Building Height Detection in Underdeveloped Regions
Prof. Sudipan Saha

Tushar Verma (2022AIY7514)
ScAI, IIT Delhi

1 Introduction

Urbanization in underdeveloped regions is rapidly increasing, yet accurate and up-to-date data on building heights is often lacking. Traditional methods for building height detection, such as LiDAR and manual surveys, are expensive, labor-intensive, and often unavailable in these regions. Accurate building height data is essential for effective urban planning, helping governments and organizations optimize land use, design infrastructure, and plan for future growth. This research aims to develop novel methods and models for building height detection, contributing to the broader field of remote sensing and computer vision.

2 Background

The 2023 IEEE GRSS Data Fusion Contest [1] focused on advancing research in urban building classification and 3D reconstruction, particularly emphasizing building height estimation. The contest provided participants with a large-scale dataset covering 17 cities across six continents, making it highly representative of diverse urban forms and architectures. The challenge was to simultaneously extract building footprints and estimate their heights using multi-task learning to enhance the accuracy and efficiency of both tasks.
Recent studies such as "Data Fusion for Multi-Task Learning of Building Extraction and Height Estimation" [2] from the 2023 IEEE GRSS Data Fusion Contest have explored advanced multi-task learning and fusion techniques, integrating optical and SAR data to improve building height estimation and urban mapping.
Another study suggests IM2ELEVATION [3] method that estimates building heights from single-view aerial images using deep learning to generate 3D urban data with minimal input.

3 Datasets

3.1 DFC23 Dataset [4]

Images from the DFC23 dataset were collected from SuperView-1, Gaofen-2, and Gaofen-3 satellites, with spatial resolutions of 0.5m, 0.8m, and 1m, respectively. Normalized Digital Surface Models (nDSMs) were produced from stereo images captured by Gaofen-7 and WorldView with a ground sampling distance (GSD) of roughly 2m.

3.2 Delhi Dataset

The Delhi dataset was collected from Google Earth Pro [5], including data from high-resolution commercial satellites. The data typically covers the urban area of Delhi and its surroundings, providing insights into land use, vegetation, infrastructure, and other features.

4 Challenges in Developing Nations

Implementing height detection in developing nations presents three major challenges:

1. **No Ground Truth:** Lack of ground truth data complicates the validation and training processes.
2. **Unstructured Arrangement of Buildings:** Buildings are closely packed, leading to loss of shadow information.
3. **Building Roofs:** Roofs do not fall under a particular type, making footprint extraction challenging.

5 Proposed Approaches

5.1 Approach 1: Initial Model

The initial model was trained on the DFC23 dataset and tested on the DFC23 (Delhi) and Delhi datasets without domain adaptation. The results highlighted significant domain shifts, leading to performance degradation.

5.1.1 Model Architecture

The model used is a modified U-Net [6] architecture that integrates a VGG16 [7] encoder, a bottleneck, and a decoder for segmentation tasks. Below is a brief description of the model components:

1. **Encoder (VGG16-based):**The encoder is built using the pre-trained VGG16 network with batch normalization, and its layers are divided into blocks.
   The model can handle both 3-channel (RGB) and 4-channel input images.
   It extracts features through five blocks of convolutional layers, gradually reducing the spatial dimensions while increasing the feature depth.
2. **Bottleneck:** After the encoder, a bottleneck block captures the most abstract features from the image. This includes the final layers from the VGG16 model and an additional convolutional block, which outputs higher-dimensional feature maps (1024 channels).
3. **Decoder (Segmentation Decoder):** The decoder uses transposed convolutions (up-convolutions) to upsample the feature maps back to the original image resolution.
   At each upsampling step, the decoder concatenates corresponding feature maps from the encoder (known as skip connections) to preserve spatial information from

the earlier layers.
The final layer reduces the output to a single channel, suitable for pixel-wise predictions like segmentation.

4. **ReLU Activation [8]:** After the decoder, a ReLU activation is applied to the output to ensure positive values, which can be useful for certain types of predictions (e.g., segmentation or regression).

Layer Type	**Layer Name**	**Input Shape**	**Output Shape**	**Parameters**
		Encoder		
Conv2d + BN + ReLU	block1	(N, 3, H, W)	(N, 64, H/2, W/2)	89,600
Conv2d + BN + ReLU	block2	(N, 64, H/2, W/2)	(N, 128, H/4, W/4)	295,424
Conv2d + BN + ReLU	block3	(N, 128, H/4, W/4)	(N, 256, H/8, W/8)	1,180,160
Conv2d + BN + ReLU	block4	(N, 256, H/8, W/8)	(N, 512, H/16, W/16)	4,722,176
Conv2d + BN + ReLU	block5	(N, 512, H/16, W/16)	(N, 512, H/32, W/32)	7,079,936
		Bottleneck		
Conv2d	bottleneck	(N, 512, H/32, W/32)	(N, 512, H/32, W/32)	4,722,176
Conv2d + BN + ReLU	conv_bottleneck	(N, 512, H/32, W/32)	(N, 1024, H/32, W/32)	4,719,616
		Decoder		
ConvTranspose2d + ReLU	up_conv6	(N, 1024, H/32, W/32)	(N, 512, H/16, W/16)	2,097,664
Conv2d + BN + ReLU	conv6	(N, 1024, H/16, W/16)	(N, 512, H/16, W/16)	4,722,176
ConvTranspose2d + ReLU	up_conv7	(N, 512, H/16, W/16)	(N, 256, H/8, W/8)	524,544
Conv2d + BN + ReLU	conv7	(N, 768, H/8, W/8)	(N, 256, H/8, W/8)	1,180,160
ConvTranspose2d + ReLU	up_conv8	(N, 256, H/8, W/8)	(N, 128, H/4, W/4)	131,200
Conv2d + BN + ReLU	conv8	(N, 384, H/4, W/4)	(N, 128, H/4, W/4)	295,168
ConvTranspose2d + ReLU	up_conv9	(N, 128, H/4, W/4)	(N, 64, H/2, W/2)	32,832
Conv2d + BN + ReLU	conv9	(N, 192, H/2, W/2)	(N, 64, H/2, W/2)	73,856
ConvTranspose2d + ReLU	up_conv10	(N, 64, H/2, W/2)	(N, 32, H, W)	8,224
Conv2d	conv11	(N, 32, H, W)	(N, 1, H, W)	33
ReLU	relu	(N, 1, H, W)	(N, 1, H, W)	0

Table 1: Summary of the UNet model layers and parameters with partitions.

5.1.2 Results of Approach-1

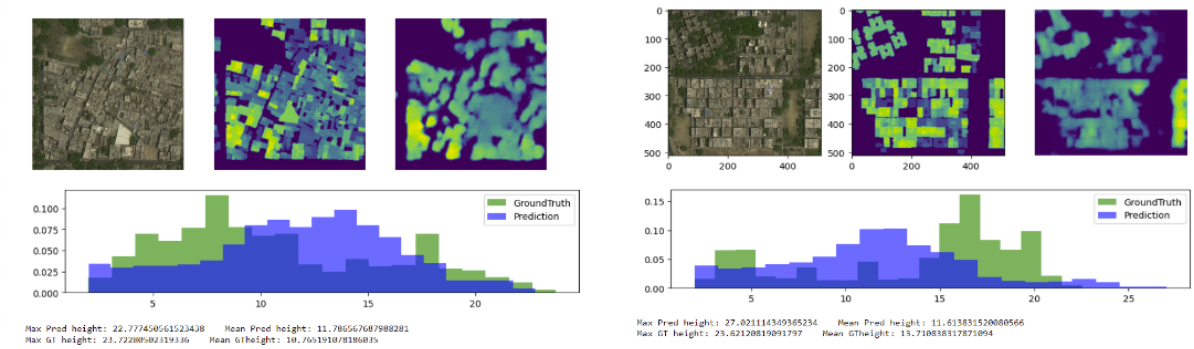

Figure 1: Performance on DFC Dataset

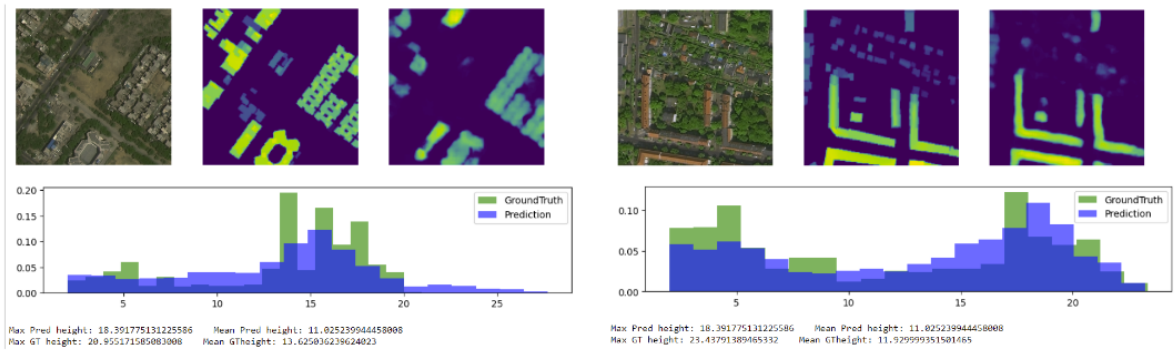

Figure 2: Performance on Delhi's images (DFC Dataset)

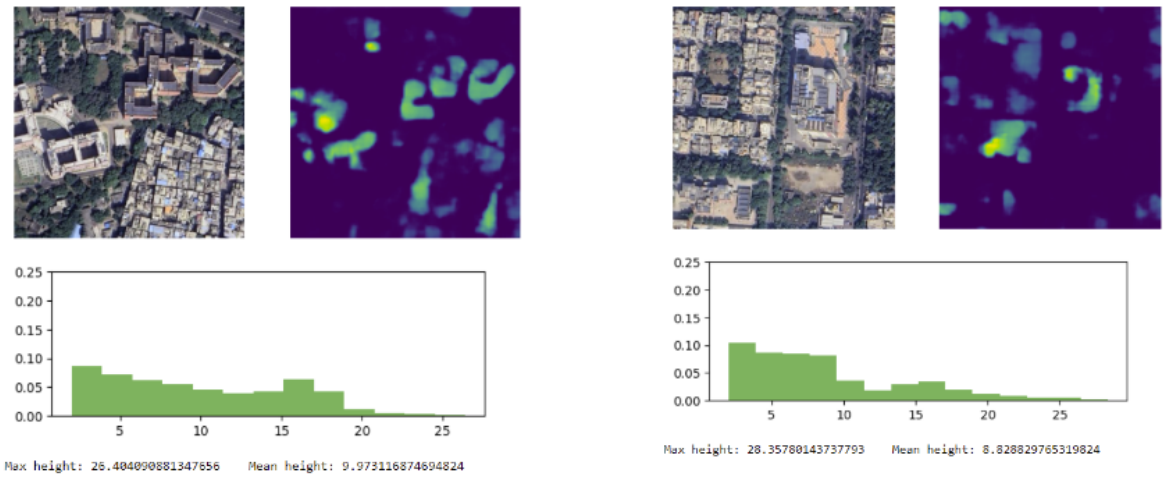

Figure 3: Performance on Delhi Dataset

5.2 Approach 2: Seam Carving [9]

To tackle the unstructured arrangement of buildings, seam carving was applied to the DFC23 dataset.
Seam carving is a content-aware image resizing technique that preserves important features while maintaining overall structure. This technique was used to remove unwanted spaces between buildings, mimicking the dense arrangements typical in developing regions.

5.2.1 Algorithm of Seam Carving

Step 1: Energy Function
The algorithm calculates an energy map of the image, which highlights areas of high importance. Energy is usually calculated based on gradients or edges, so high-energy areas are typically where the important features are.
Step 2: Seam Finding
The algorithm then finds the lowest-energy seam—a continuous path of pixels from top to bottom (or left to right) with the least total energy.
Step 3: Seam Removal
The identified seam is removed from the image, and the process is repeated as many times as needed to achieve the desired size.

5.2.2 Height Preservation

Step 1: Blurring The Ground Truth
- **Objective:** To preserve shadow details that may be lost due to the closely packed nature of buildings in developing regions.
- **Process:** Apply a Gaussian blur to the ground truth data. This blurring helps in retaining shadow information by smoothing out the fine details, which can be useful in maintaining the visual consistency of shadows during resizing.

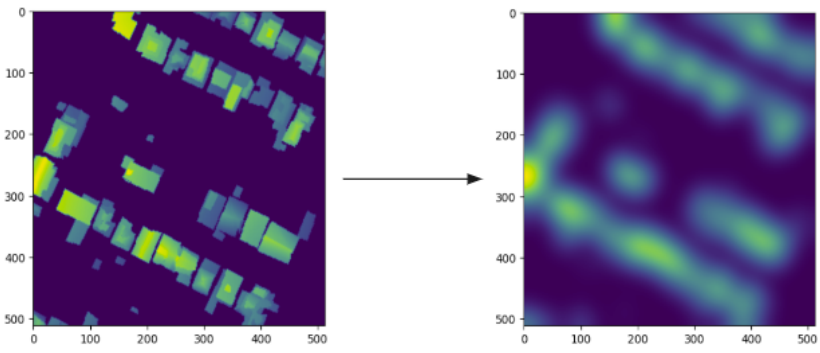

Figure 4: Blurred Ground Truth

Step 2: Creating an Importance Map
- **Objective:** To integrate the blurred data with the original ground truth in order to highlight areas of significance.
- **Process:** Combine the blurred ground truth image with the original ground truth to

generate an importance map. This map assigns higher importance to regions that are critical for maintaining the structural integrity and visual features of the buildings, including shadowed areas.

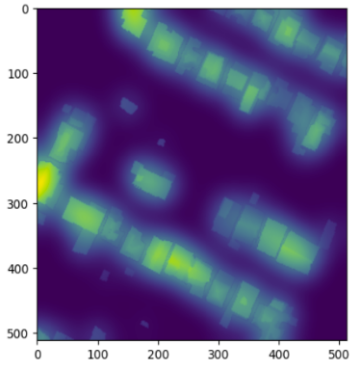

Figure 5: Importance Map

5.2.3 Results of Seam Carving on DFC23 Dataset

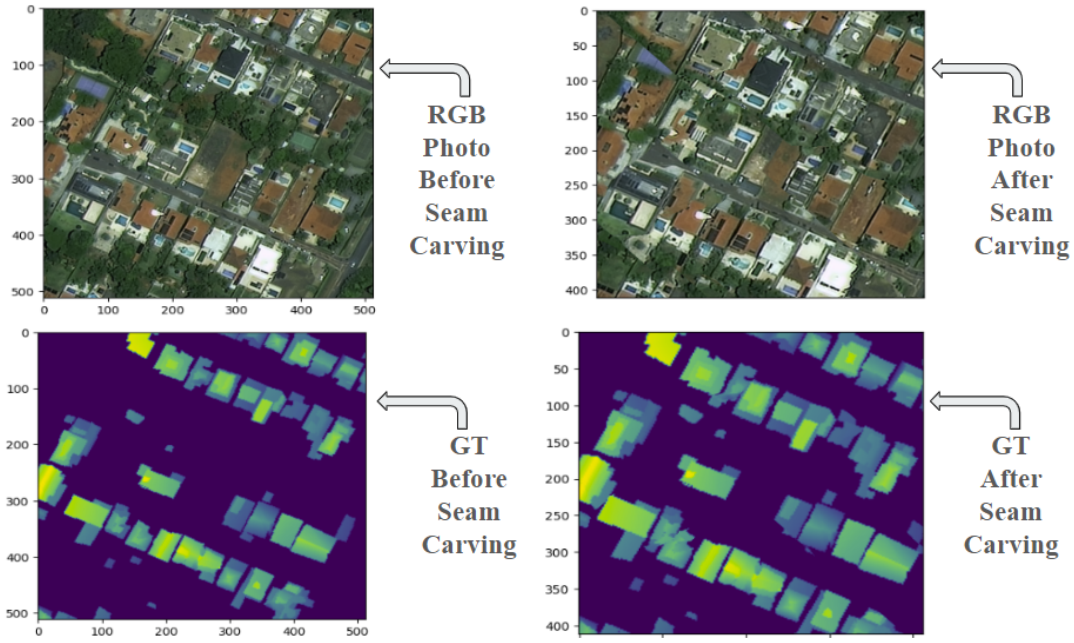

Figure 6: Results of Seam Carving on DFC23 Dataset

5.2.4 Results of Approach - 2

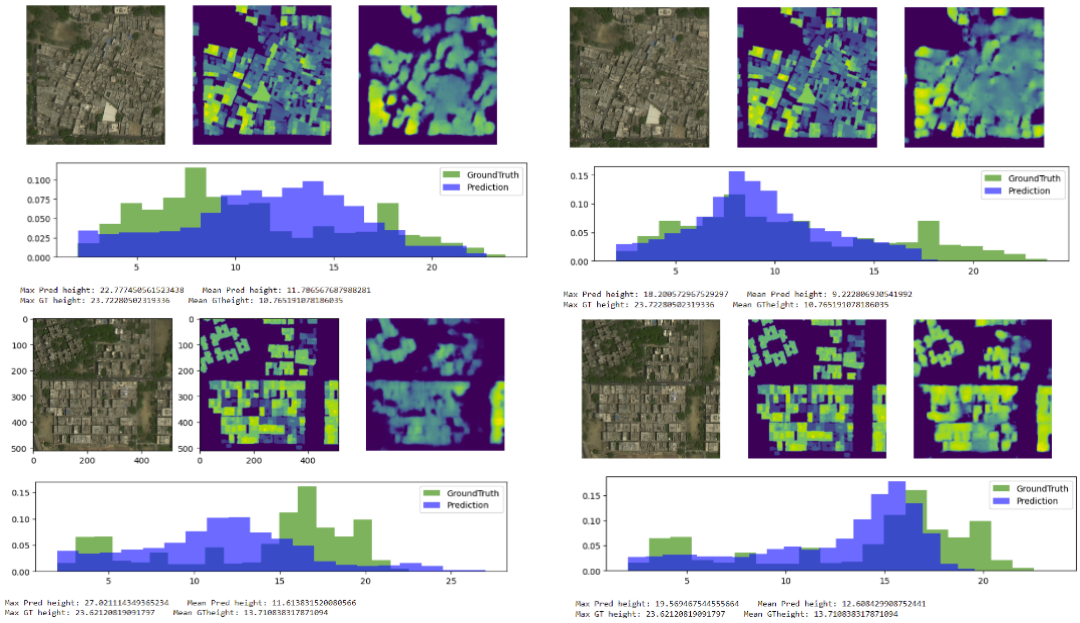

Figure 7: Performance on Delhi's images (DFC Dataset) compared to Approach - 1: Result of Approach - 1 on LHS and Result of Approach - 2 on RHS

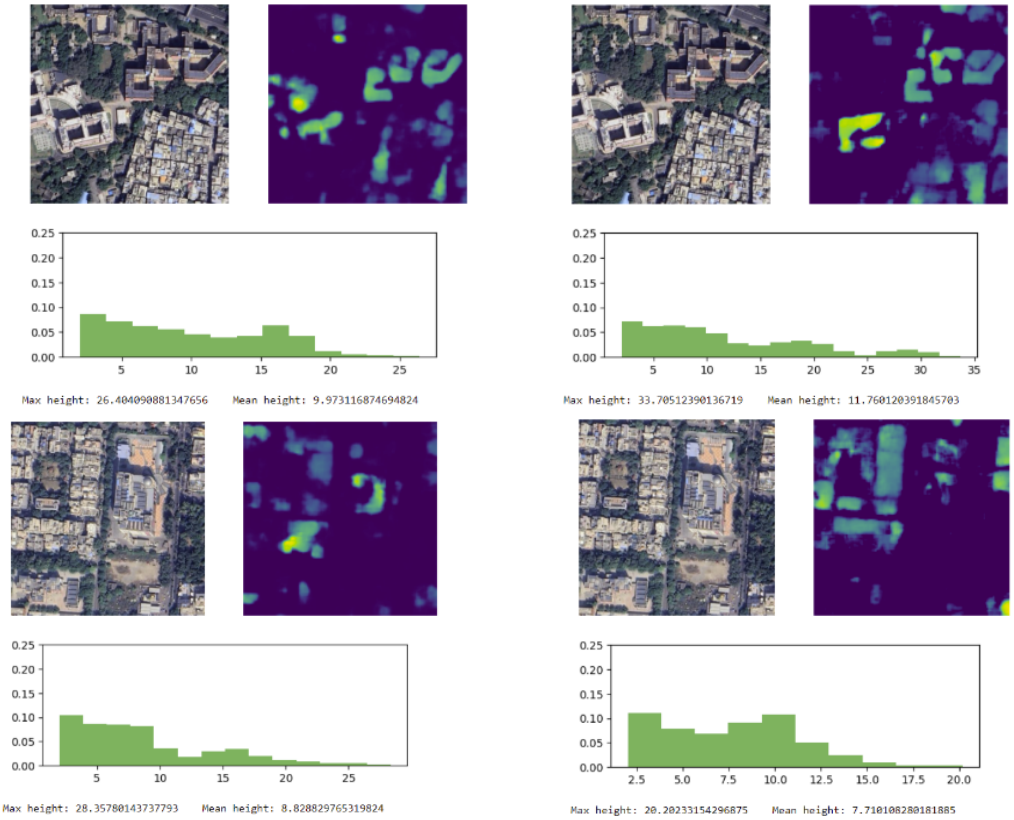

Figure 8: Performance on Delhi Dataset compared to Approach - 1: Result of Approach - 1 on LHS and Result of Approach - 2 on RHS

5.3 Approach 3: Footprint Inclusion

Building footprints from the Google Open Buildings Project [10] were integrated into the model to improve accuracy. These footprints provided high-quality building outlines, downloaded in patches that align with images from Google Earth Pro.

5.3.1 Details:

- Trained on Seam Carving Dataset along with Footprints from Google Open Building Project.
- Total 4 inputs: RGB + Footprints from Google Open Buildings Project.
- Tested on the Delhi dataset.
- No Domain Adaptation.
- L1 loss.

5.3.2 Results of Approach - 3

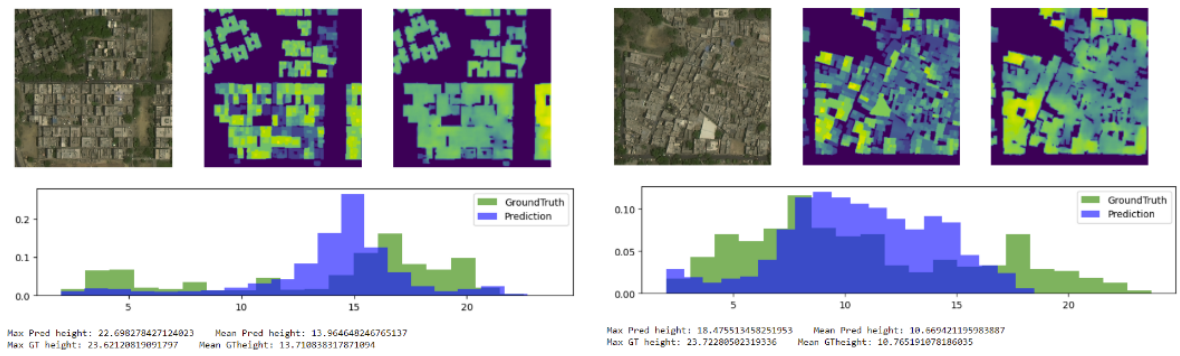

Figure 9: Performance on Delhi's images (DFC Dataset)

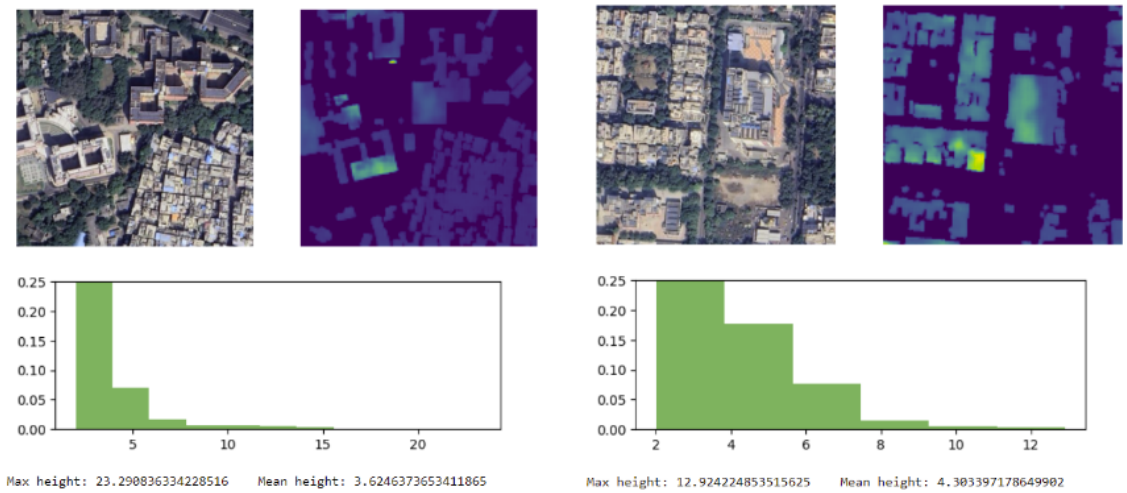

Figure 10: Performance on Delhi Dataset

5.4 Approach 4: Multi-task Learning[11] & Domain Adaptation[12]

A multi-task learning approach was employed, with one epoch for height detection and one for footprint detection.
We defined a deep learning model based on the UNet architecture, with some key modifications for height prediction tasks.

5.4.1 Architecture of the Model

The model comprises an encoder-decoder structure with segmentation and regression tasks, leveraging self-attention and cross-attention mechanisms. Below is a detailed configuration of the model:

5.4.2 Encoder

The encoder utilizes a modified U-Net architecture with atrous spatial pyramid pooling (ASPP) and depthwise separable convolutions. The filters for each layer are specified as {64, 128, 256, 512, 512}, and max pooling is used for downsampling. The input consists of 3 channels, and the ASPP block is configured with the following settings:

- **Kernel size:** 5
- **Squeeze-and-Excitation Operator:** False
- **Dilation layers:**
  - Stage 1: {1, 2, 4, 7}
  - Stage 2: {1, 2, 4, 7}
  - Stage 3: {1, 2, 4}
  - Stage 4: {1, 2, 4}
  - Stage 5: {1, 2}

5.4.3 Decoder

The decoder is split into two tracks for multi-task learning, focusing on segmentation and regression tasks.

Segmentation Track

- **Produced Outputs:** Refined shadow, building footprint
- **Self-Attention:** Enabled
- **Final Activation:** Steep sigmoid
- **Cross-Attention:** Not used

Regression Track

- **Produced Outputs:** Digital Surface Model (DSM)
- **Self-Attention:** Enabled

- **Final Activation:** ReLU
- **Cross-Attention:** Cross-attention is applied between the regression and segmentation tracks. The attention mechanism uses:
  - **Query:** Regression
  - **Key:** Segmentation
  - **Value:** Segmentation
  - **Window sizes:** {4, 4, 8, 16, 20}

5.4.4 Attention Mechanisms

Self-attention : is applied at each layer of the decoder, with layer normalization before attention. The cross-attention mechanism takes windows of the query, key, and value tensors, applies attention on these smaller patches, and then reassembles them.

Windowed Cross-Attention : The cross-attention mechanism divides the input tensors into windows of varying sizes (4, 4, 8, 16, 20). Attention is applied to each window separately, followed by reassembly into the full-sized feature map.

5.4.5 Architecture Blocks

Encoder Block:
Each encoder block consists of a depthwise separable convolution followed by atrous convolutions in the ASSP block. The general structure is:

- Basic convolution
- Depthwise convolution
- Batch normalization
- ReLU activation
- Dropout (p = 0.1)
- Atrous Spatial Pyramid Pooling (ASPP) with dilation rates as per configuration
- Batch normalization and ReLU
- Dropout (p = 0.1)

Decoder Block:
The decoder block is composed of a series of convolutions, batch normalization, and ReLU activations followed by transposed convolutions for upsampling. Each block includes:

- Basic convolution
- Depthwise convolution
- Batch normalization
- ReLU activation

- Dropout (p = 0.1)
- Transposed convolution for upsampling
- Batch normalization and ReLU activation

5.4.6 Pixel Shuffler for Upsampling

For the final upsampling stage, a pixel shuffler is used to produce high-resolution outputs. This block includes:

- Depthwise convolution
- Pixel shuffle with a scaling factor of 2
- Final convolution layer to map the channels to the desired output size

Component	Layer/Operation	Details
Encoder	Input Channels	3 (RGB input)
	Block 1	UNet Filter: 64, Dilation: [1, 2, 4, 7], MaxPool, SEOperator: False, ASSP Kernel: 5
	Block 2	UNet Filter: 128, Dilation: [1, 2, 4, 7], MaxPool, SEOperator: False, ASSP Kernel: 5
	Block 3	UNet Filter: 256, Dilation: [1, 2, 4], MaxPool, SEOperator: False, ASSP Kernel: 5
	Block 4	UNet Filter: 512, Dilation: [1, 2, 4], MaxPool, SEOperator: False, ASSP Kernel: 5
	Block 5	UNet Filter: 512, Dilation: [1, 2], MaxPool, SEOperator: False, ASSP Kernel: 5
Decoder		
Segmentation Track	Self-Attention (All Layers)	Enabled: Multi-head attention with 4 heads, Layer Norm applied
	Block 1	UNet Decoder: 512 -> 512, Transposed Conv, Self-Attention, BatchNorm, Dropout
	Block 2	UNet Decoder: 512 -> 256, Transposed Conv, Self-Attention, BatchNorm, Dropout
	Block 3	UNet Decoder: 256 -> 128, Transposed Conv, Self-Attention, BatchNorm, Dropout
	Block 4	UNet Decoder: 128 -> 64, Transposed Conv, Self-Attention, BatchNorm, Dropout
	Final Layer	Output: ["refined_shadow", "footprint"], Activation: Steep Sigmoid
Regression Track	Self-Attention (All Layers)	Enabled: Multi-head attention with 4 heads, Layer Norm applied
	Cross-Attention	From Segmentation to Regression: Query from Regression, Key and Value from Segmentation, Window Sizes: [4, 4, 8, 16, 20]
	Block 1	UNet Decoder: 512 -> 512, Transposed Conv, Cross-Attention, BatchNorm, Dropout
	Block 2	UNet Decoder: 512 -> 256, Transposed Conv, Cross-Attention, BatchNorm, Dropout
	Block 3	UNet Decoder: 256 -> 128, Transposed Conv, Cross-Attention, BatchNorm, Dropout
	Block 4	UNet Decoder: 128 -> 64, Transposed Conv, Cross-Attention, BatchNorm, Dropout
	Final Layer	Output: ["dsm"], Activation: ReLU
Final Processing	Pre-Pixel Shuffler	Depthwise Conv (1024 -> 512), Basic Conv (512), BatchNorm, ReLU
	Pixel Shuffler	Pixel Shuffler for upscaling
	Post-Pixel Shuffler	Depthwise Conv (512 + 512 -> 512), Final Output: ["refined_shadow", "footprint", "dsm"]

Figure 11: Architecture Table

5.4.7 Results of Approach - 4

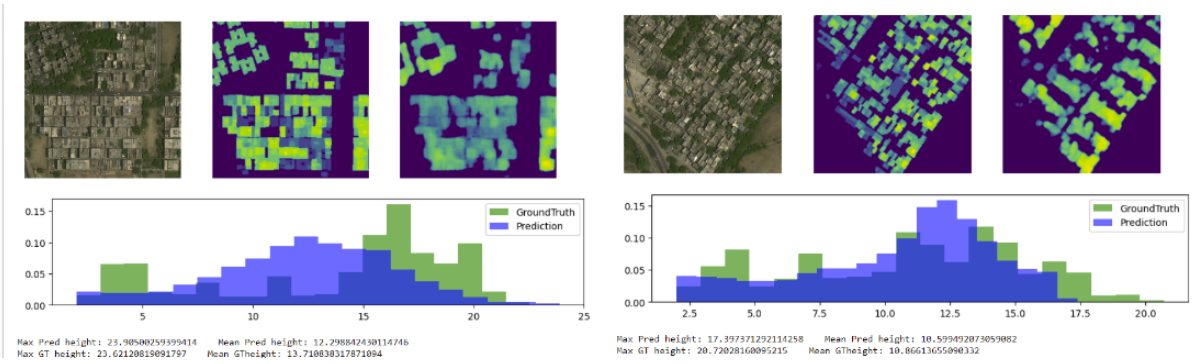

Figure 12: Performance on Delhi's images (DFC Dataset)

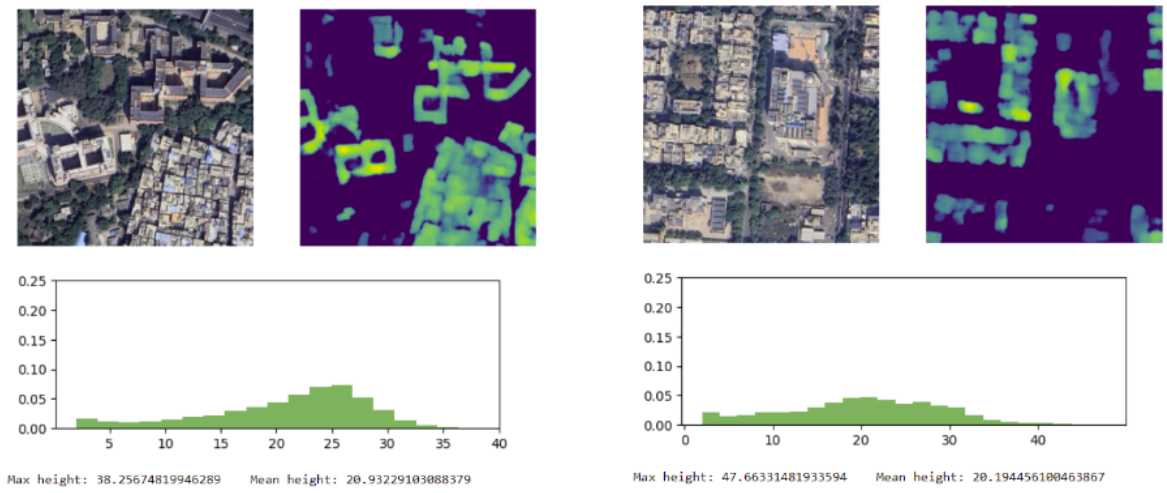

Figure 13: Performance on Delhi Dataset

6 Conclusion

The model trained with seam carving and footprint inclusion showed improved visual results. However, due to domain shifts, the accuracy of height prediction was not as high as anticipated. The multi-task learning approach with domain adaptation demonstrated promise in addressing these issues but further refinement is needed.

References

[1] G. Liu, B. Peng, T. Liu, P. Zhang, M. Yuan, C. Lu, N. Cao, S. Zhang, S. Huang, T. Wang, *et al.*, "Large-scale fine-grained building classification and height estimation for semantic urban reconstruction: Outcome of the 2023 ieee grss data fusion contest," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2024.

[2] S. A. Jamal and A. Aribisala, "Data fusion for multi-task learning of building extraction and height estimation," *arXiv preprint arXiv:2308.02960*, 2023.

[3] C.-J. Liu, V. A. Krylov, P. Kane, G. Kavanagh, and R. Dahyot, "Im2elevation: Building height estimation from single-view aerial imagery," *remote sensing*, vol. 12, no. 17, p. 2719, 2020.

[4] C. Persello, R. Hänsch, G. Vivone, K. Chen, Z. Yan, D. Tang, H. Huang, M. Schmitt, and X. Sun, "2023 ieee grss data fusion contest: Large-scale fine-grained building classification for semantic urban reconstruction [technical committees]," *IEEE Geoscience and Remote Sensing Magazine*, vol. 11, no. 1, pp. 94–97, 2023.

[5] Google Inc., "Google earth pro." `https://www.google.com/earth/`, 2024. Version 7.3.4.

[6] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pp. 234–241, Springer, 2015.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[8] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

[9] S. Avidan and A. Shamir, "Seam carving for content-aware image resizing," in *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pp. 609–617, 2023.

[10] W. Sirko, S. Kashubin, M. Ritter, A. Annkah, Y. S. E. Bouchareb, Y. Dauphin, D. Keysers, M. Neumann, M. Cisse, and J. Quinn, "Continental-scale building detection from high resolution satellite imagery," *arXiv preprint arXiv:2107.12283*, 2021.

[11] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, pp. 41–75, 1997.

[12] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.