THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

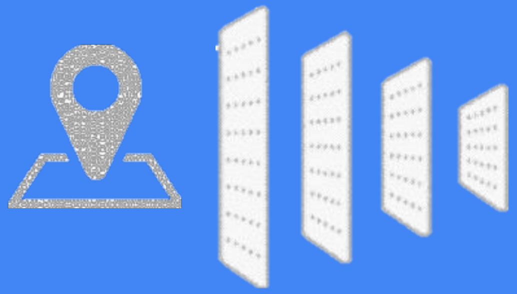
Using Deep Learning to Map Deprived Areas

Project Team:

Abdulaziz Gebril,
Mina Hanna,
Mojahid Osman

Supervised by:

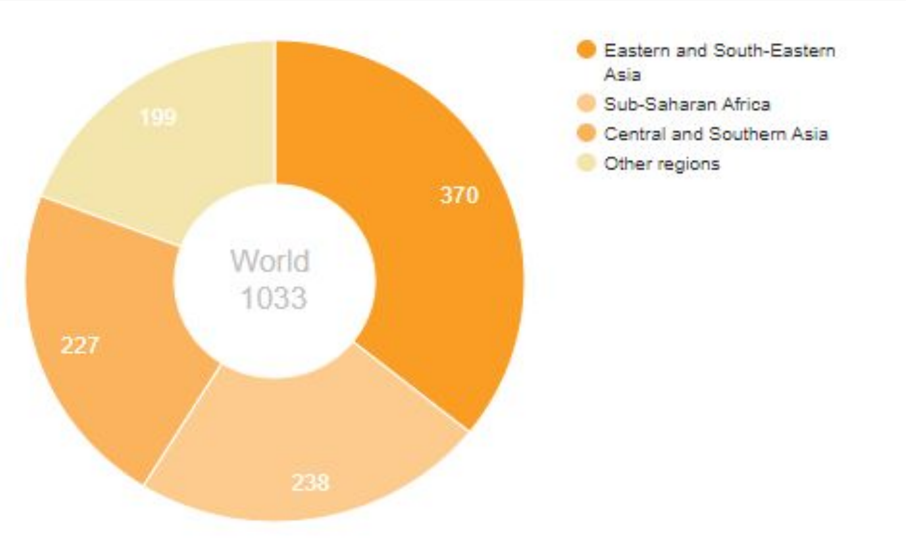
Prof. Amir Jafari



The Problem

- According to the UN,[1] more than 1 billion people are living in deprived areas

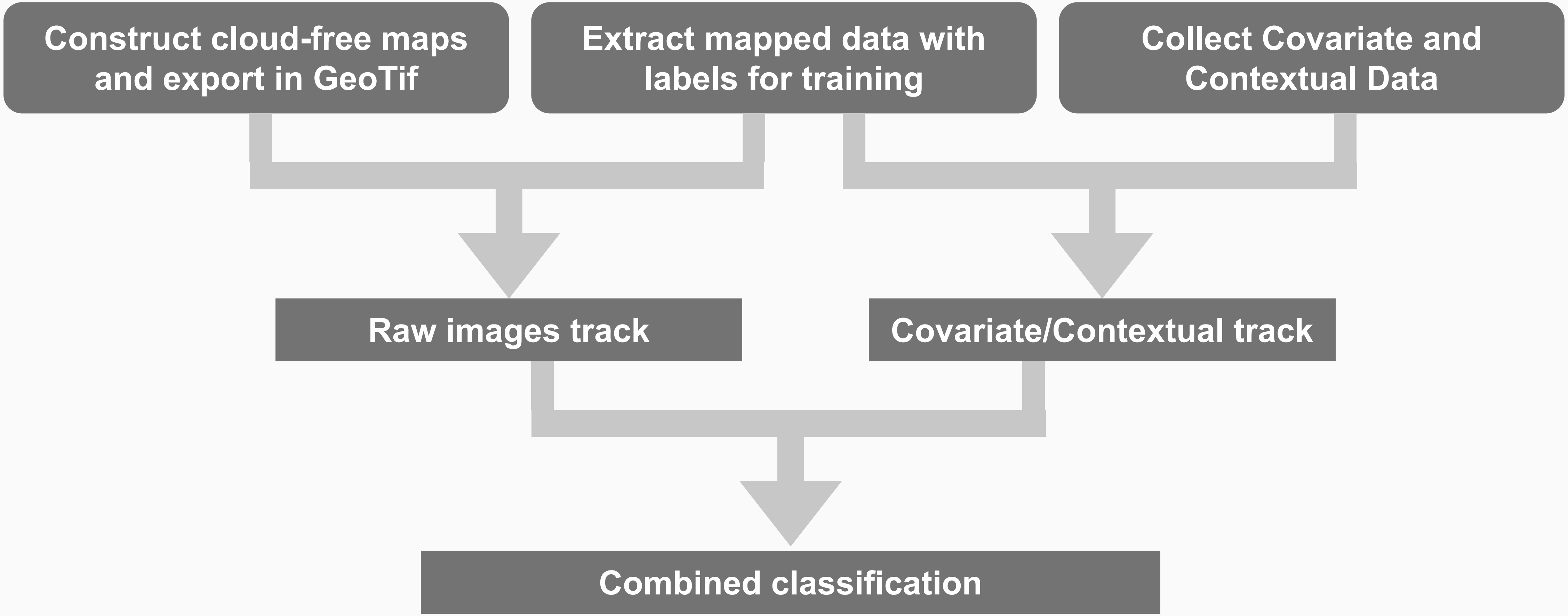
- Policy makers, government and global organizations are seeking detailed identification of deprived areas to enhance assignment of resources and track progress of development projects



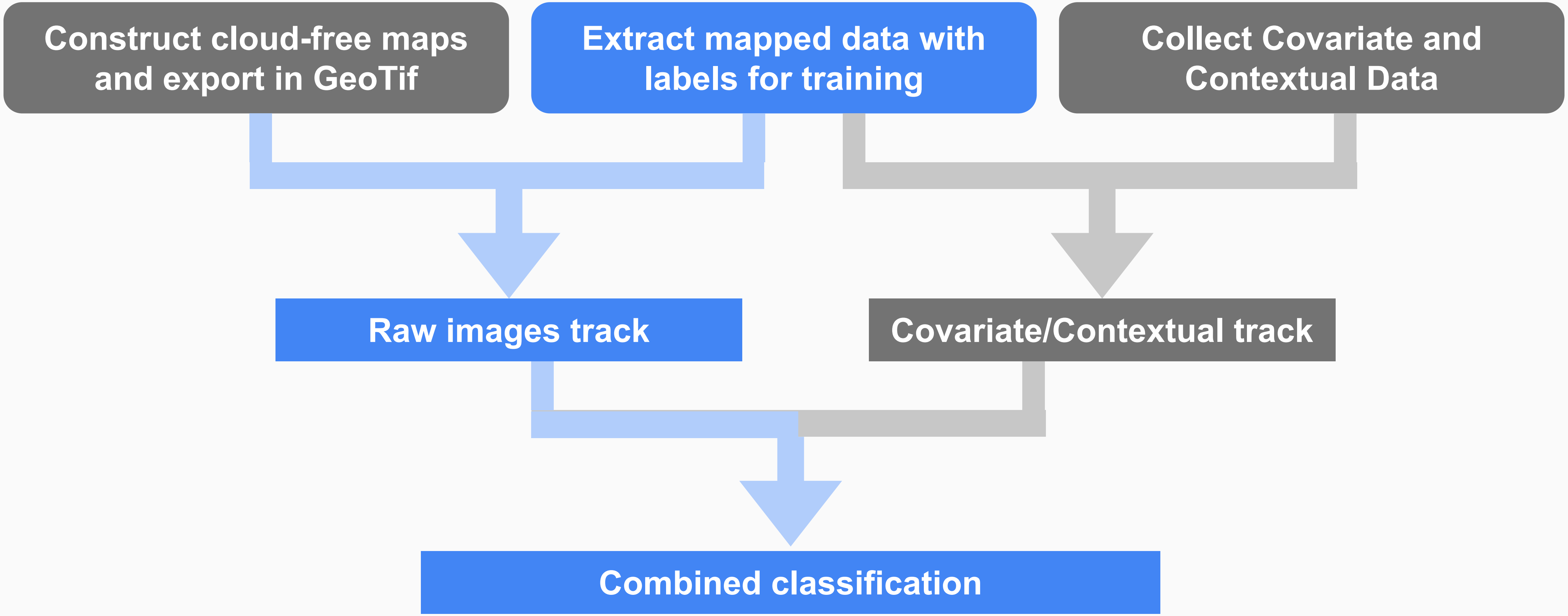
This project aims to process open source geospatial data and utilize various Computer Vision and Machine Learning techniques to help identifying deprived areas while using open source data and accessible satellite images

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

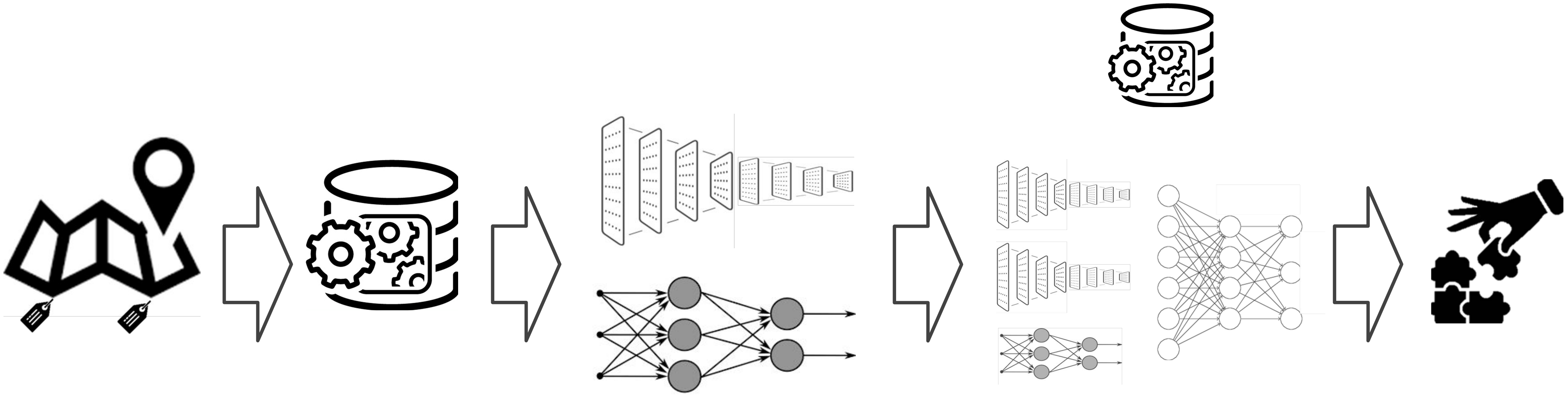
The project big picture



Focus area for this presentation



Agenda



The Data

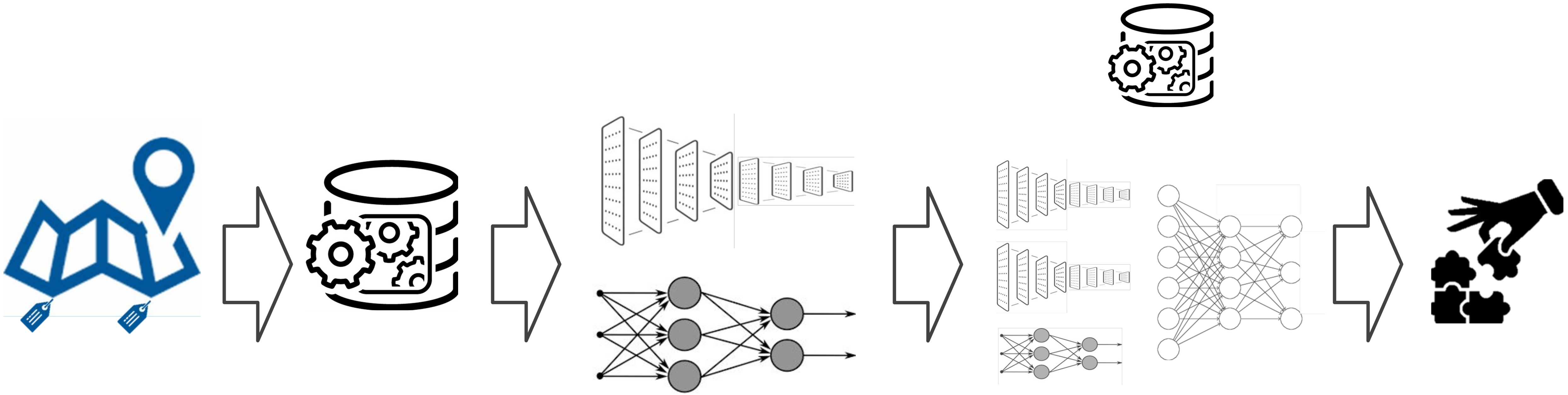
Processing

Modeling | Results 1

Modeling | Results 2

Conclusion

Agenda

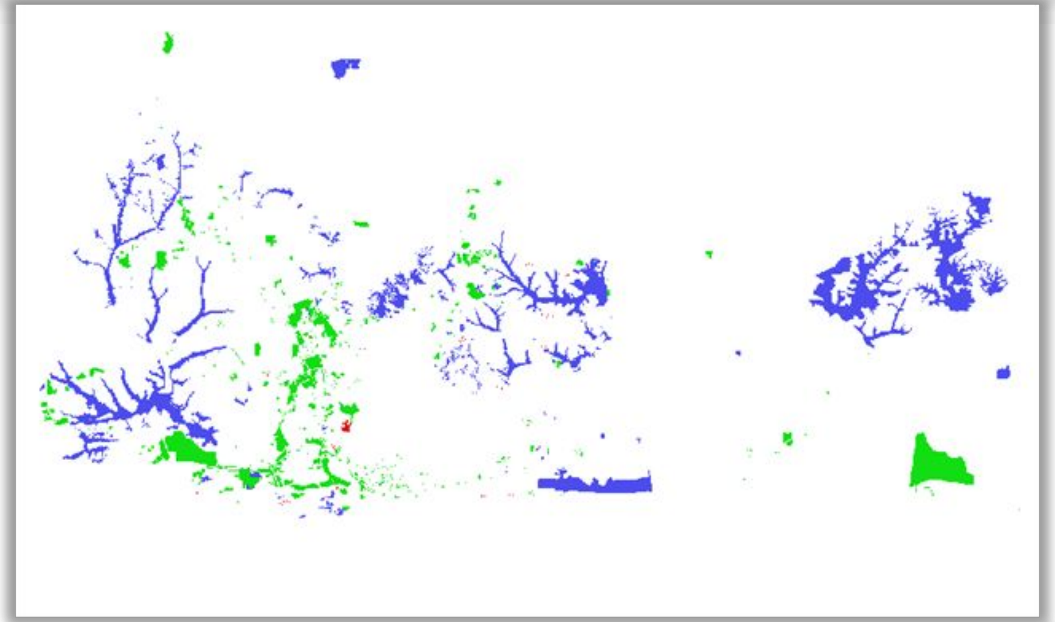


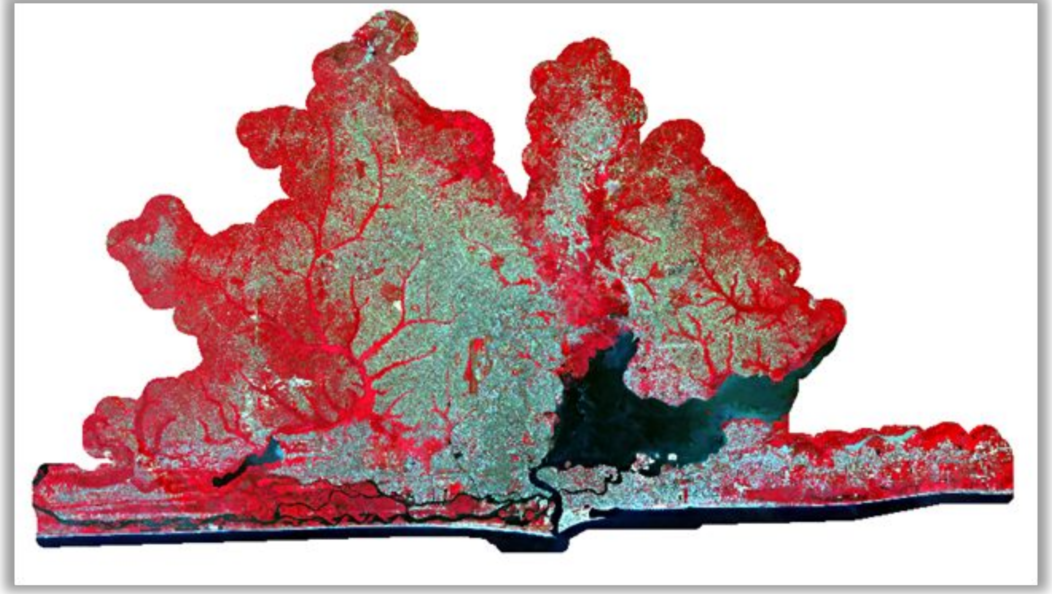
The Data | Processing | Modeling | Results 1 | Modeling | Results 2 | Conclusion

The Data

All labled data was collected with joint efforts led by Idea Maps Network. IdeaMaps network mapped 100x100 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

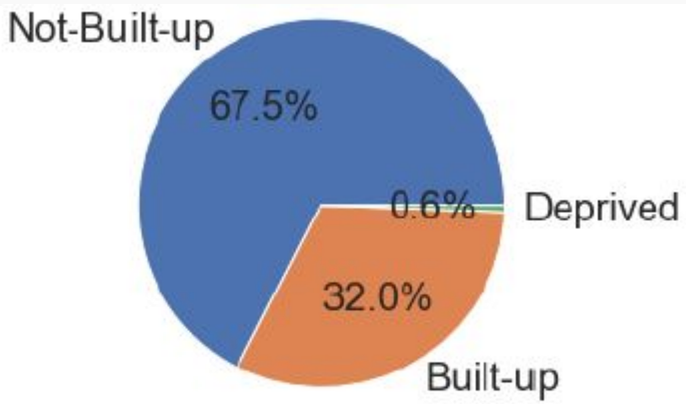
The data is the two following GeoTiff files:
1- Map image extracted from Google earth engine (which is another track of our project)
2- Labeled Tiff image contains labeled areas





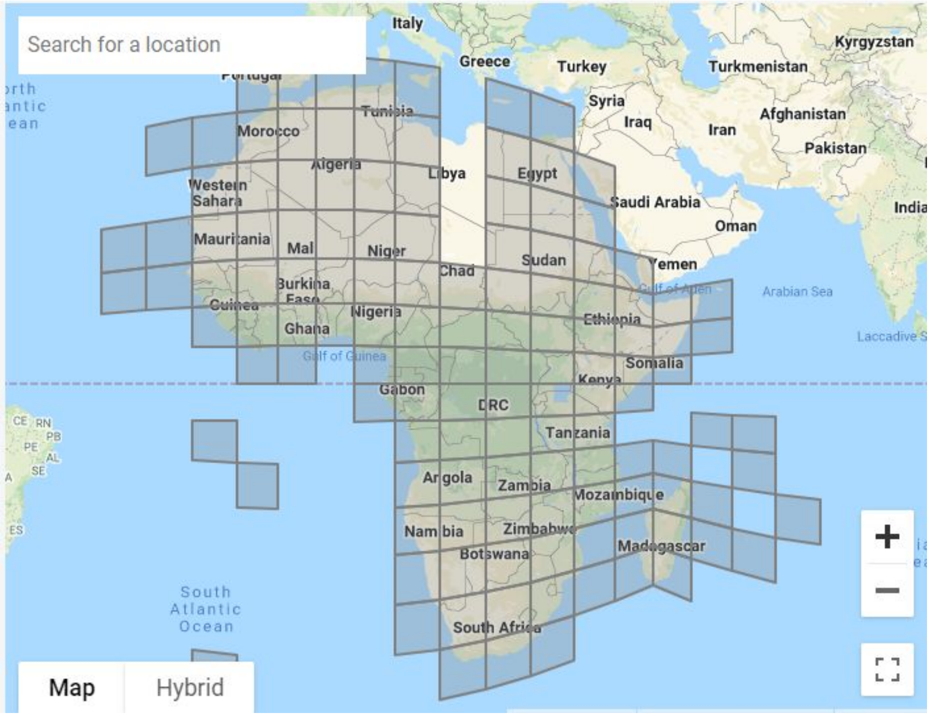
Key Findings about the data

- 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
- 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



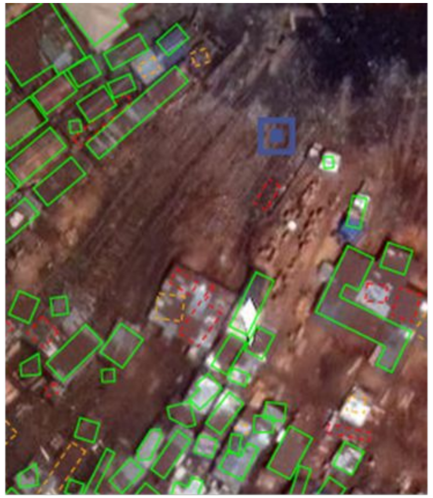
Google Open Building

- 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).

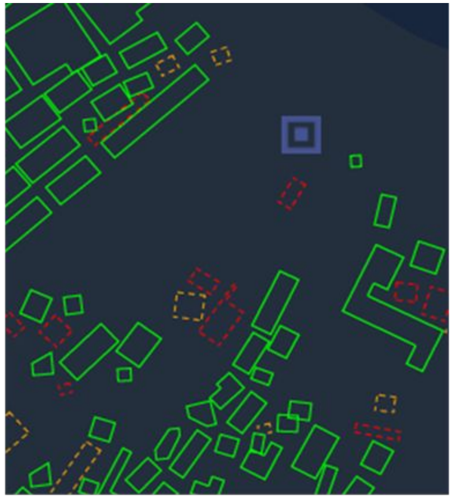


Google Open Building – Data Description

- 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**:



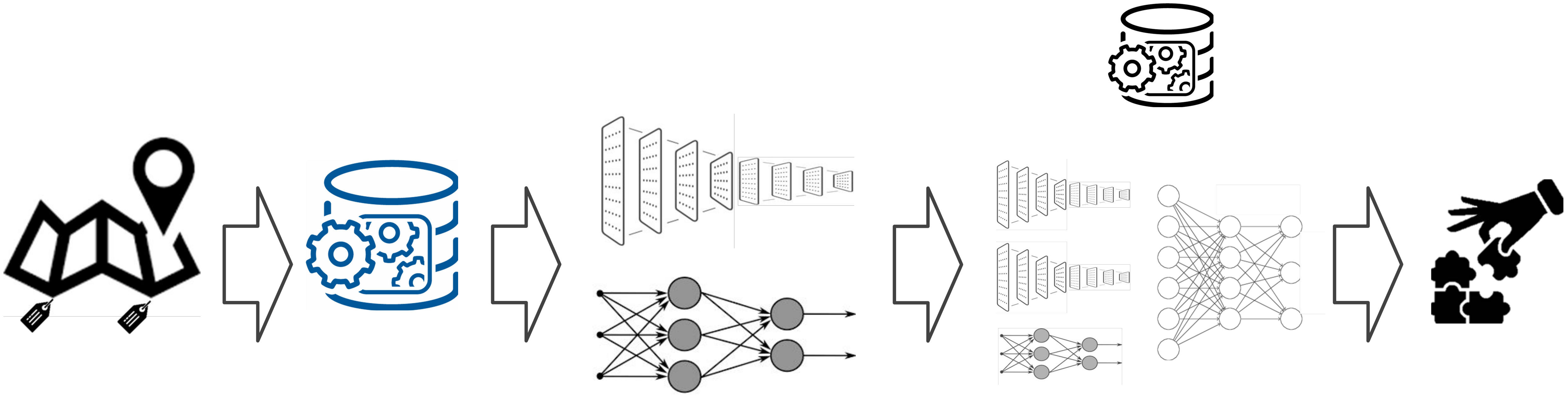
Hybrid map from google



Building map from google

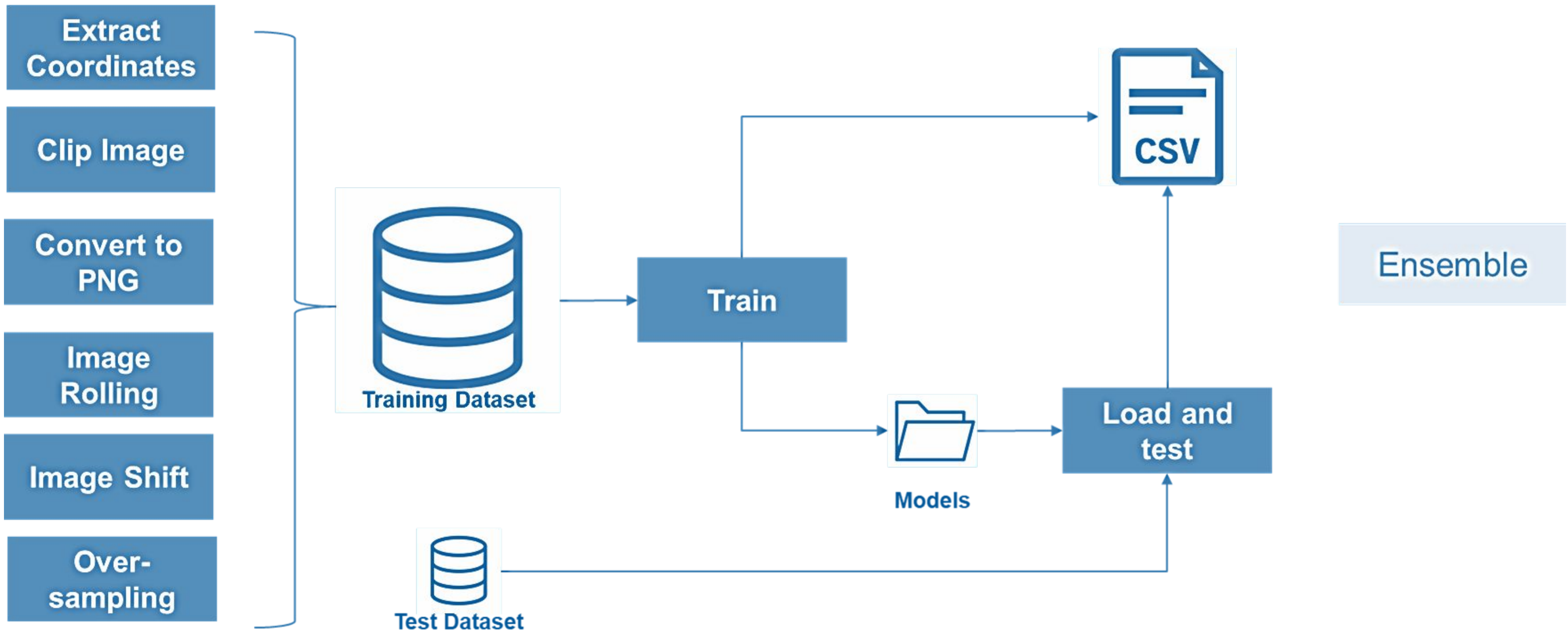
latitude	longitude	area_in_meters	confidence	geometry	full_plus_code
7.80710352	3.89791178	17.1779	0.6448	POLYGON((3.89794037826445 7.80711403192458, 3.89788512265294 7.80711822	6FV5RV4X+R5V2
6.60130215	3.61289427	9.5202	0.6181	POLYGON((3.61290983036891 6.60128999343064, 3.61290792883939 6.60131638	6FR5JJ27+G5C4
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

Agenda



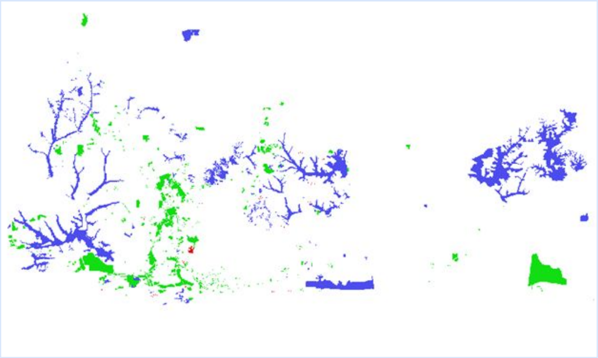
The Data | **Processing** | **Modeling | Results 1** | **Modeling | Results 2** | **Conclusion**

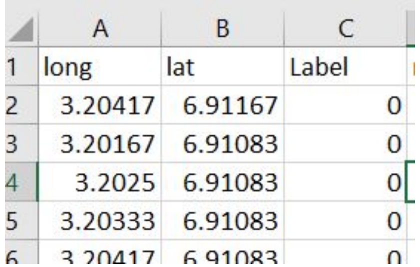
The Big Picture



Data Processing Steps

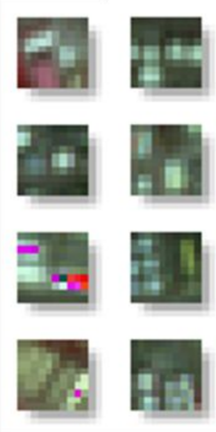
1 Extract Data





	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

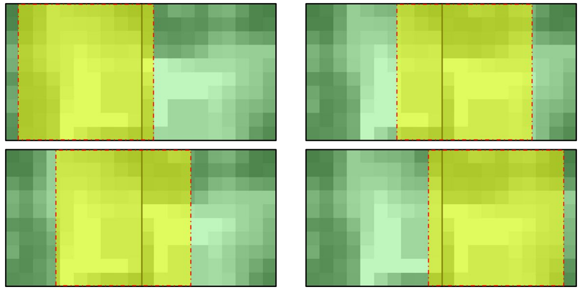


Labeled

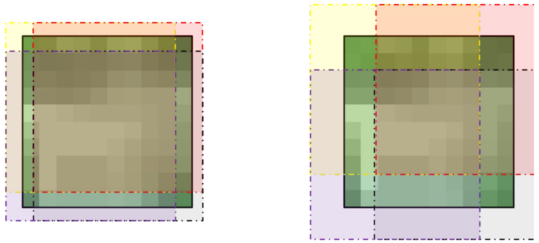
Data Processing Steps

3 Image Augmentation

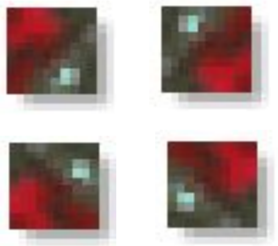
Roll through adjacent images



Shift image by one or two pixels



Rotate all images three times



4 Conversion to PNG

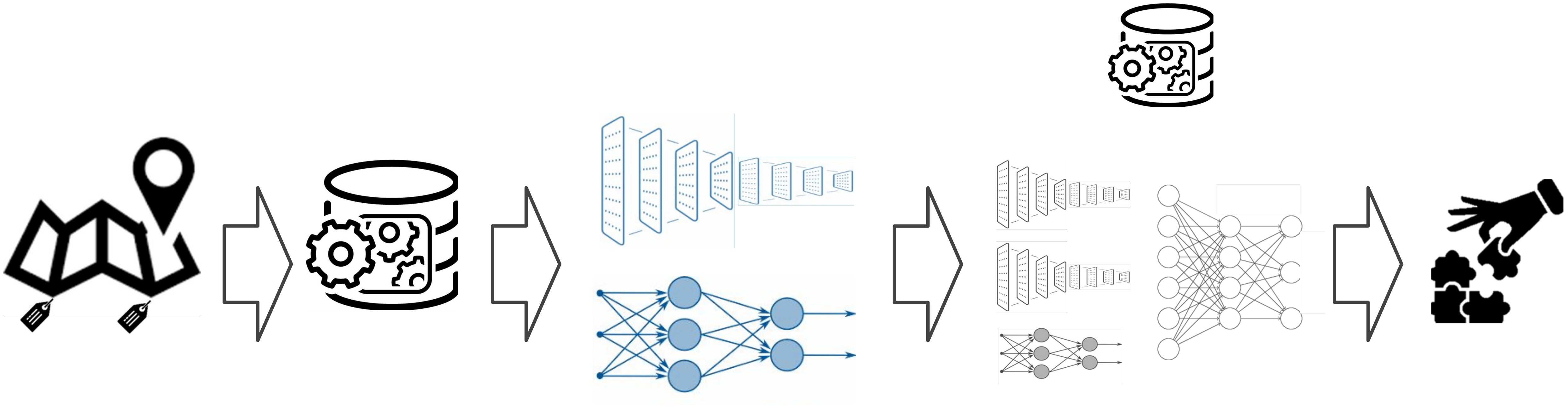
- Using NIR band
- Normalize pixel values
- Follow naming convention

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

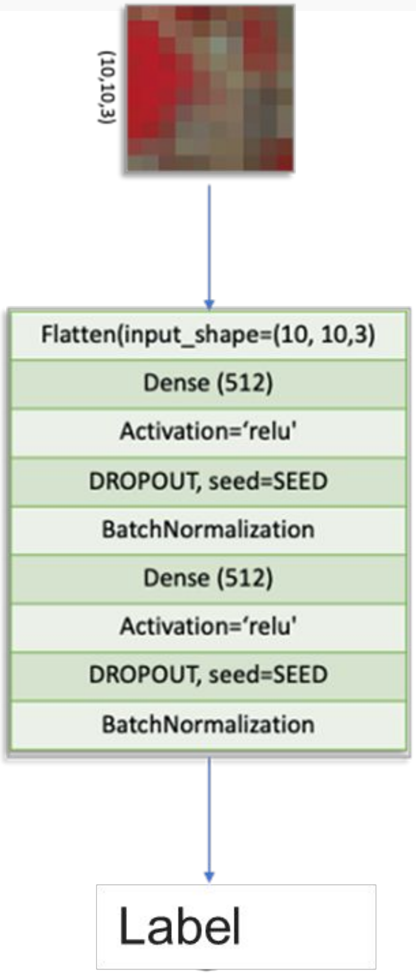
- Generation of more than 8000 labeled images from minority class (~215 images)
- Using of GDAL and Rasterio libraries to process geo-tiff files
- Using QGIS to validate data and overlay different images
- Data was split using a stratified random sampling to maintain ratios

Agenda

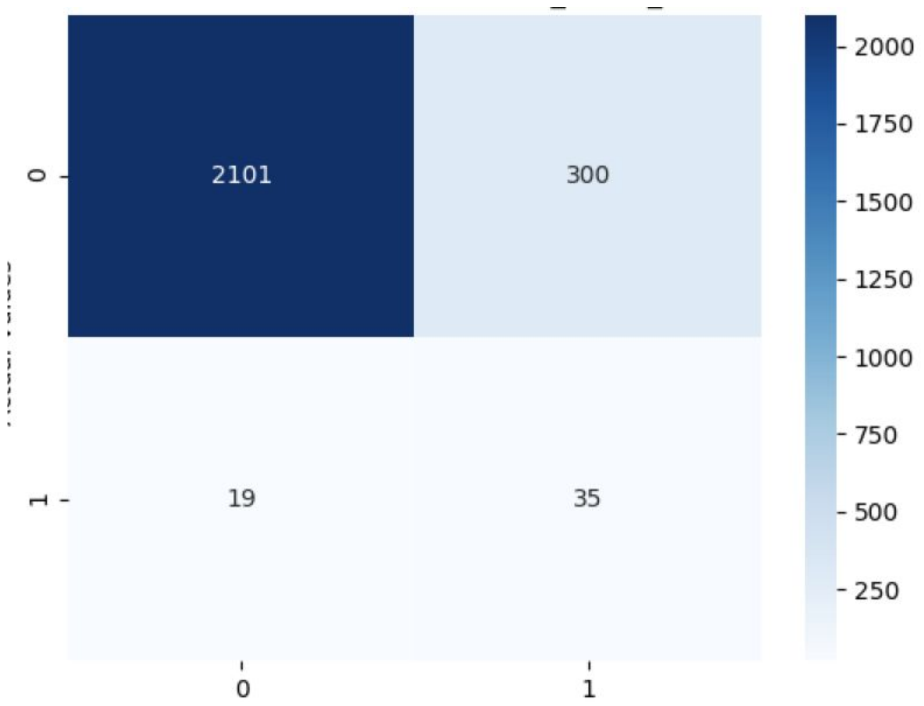


The Data | Processing | **Modeling | Results 1** | Modeling | Results 2 | Conclusion

Deep Neural Network – MLP

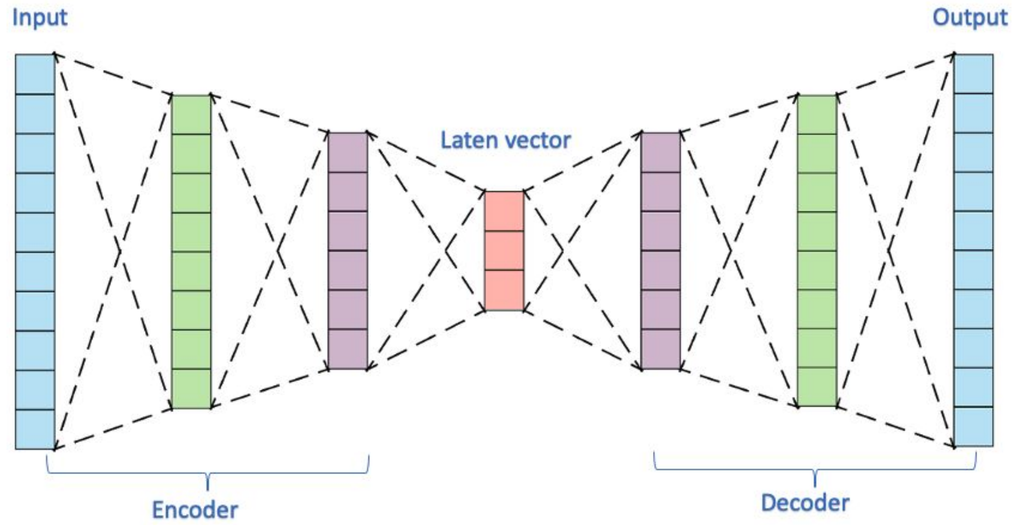


Model	Optimizer	Accuracy	Precision	Recall	F1 score	Cohen Kappa
Model -1	RMSProp	0.952	0.090	0.12963	0.106	0.083
Model -2	Adam	0.866	0.074	0.4444	0.127	0.093
Model -3	Adamax	0.840	0.081	0.32111	0.144	0.087

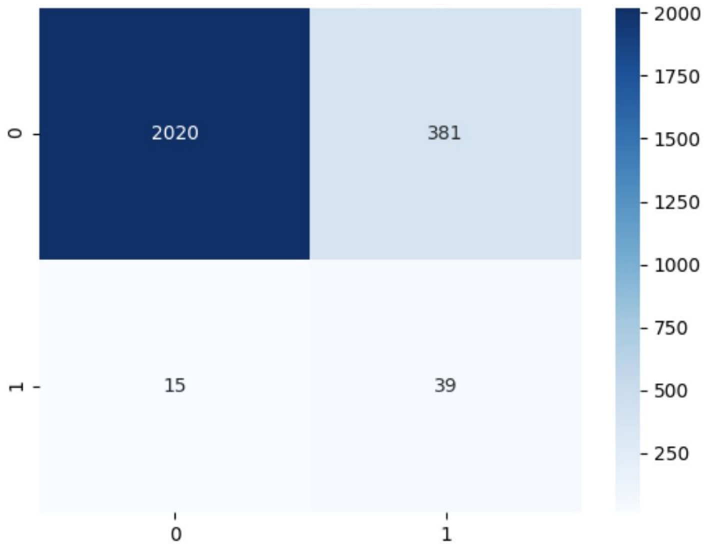


- All models were trained for 200 epochs with early stopping

Deep Neural Network – MLP with more images from auto-encoder

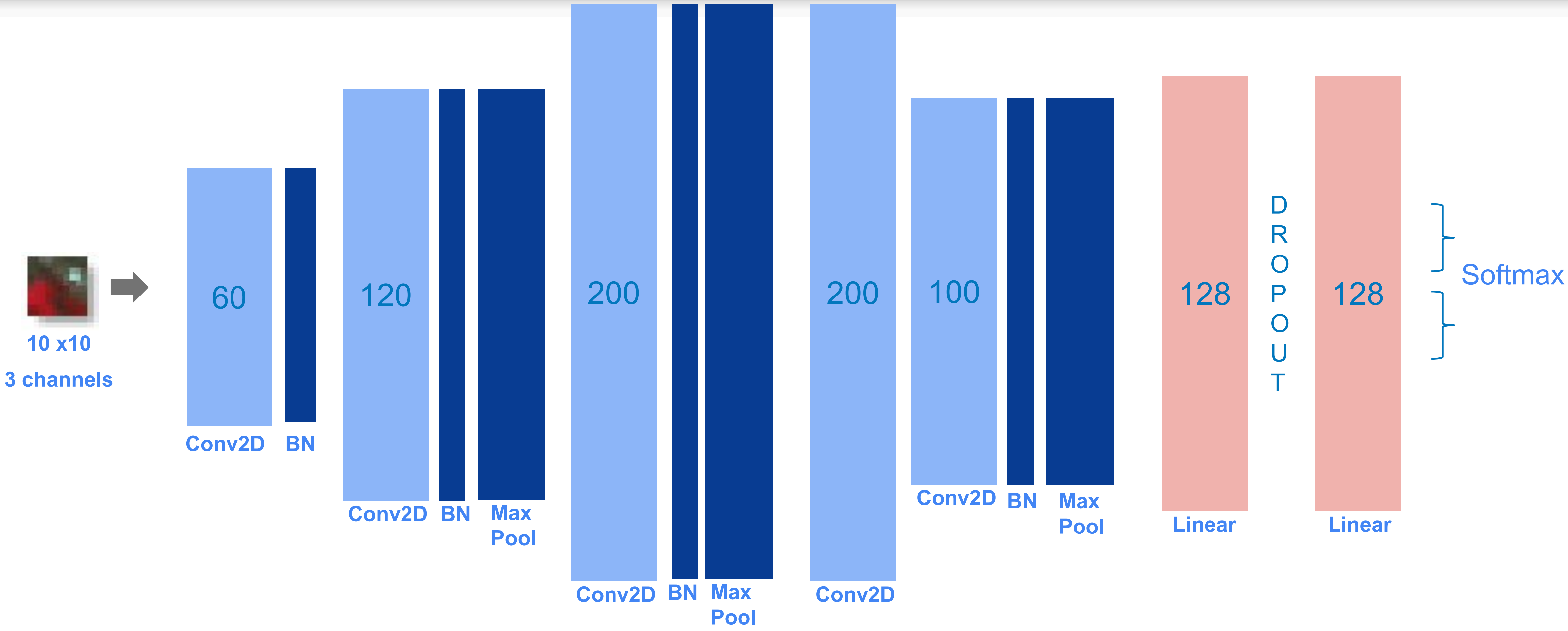


Model	Training Acc	Training loss	Val. Acc	Val. Loss	Test Acc	Precision "1"	Recall "1"	f1 score "1"	MacroAvg Precision	MacroAvg Recall	MacroAvg F1 score
Model-1	0.968	0.0909	0.828	0.9607	0.82	0.07	0.61	0.13	0.53	0.72	0.52
Model-2	0.973	0.0743	0.8222	0.2273	0.82	0.06	0.54	0.11	0.53	0.68	0.51
Model-3	0.966	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



- 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

Deep Neural Network – CNN



- Elimination of label 2 that represents 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

Deep Neural Network – CNN Highlights

- 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)

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)

	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

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

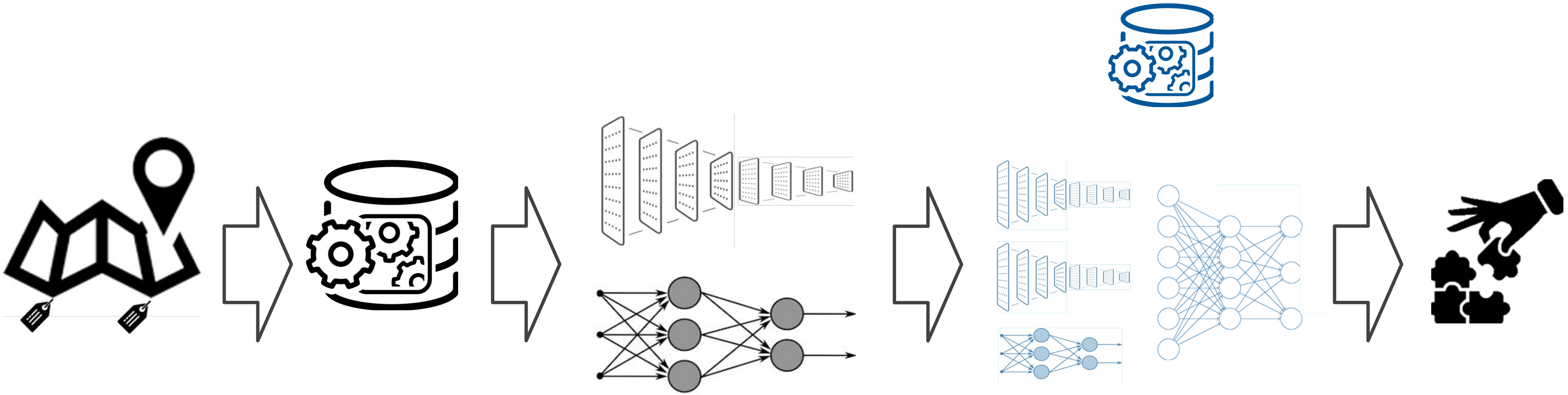
Grid Search

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

Agenda



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

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

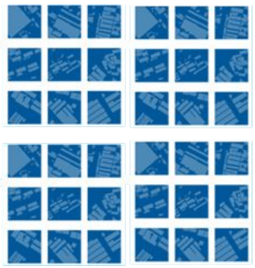
2

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

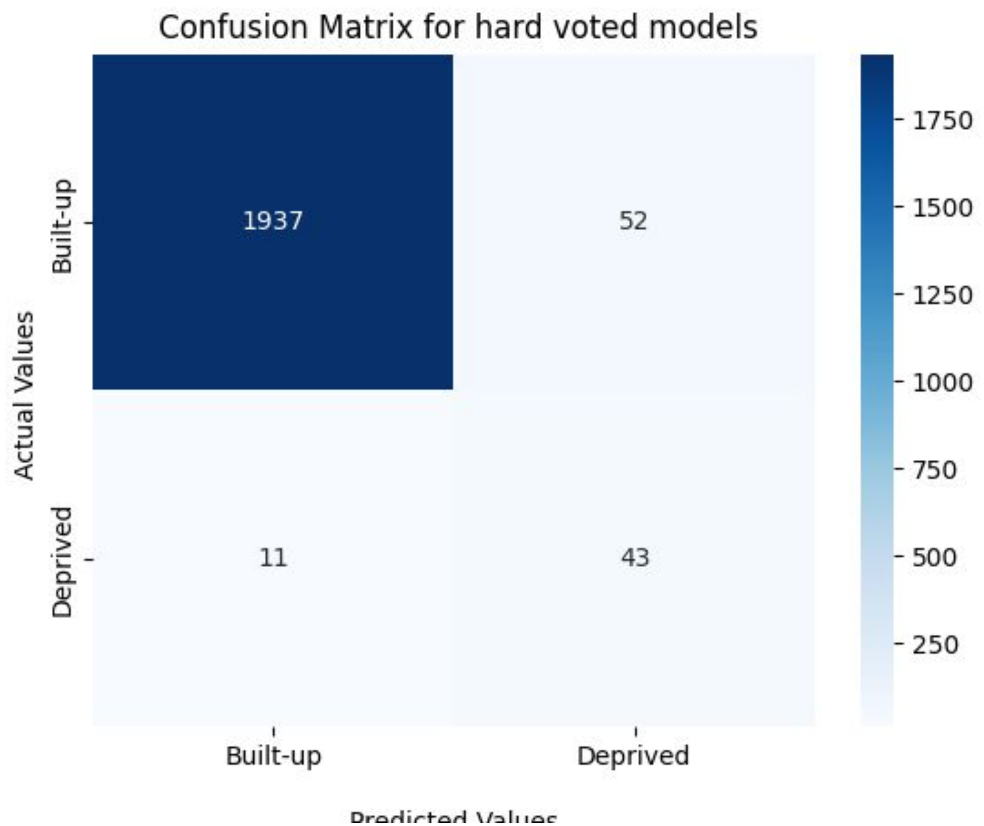
- Using VGG16 with partial layers enabled for training we achieved better results

	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

Using Ensemble Model

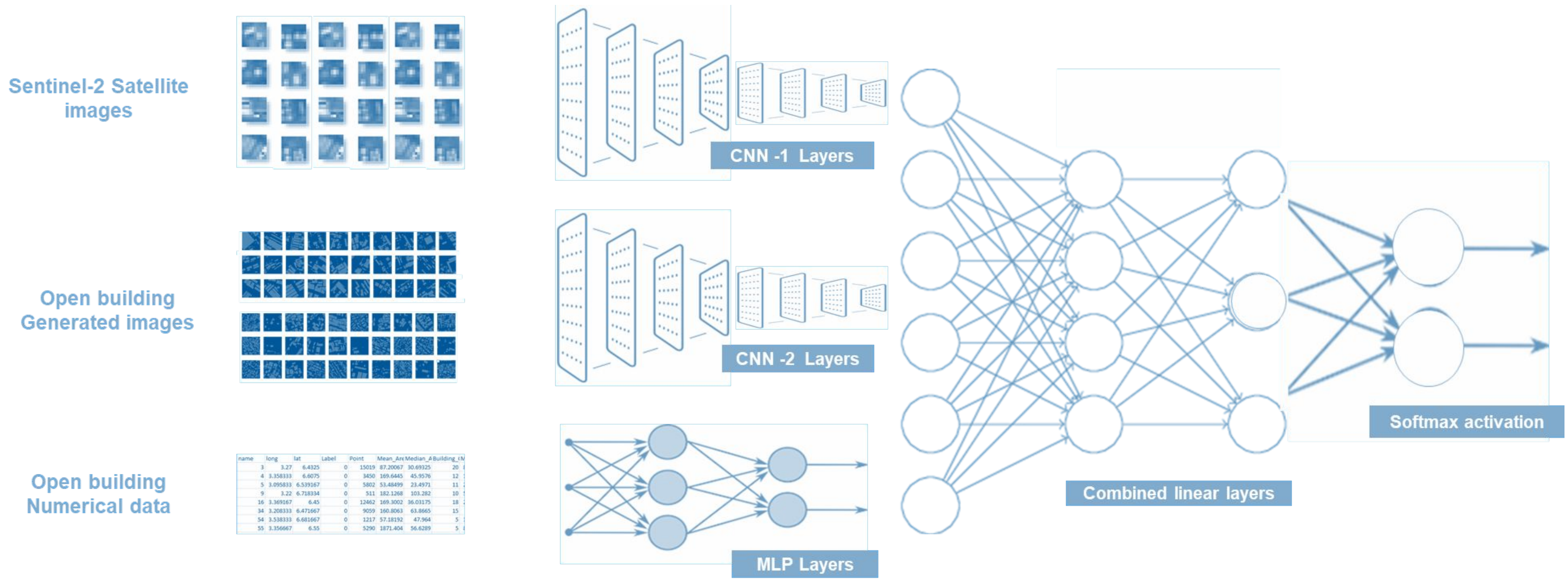
- Best CNN Model -1
- Best CNN Model -1
- Best Pre-Trained CNN
- Best MLP Model
- Best Open Building Model -1
- Best Open Building Model -2
- Best Pre-trained open building

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

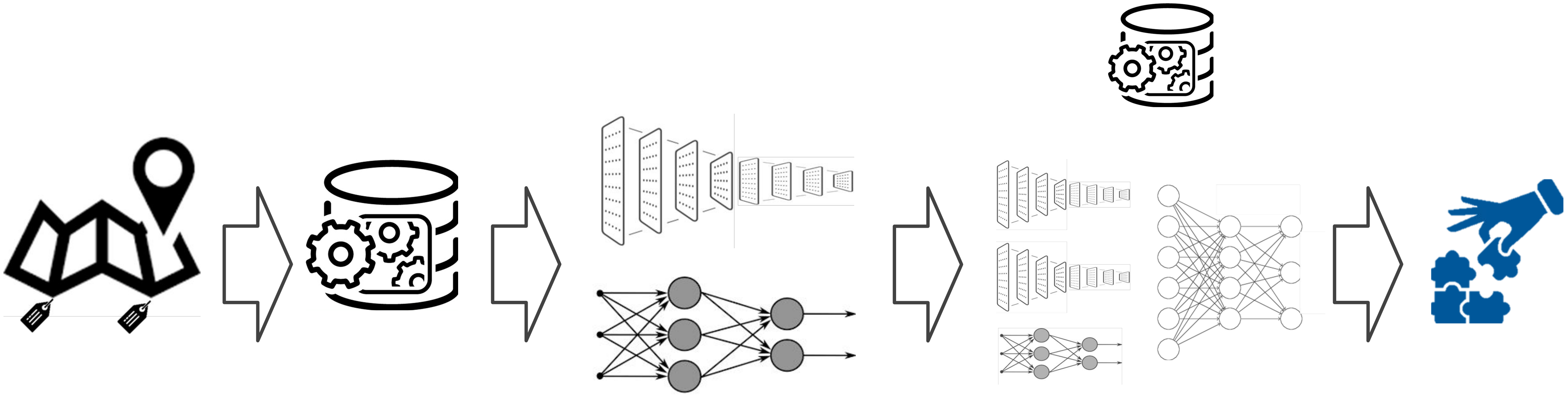
Google Open Building – Hybrid Model



Hybrid Model Results

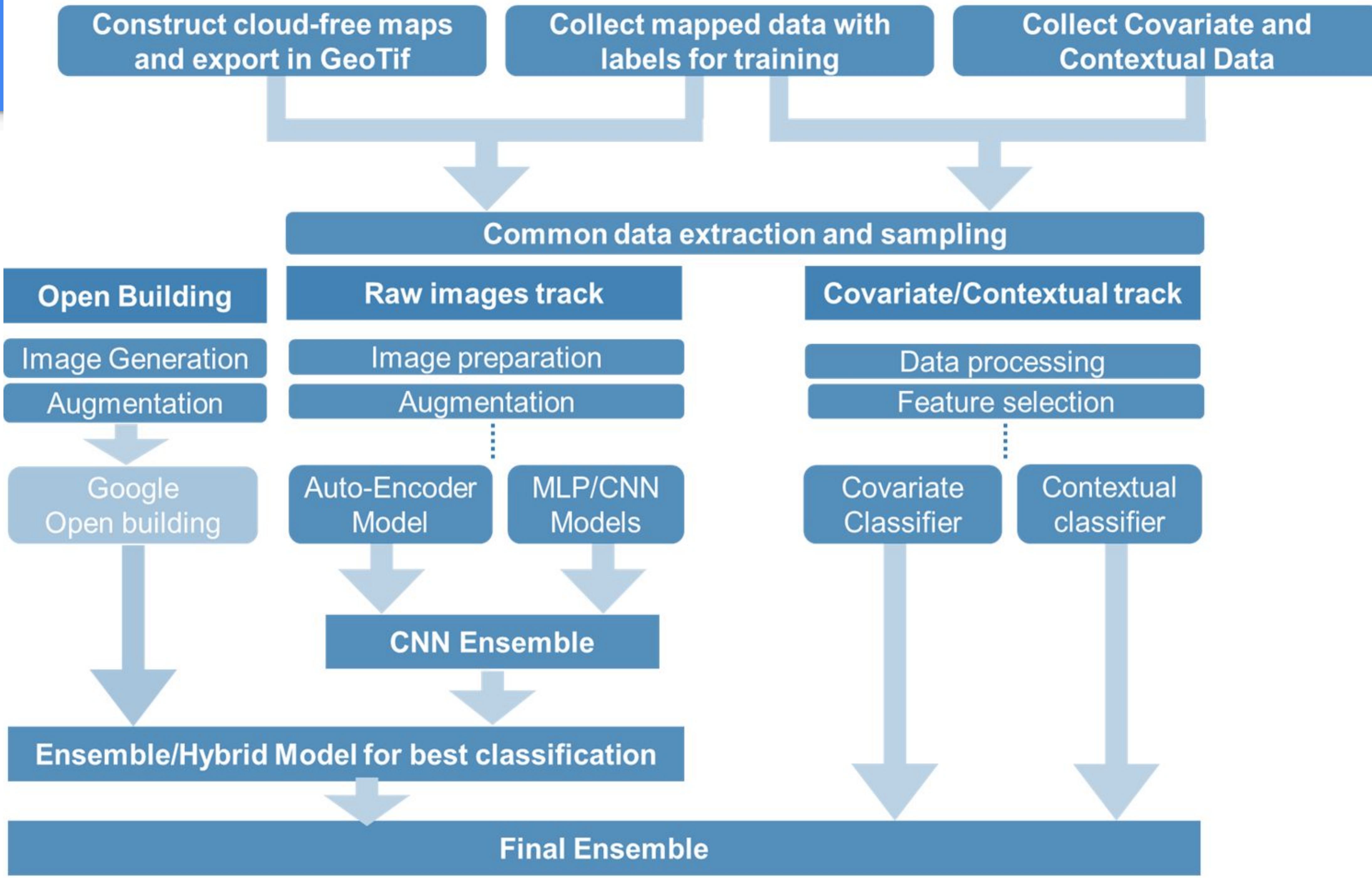
Parameters	Accuracy (%)	Precision (deprived)	Recall (deprived)	F1 score (deprived)
Model -1: {'batch_size: 64', epochs': 37, 'optimizer':'adamax'} **Raw-CNN**: {'Conv2D': (100, 120, 150), 'activation': 'relu', 'Dense_n_neurons: (100)}' **OB_CNN**: {'Conv2D': (100, 150, 200,200), 'activation': 'relu', '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
Image Generation
Augmentation
Google Open building
Raw images track
Image preparation
Augmentation
Auto-Encoder Model
MLP/CNN Models
CNN Ensemble
Covariate/Contextual track
Data processing
Feature selection
Covariate Classifier
Contextual classifier
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.
- 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.
- 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.