image + augm.(rotaion,rolling and shifting)	adam	0.10	0.67	0.17	0.55	0.77	0.55

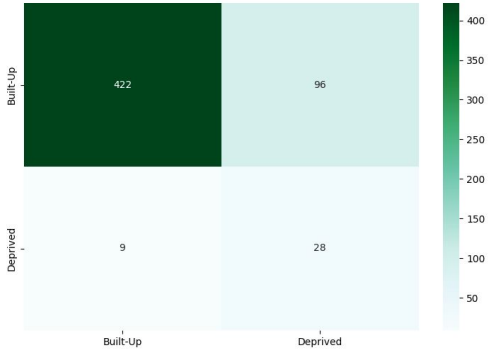
	MLP Models	Optimizer	Precision "1"	Recall "1"	f1 score "1"	MacroAvg Precision	MacroAvg Recall	MacroAvg f1 score
M05	2 Labels | cloud free image + augm.(rotaion,rolling and shifting)	adam	0.08	0.56	0.14	0.54	0.71	0.53

Confusion Matrix - MLP		
	Built-up	Deprived
Built-up	2071	330
Deprived	24	30

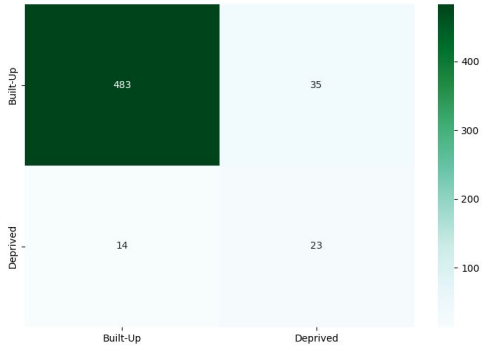
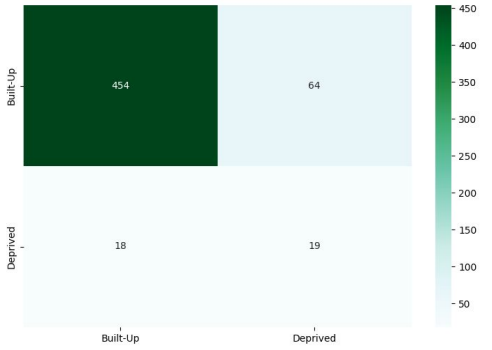
Confusion Matrix - Autoencoder		
	Built-up	Deprived
Built-up	2073	300
Deprived	18	36

Classifying Training Dataset - Accra

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

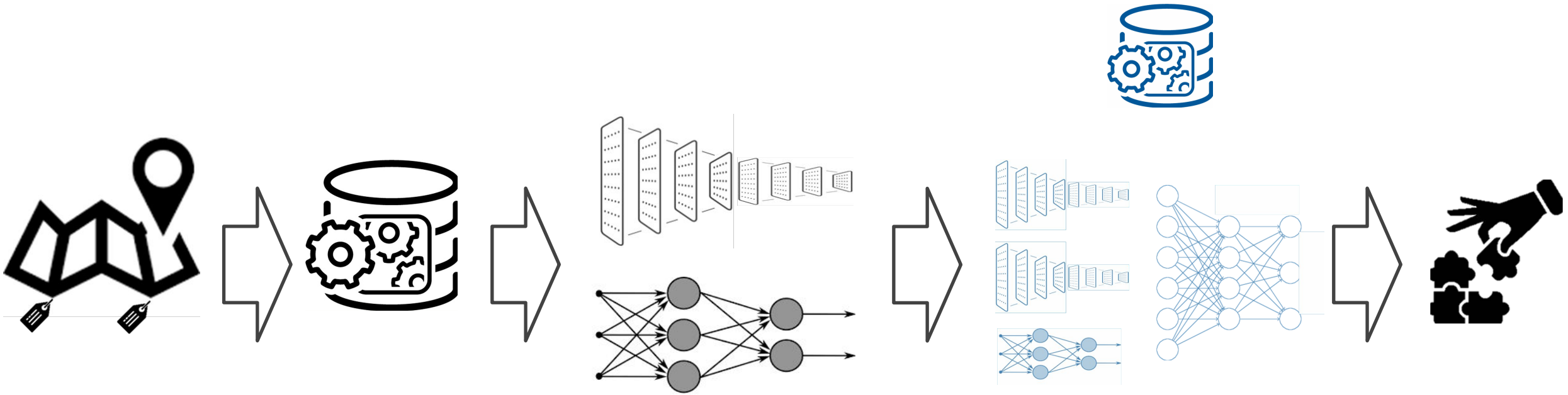
W/O Scheduler	CNN		MLP	
	Build-up (Class – 0)	***Deprived (Class – 1)***	***Build-up (Class – 0)***	***Deprived (Class – 1)***
F1 Score	0.95	0.48	0.92	0.32
Precision	0.97	0.40	0.94	0.23
Recall	0.93	0.62	0.88	0.51
Accuracy	0.91		0.85	
Confusion Matrix				
Macro Avg F1	0.72		0.62	
Weighted Avg F1	0.92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

Batch size : 32 | Epochs : 100 | LR : 1 e-4 | Optimizer : Adam

Key Findings from Autoencoder Model Results

- CNN autoencoder model exceeds in performance compared to MLP autoencoder
- Autoencoder models were more successful in classifying deprived areas in Accra
- Adding more data improved the accuracy of the model predictions
- Further improvisions are required to address the challenges of extracting 100% cloud free satellite images and insufficient training data for deprived class

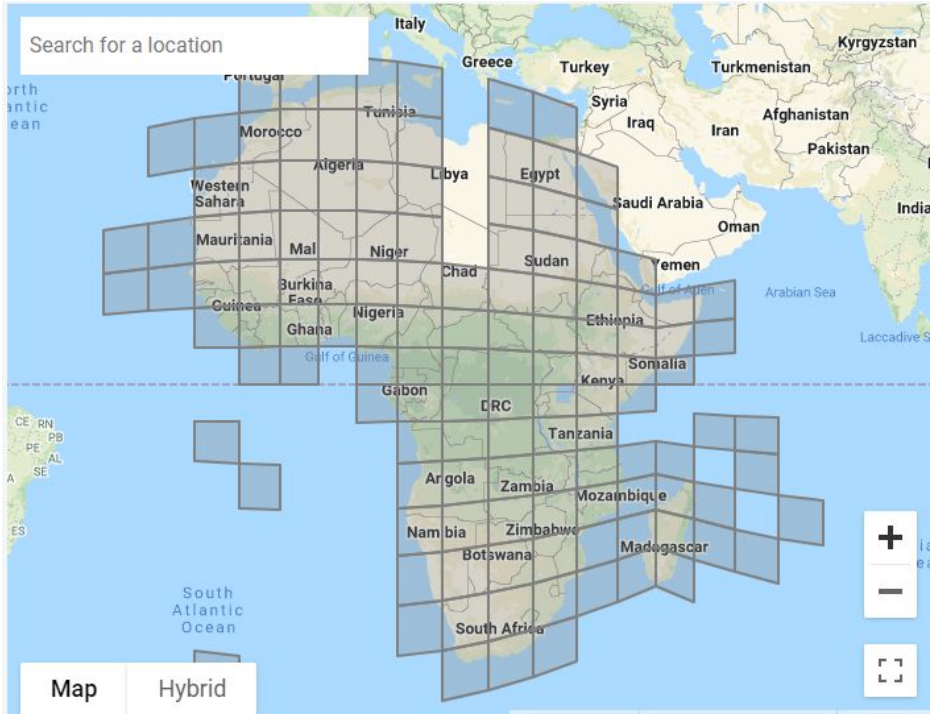
Agenda



The Data **Processing** **Modeling | Results 1** **Modeling | Results 2** **Conclusion**

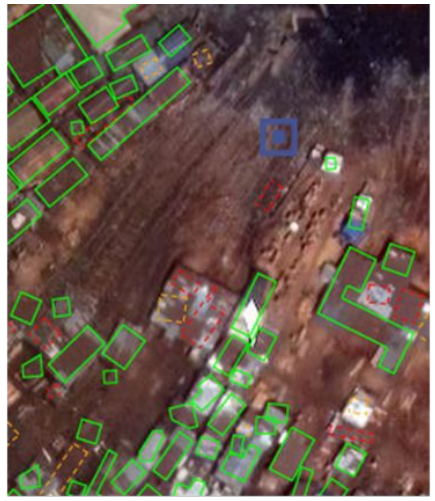
Google Open Building

- A dataset of building footprints to support social good applications
- Creative Commons Attribution (CC BY-4.0) license and the Open Data Commons Open Database License (ODbL) v1.0 license
- The dataset contains 516M building detections, across an area of 19.4M km^2 (64% of the African continent).



Google Open Building – Data Description

- Building polygons are stored in spatially sharded CSVs with one CSV per S2 cell level 4. Each row in the CSV represents one building polygon and has the following columns:
- **latitude**
- **longitude**:
- **area_in_meters**:
- **confidence**:
- **geometry**:
- **full_plus_code**:



Hybrid map from google

Building map from google

latitude	longitude	area_in_meters	confidence	geometry	full_plus_code
7.80710352	3.89791178	17.1779	0.6448	POLYGON((3.89794037826445 7.80711403192458, 3.89788512265294 7.80711822	6FV5RV4X+R5V2
6.60130215	3.61289427	9.5202	0.6181	POLYGON((3.61290983036891 6.60128999343064, 3.61290792883939 6.60131638	6FR5JJ27+G5C4
6.49747113	3.17234674	211.5818	0.7576	POLYGON((3.17248410933092 6.49748719135761, 3.17246218148593 6.49754731	6FR5F5WC+XWQR
10.42324723	3.19575095	33.9013	0.7967	POLYGON((3.1957799991179 10.4232236512091, 3.19577640518097 10.42327456	7F25C5FW+78RR
6.61186314	3.2104962	186.1675	0.8503	POLYGON((3.21062843243659 6.61183939403296, 3.21062618478783 6.61189711	6FR5J666+P5XH

Google Open Building – Data Processing

The first step is to extract the subset for our city boundary and then the following processing will take place

1. Extract buildings that correspond to each labeled boundary (10x10 meters) and calculate different numerical data points for each label (number of buildings, mean/median area.. Etc.)
2. Render an image for the building polygons
3. Augment the data the same way the satellite images were augmented

1

Open Building Dataset

Extract and generate numerical data

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

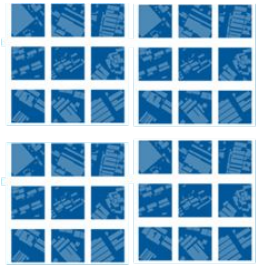
2

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

Generate corresponding building images

3

Apply image augmentation



Update numerical data with augmented images data

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

Using Google Open Building for mapping deprived areas

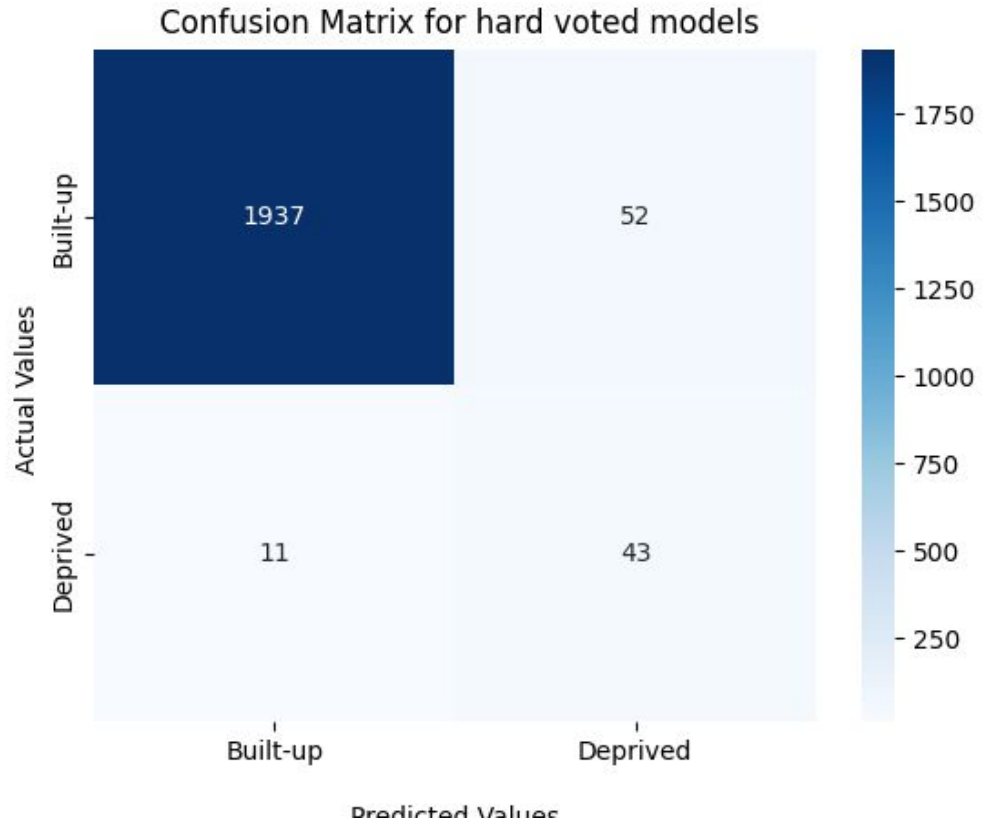
- Using Google Building Images only to train a CNN model was showing a promising results but was very computationally expensive and was not showing a significant increase in performance compared to raw satellite images
- Using VGG16 with partial layers enabled for training we achieved better results compared to original satellite image

	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	0.97	0.44	0.39	0.41	0.71	0.69	0.70

Using Ensemble Model

- Best CNN Model -1
- Best CNN Model -1
- Best Pre-Trained CNN
- Best MLP Model
- Best Open Building Model -1
- Best Open Building Model -2
- Best Pre-trained open building

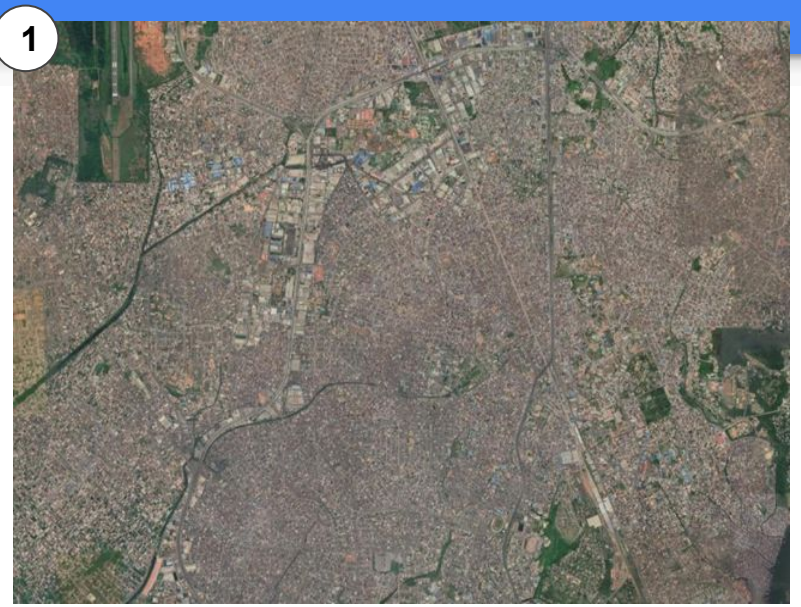
→ Hard Vote Ensemble



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

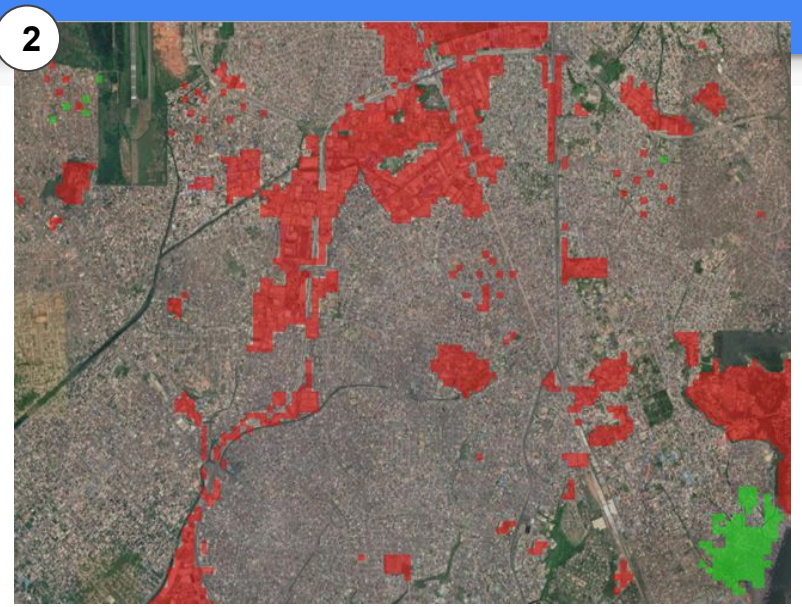
Using the Model

1. Original Map



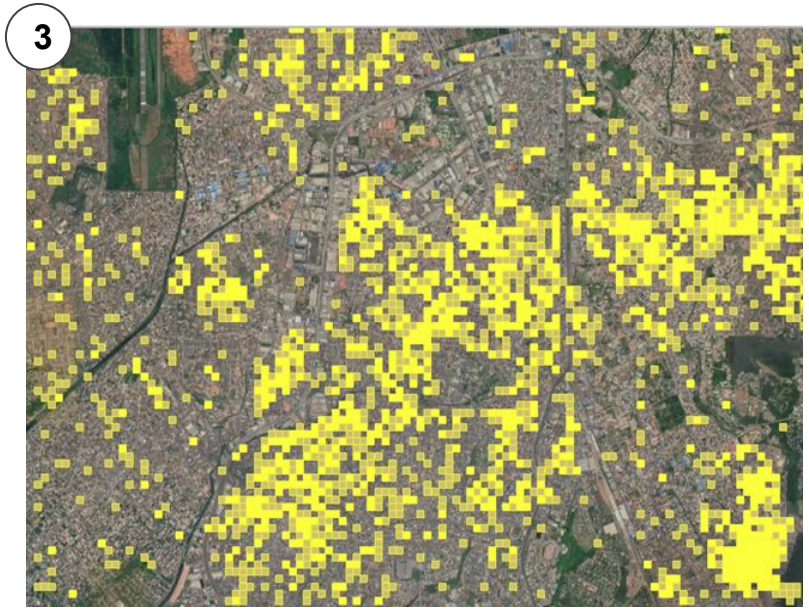
2. Original Map with Labels

Deprived Built-up

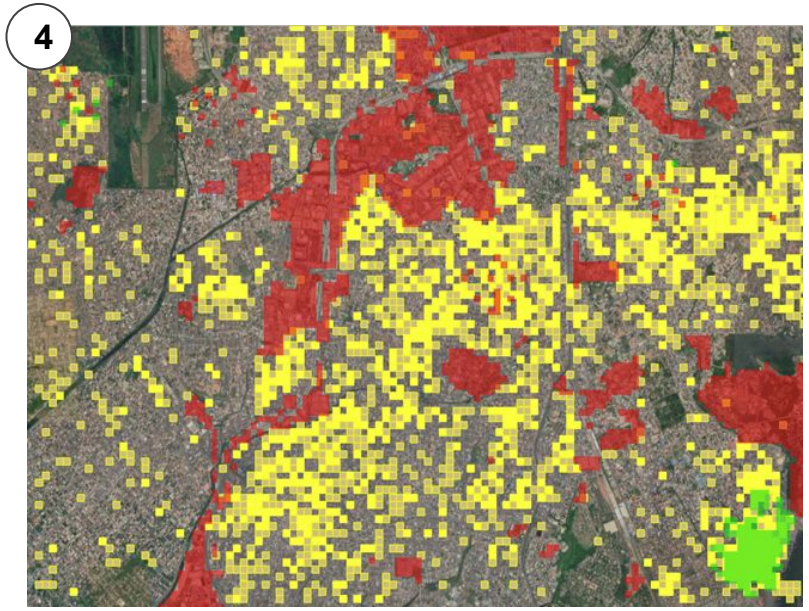


3. Original Map with predictions

High conf. Low conf.

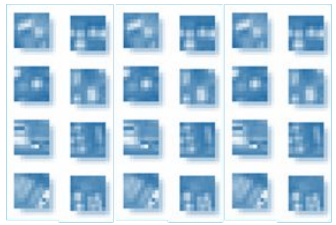


4. All together

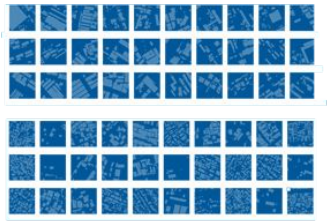


Google Open Building – Hybrid Model

Sentinel-2 Satellite images

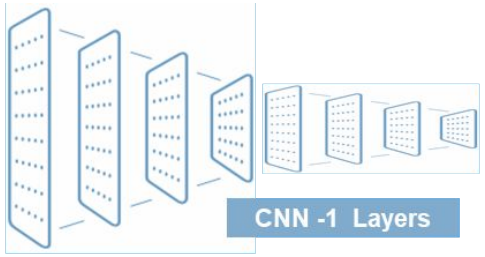


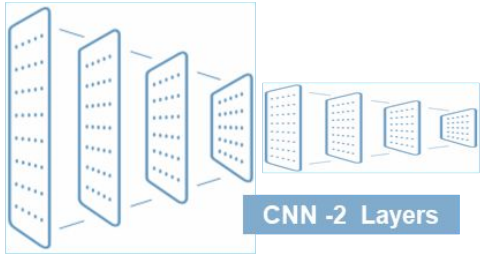
Open building Generated images

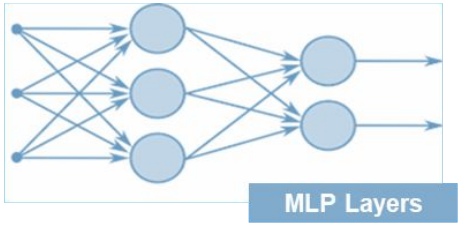


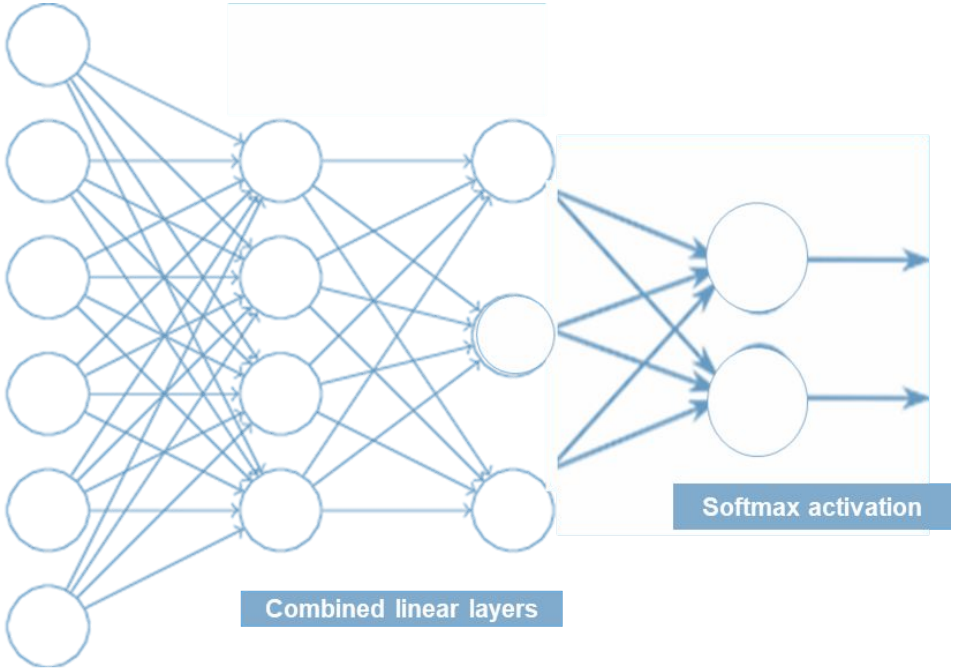
Open building Numerical data

name	long	lat	Label	Point	Mean_Are	Median_A	Building_(M
3	3.27	6.4325	0	15019	87.20067	30.69325	20
4	3.358333	6.6075	0	3450	169.6445	45.9576	12
5	3.095833	6.539167	0	5802	53.48499	23.4971	11
9	3.22	6.718334	0	511	182.1268	103.282	10
16	3.369167	6.45	0	12462	169.3002	36.03175	18
34	3.208333	6.471667	0	9059	160.8063	63.8665	15
54	3.538333	6.681667	0	1217	57.18192	47.964	5
55	3.356667	6.55	0	5290	1871.404	56.6289	5





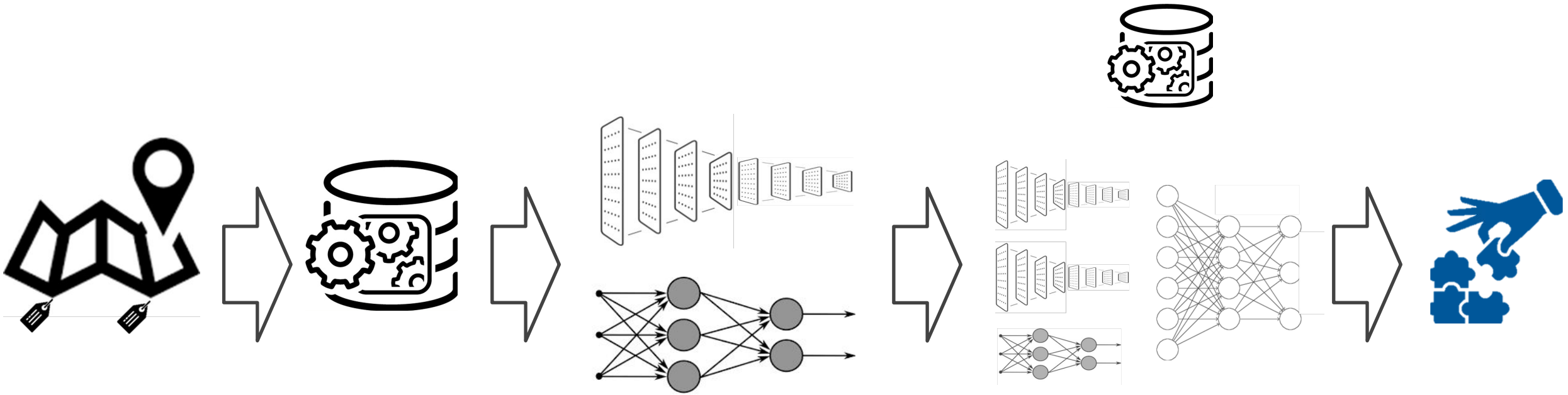




Hybrid Model Results

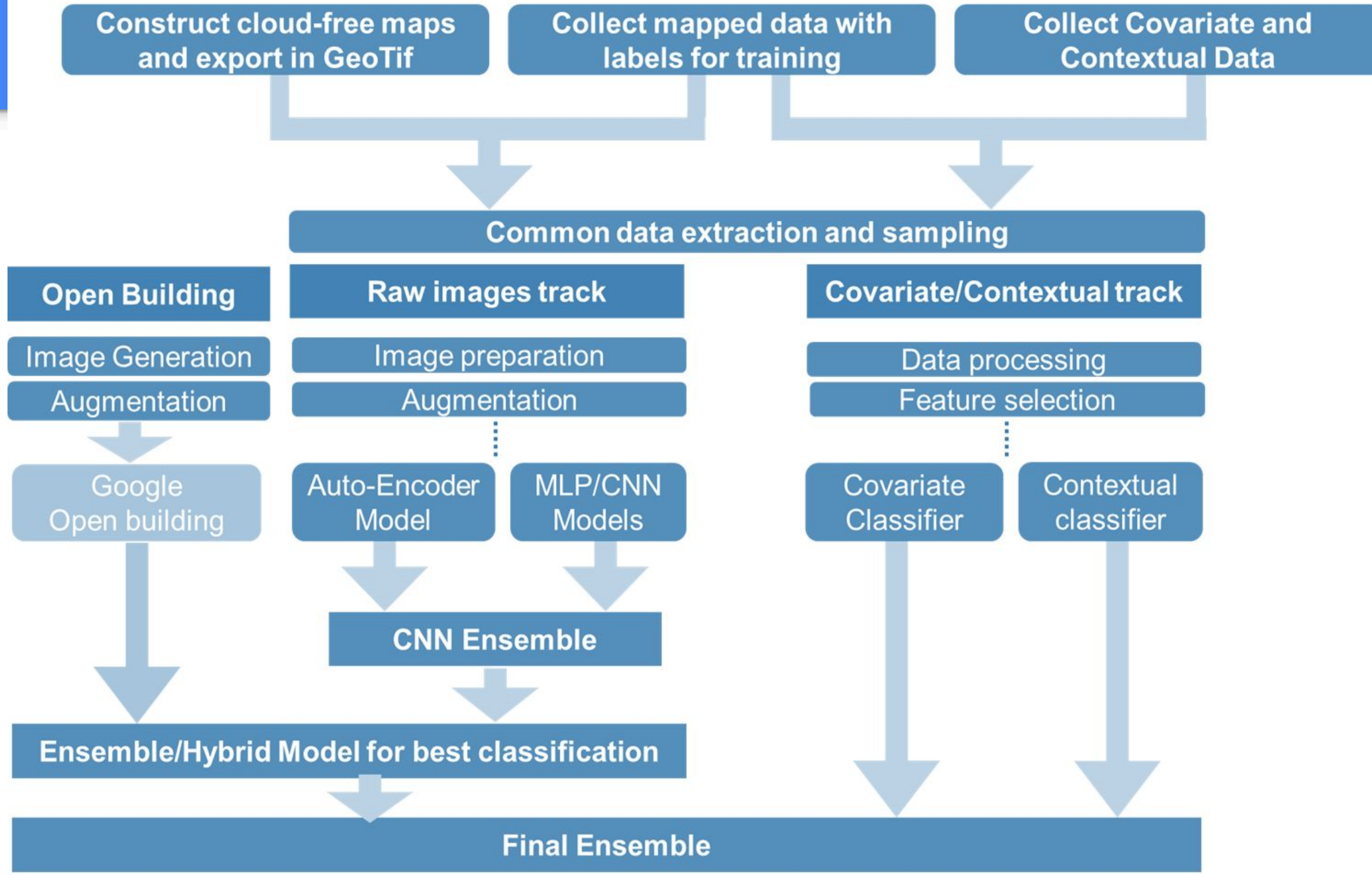
Parameters	Accuracy (%)	Precision (deprived)	Recall (deprived)	F1 score (deprived)
Model -1: {'batch_size: 64', epochs': 37, 'optimizer':'adamax'} **Raw-CNN**: {'Conv2D': (100, 120, 150), 'activation': 'relu', 'Dense_n_neurons: (100)}' **OB_CNN**: {'Conv2D': (100, 150, 200,200), 'activation': 'relu', 'Dense_n_neurons: (100)}' **MLP**: ('n_neurons': (20,15,10), 'activation': 'relu'} **MLP_Concatenated**: ('n_neurons': (210,50,25, 10), 'activation': 'relu'}	84	0.12	0.85	0.21
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

Agenda



The Data | Processing | Modeling | Results 1 | Modeling | Results 2 | Conclusion

Summary


Construct cloud-free maps and export in GeoTif
Collect mapped data with labels for training
Collect Covariate and Contextual Data
Common data extraction and sampling
Open Building
Raw images track
Covariate/Contextual track
Image Generation
Augmentation
Image preparation
Augmentation
Data processing
Feature selection
Google Open building
Auto-Encoder Model
MLP/CNN Models
Covariate Classifier
Contextual classifier
CNN Ensemble
Ensemble/Hybrid Model for best classification
Final Ensemble

Conclusion

- Machine and Deep Learning could be leveraged to obtain better solutions for GIS applications

- Utilizing non expensive computational power, open source low resolution satellite imagery and Licensed free datasets. It was found that acceptable performance was achieved compared to other research conducted using high resolution imagery.

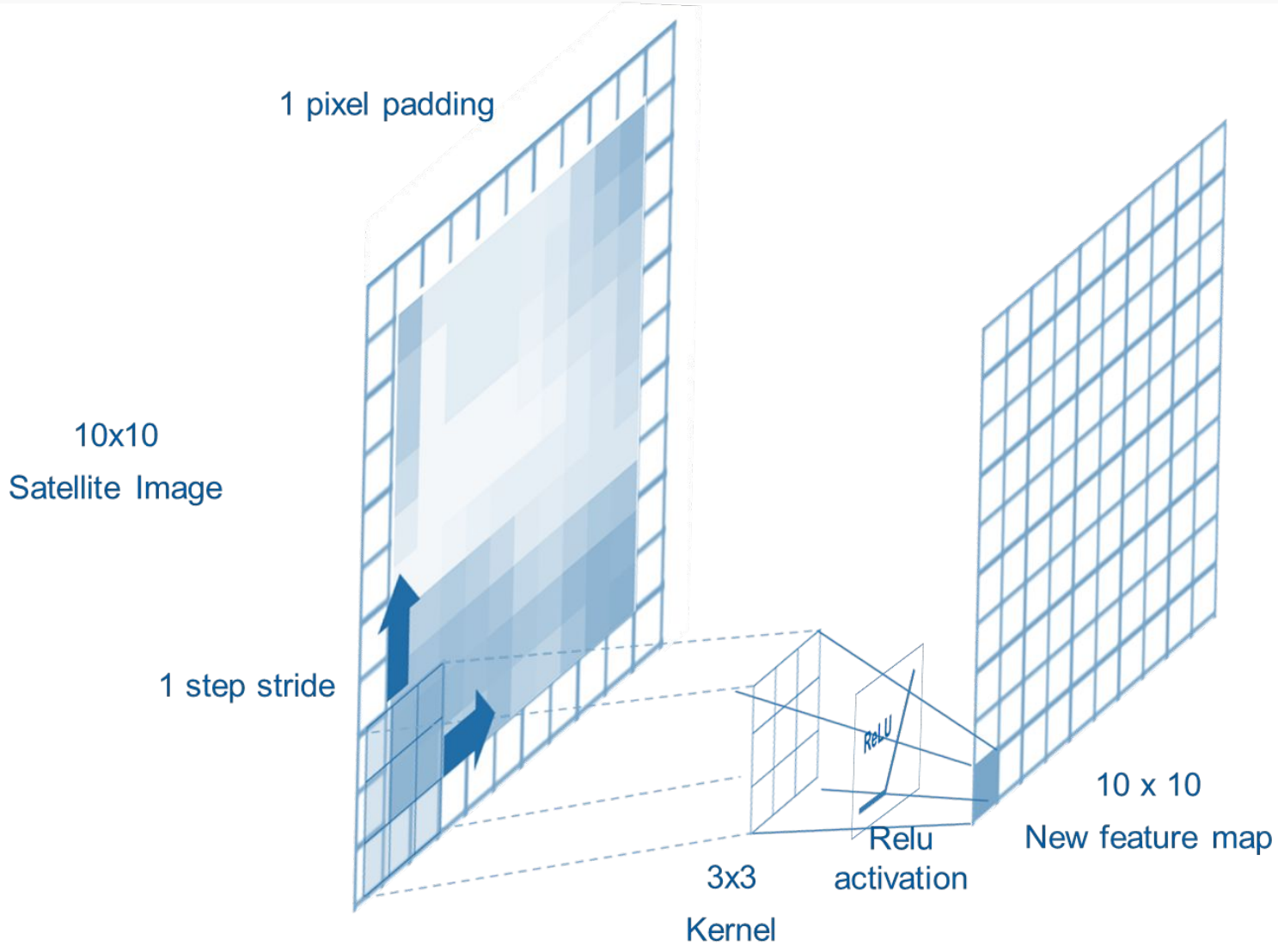
Image Source	**Sentinel - 2**
Total Deprived labeled images	**260 and reaching 8000 with augmentation**
Image size for classification	**10x10 pixels covering 100 square meters**
Best performance reached	**Macro AVG F1 score of 0.78**

Future work

- Acquiring labeled data is a lengthy and costly process. This resulted in creating imbalance classes. Overcoming this challenge using different sampling techniques could be researched.
- Auto-Encoder can be a powerful tool to generate more images for imbalanced classes
- Incorporate other data such as contextual and covariate data during model development.
- Asses the use of trained models to be used for transfer learning to other cities.

THANK YOU

CNN Details


1 pixel padding
10x10
Satellite Image
1 step stride
3x3
Kernel
Relu
activation
10 x 10
New feature map

Deep Neural Network – Grid Search

- Grid Search is configured to divide Hyper Parameters (Epochs, Neurons, LR)
- It was noticed that increasing the number of neurons, decreasing learning rate , and increasing number of epochs during training generally resulted in better performance

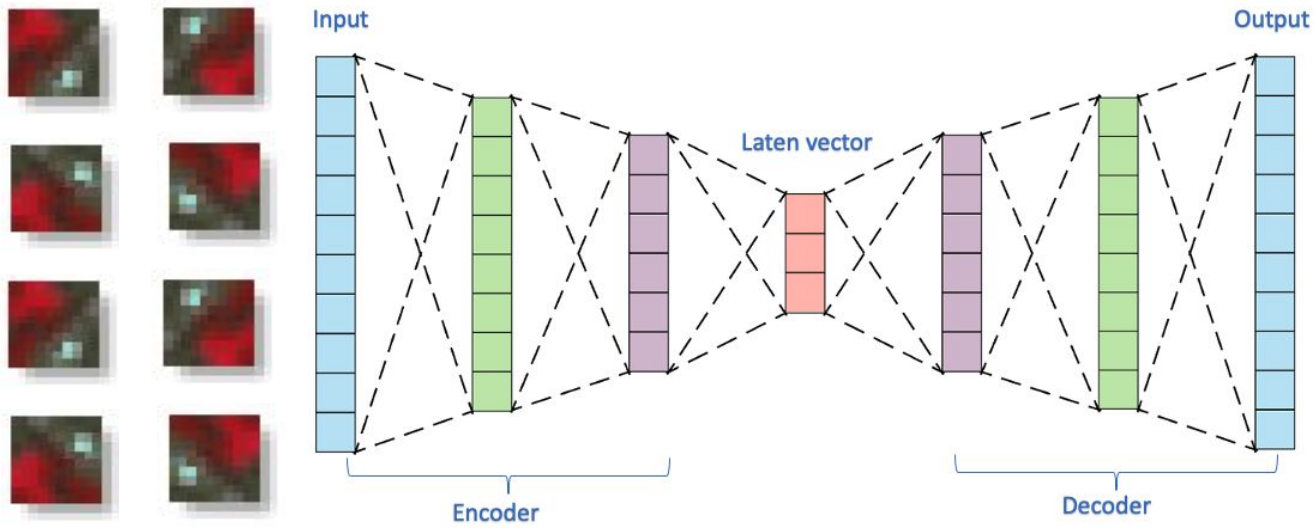
Grid Search

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)}

Best Models

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

Deep Neural Network – Auto-Encoder training and generation of new images



1. Build an Auto-Encoder model with input and output size according to Satellite Image requirements
2. Train the model on all satellite images
3. Use the trained Auto-Encoder to generate more images from the minority class

- 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