Bottleneck-Gated Segmentation Network: Efficient Conditional Decoder Activation for Semantic Segmentation

Bhoumik Patidar
Computer Science And Engineering
Indian Institute Of Technology, Gandhinagar
bhoumik.patidar@iitgn.ac.in

Abstract—Efficient semantic segmentation of satellite imagery remains a significant challenge due to the computational demands of processing large-scale data where regions of interest (ROIs) are often sparsely distributed. This paper introduces the Bottleneck-Gated Segmentation Network (BGS-Net), a novel approach that selectively activates the decoder component of a U-Net architecture only for image patches likely to contain target objects. By mathematically analyzing bottleneck feature representations, I develop a two-stage gating mechanism that achieves a 56.62% reduction in decoder computation while maintaining competitive segmentation quality (Building IoU of 0.6981 compared to 0.7145 for the standard U-Net). My approach requires no re-training of the base model and introduces negligible inference overhead, making it particularly suitable for large-scale satellite image processing where computational resources are constrained. Experimental results on the LandCoverAI dataset demonstrate that BGS-Netachieves an optimal balance between computational efficiency and segmentation accuracy for sparse ROI scenarios.

Index Terms—semantic segmentation, computational efficiency, satellite imagery, conditional computation, feature analysis, U-Net, bottleneck features

1 Introduction

Semantic segmentation of satellite imagery plays a crucial role in various applications including urban planning, environmental monitoring, and disaster response. Despite significant advances in deep learning architectures for this task, the computational complexity of processing large-scale satellite data remains a major challenge, especially in resource-constrained environments or time-sensitive applications.

A key observation in satellite imagery is that regions of interest (ROIs) such as buildings, roads, or water bodies often occupy only a small fraction of the total image area. For instance, in typical rural or suburban scenes, buildings may cover less than 15% of the image. Despite this sparsity, conventional segmentation approaches process the entire image with the same computational intensity, leading to significant inefficiency.

Encoder-decoder architectures like U-Net [1] have become the standard approach for semantic segmentation tasks. In these architectures, the encoder progressively downsamples the input to extract high-level features, while the decoder upsamples these features to produce the final segmentation map. Notably, the decoder component typically accounts for 65-70% of the total computation due to its upsampling operations and skip connections [17].

In this paper, I introduce the Bottleneck-Gated Segmentation Network (BGS-Net), which leverages the bottleneck representations between the encoder and decoder to selectively activate the decoder only for patches likely to contain target objects. My key insight is that the bottleneck features contain sufficient information to reliably determine the presence or absence of objects of interest before committing to the computationally expensive decoder operations.

The main contributions of this paper are:

- A mathematical framework for analyzing bottleneck feature representations to identify discriminative channels for target object detection
- A two-stage gating mechanism that achieves high recall for regions containing targets while filtering out false alarms from non-target regions
- An efficient implementation that reduces decoder computations by 56.62% while maintaining competitive segmentation quality (Building IoU of 0.6981 vs. 0.7145)
- A methodology that requires no re-training of the base model and introduces negligible inference overhead (approximately 0.0007% of a single decoder convolution)

My experimental results demonstrate that BGS-Netprovides an effective solution for efficient semantic segmentation of satellite imagery, particularly when target objects are sparsely distributed throughout the scenes.

2 Related Work

2.1 Semantic Segmentation Architectures

Fully Convolutional Networks (FCNs) [2] established the foundation for modern semantic segmentation by replacing fully connected layers with convolutional layers. U-Net [1] extended this approach with skip connections between encoder and decoder paths, which has proven particularly effective for satellite and medical imaging where fine details are crucial. Various architectural improvements have been proposed, including DeepLab [3] with atrous convolutions, PSPNet [4] with pyramid pooling, and HRNet [5] with high-resolution feature maintenance.

For satellite imagery specifically, architectures such as DeepGlobe [6] and SpaceNet [7] have adapted these fundamental designs to address the unique challenges of remote sensing data, including large spatial coverage, variable resolutions, and diverse object scales.

2.2 Computation Optimization in Deep Learning

Several approaches have been proposed to optimize the computational efficiency of deep neural networks:

Model Compression: Techniques like pruning [8], quantization [9], and knowledge distillation [10] reduce model size and computation. While effective, these approaches often trade off accuracy for efficiency and typically require retraining or fine-tuning.

Early-Exit Mechanisms: Multi-exit architectures [11], [12] allow inference to terminate early for simple inputs. These approaches require architectural modifications and additional training parameters.

Conditional Computation: Methods that selectively activate parts of the network based on input characteristics [13], [14] have shown promise. However, most implementations focus on classification rather than dense prediction tasks like segmentation.

Region-Based Processing: Approaches that identify regions of interest before detailed processing [15], [16] are conceptually similar to my approach but typically involve multi-stage pipelines with separate detection and segmentation networks.

My approach differs from these existing methods by leveraging the bottleneck representation of a pre-trained segmentation network to selectively execute the decoder component without requiring architectural changes, additional parameters, or retraining of the base model.

3 METHODOLOGY

3.1 Dataset and Baseline Model

I conduct my experiments on the LandCoverAI dataset [18], which contains high-resolution satellite imagery with pixel-level annotations for five classes: background, building, woodland, water, and road. The dataset comprises satellite images at 25cm/pixel resolution, providing detailed ground features appropriate for fine-grained segmentation tasks.

For my baseline,Iimplement a standard U-Net architecture with an encoder comprising four downsampling blocks, a bottleneck layer, and a decoder with four upsampling blocks. Each encoder block consists of two 3×3 convolutional layers followed by batch normalization and ReLU activation, with max-pooling for downsampling. The decoder follows a similar structure with transposed convolutions for upsampling, with skip connections from the encoder. The bottleneck layer contains 1024 channels at a spatial resolution of 32×32 for my input size of 512×512 pixels.

The network was trained using a Jaccard loss function (IoU loss) with an Adam optimizer at a learning rate of 5e-5 for 60 epochs with early stopping. The final model achieves the following per-class IoU scores on the test set:

TABLE 1: Baseline U-Net Performance

Class	IoU
Background	0.8911
Building	0.7145
Woodland	0.8660
Water	0.8362
Road	0.5624

3.2 Bottleneck-Gated Segmentation Network (BGS-Net)

The core innovation of BGS-Netis a gating mechanism that determines whether to execute the decoder based on an analysis of the bottleneck features. The approach can be formally described as follows:

Let $\mathcal{E}$, $\mathcal{B}$, and $\mathcal{D}$ represent the encoder, bottleneck, and decoder components of a U-Net architecture, respectively. For an input image $\mathbf{X} \in \mathbb{R}^{H\times W\times C}$, the standard U-Net computation follows:

$$\mathbf{F} = \mathcal{E}(\mathbf{X}) \tag{1}$$

$$\mathbf{B} = \mathcal{B}(\mathbf{F}) \tag{2}$$

$$\mathbf{Y} = \mathcal{D}(\mathbf{B}, \mathbf{F}) \tag{3}$$

where $\mathbf{F}$ represents the feature maps from the encoder (including skip connections), $\mathbf{B}$ represents the bottleneck features, and $\mathbf{Y}$ is the final segmentation output.

In BGS-Net, I introduce a gating function $G(\mathbf{B})$ that decides whether to execute the decoder:

$$\mathbf{Y} = \begin{cases} \mathcal{D}(\mathbf{B}, \mathbf{F}) & \text{if } G(\mathbf{B}) = 1 \\ \mathbf{Y}_{\text{default}} & \text{if } G(\mathbf{B}) = 0 \end{cases} \tag{4}$$

where $\mathbf{Y}_{\text{default}}$ is a default output (e.g., all-background segmentation) for patches deemed unlikely to contain targets, and $G(\mathbf{B})$ is a binary function that evaluates to 1 if the bottleneck features indicate the presence of target objects.

3.3 Mathematical Analysis of Bottleneck Representations

At the heart of BGS-Netis a mathematical framework for analyzing bottleneck features to identify discriminative patterns that indicate the presence of target objects. For a bottleneck tensor $\mathbf{B} \in \mathbb{R}^{C\times H'\times W'}$ with C channels and spatial dimensions $H' \times W'$, I define the channel-wise mean activation $\mu_c(\mathbf{B})$ for channel c as:

$$\mu_c(\mathbf{B}) = \frac{1}{H' \times W'} \sum_{i=1}^{H'} \sum_{j=1}^{W'} \mathbf{B}_{c,i,j} \tag{5}$$

To identify channels that are discriminative for a target class (e.g., buildings), I analyze the statistical distributions of these channel-wise activations across different classes of patches.Idefine the following sets:

$$\mathcal{P}_{\text{target}} = \{\mathbf{B} \mid \text{patch contains target object}\} \tag{6}$$

$$\mathcal{P}_{\text{non-target}} = \{\mathbf{B} \mid \text{patch does not contain target object}\} \tag{7}$$

For each channel c,Icompute the mean activation across all patches in each set:

$$\mu_c^{\text{target}} = \frac{1}{|\mathcal{P}_{\text{target}}|} \sum_{\mathbf{B} \in \mathcal{P}_{\text{target}}} \mu_c(\mathbf{B}) \tag{8}$$

$$\mu_c^{\text{non-target}} = \frac{1}{|\mathcal{P}_{\text{non-target}}|} \sum_{\mathbf{B} \in \mathcal{P}_{\text{non-target}}} \mu_c(\mathbf{B}) \tag{9}$$

The discriminative power of channel c can then be quantified by the absolute difference between these means:

$$\delta_c = |\mu_c^{\text{target}} - \mu_c^{\text{non-target}}| \tag{10}$$

Channels with higher δ_c values are more discriminative for distinguishing between target and non-target patches. However, the mean difference alone is insufficient for determining optimal decision thresholds.Ialso analyze the distributions $P(\mu_c(\mathbf{B}) \mid \mathbf{B} \in \mathcal{P}_{\text{target}})$ and $P(\mu_c(\mathbf{B}) \mid \mathbf{B} \in \mathcal{P}_{\text{non-target}})$ to understand the overlap and separation between classes.

I further refine my analysis by examining specific non-target subclasses that commonly trigger false alarms. For instance,Idefine:

$$\mathcal{P}_{\text{false-alarm}} = \{\mathbf{B} \mid \text{patch triggers false alarm, contains specific non-target class}\} \tag{11}$$

This allows us to identify channels that are particularly effective at distinguishing target objects from specific confounding classes.

3.4 Threshold Determination Framework

Based on my bottleneck feature analysis,Idevelop a two-stage gating mechanism that combines high recall for target objects with effective filtering of false alarms.

3.4.1 Stage 1: Target Detection

For the first stage,Iselect K_1 channels with the highest discriminative power for identifying target objects and determine optimal thresholds for each channel.

For a channel c where target objects typically show higher activations than non-targets,Idefine a decision function $g_c(\mathbf{B}, \theta_c)$:

$$g_c(\mathbf{B}, \theta_c) = \begin{cases} 1 & \text{if } \mu_c(\mathbf{B}) > \theta_c \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

Conversely, for a channel where target objects show lower activations:

$$g_c(\mathbf{B}, \theta_c) = \begin{cases} 1 & \text{if } \mu_c(\mathbf{B}) < \theta_c \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

The first-stage gate $G_1(\mathbf{B})$ combines these channel decisions using a logical OR to maximize recall:

$$G_1(\mathbf{B}) = \bigvee_{i=1}^{K_1} g_{c_i}(\mathbf{B}, \theta_{c_i}) \tag{14}$$

The thresholds θ_c are determined through ROC curve analysis to maximize the difference between true positive rate (TPR) and false positive rate (FPR):

$$\theta_c^* = \arg\max_{\theta}(TPR(\theta) - FPR(\theta)) \tag{15}$$

where:

$$TPR(\theta) = \frac{|\{\mathbf{B} \in \mathcal{P}_{\text{target}} \mid g_c(\mathbf{B}, \theta) = 1\}|}{|\mathcal{P}_{\text{target}}|} \tag{16}$$

$$FPR(\theta) = \frac{|\{\mathbf{B} \in \mathcal{P}_{\text{non-target}} \mid g_c(\mathbf{B}, \theta) = 1\}|}{|\mathcal{P}_{\text{non-target}}|} \tag{17}$$

3.4.2 Stage 2: False Alarm Filtering

For the second stage,Iidentify K_2 channels that are particularly effective at distinguishing target objects from common false alarm classes. Based on my analysis,Iobserved that certain non-target classes (e.g., dense woodland) consistently trigger false alarms in the first stage.

For these channels,Idefine a false alarm detection function $f_c(\mathbf{B}, \gamma_c)$:

$$f_c(\mathbf{B}, \gamma_c) = \begin{cases} 1 & \text{if } \mu_c(\mathbf{B}) > \gamma_c \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

The second-stage filter $G_2(\mathbf{B})$ combines these channel decisions using a logical AND to ensure only clear false alarms are filtered:

$$G_2(\mathbf{B}) = \bigwedge_{i=1}^{K_2} f_{c_i}(\mathbf{B}, \gamma_{c_i}) \tag{19}$$

3.4.3 Combined Gating Mechanism

The final gating function $G(\mathbf{B})$ combines both stages:

$$G(\mathbf{B}) = G_1(\mathbf{B}) \land \neg G_2(\mathbf{B}) \tag{20}$$

This ensures that the decoder is only executed for patches that both pass the initial target detection stage and are not identified as false alarms in the second stage.

The complete algorithm for the two-stage gating mechanism is presented in Algorithm 1.

4 Experimental Setup

4.1 Implementation Details

I implemented BGS-Netbased on a standard U-Net architecture with a 1024-channel bottleneck layer. The baseline model was trained on the LandCoverAI dataset, which was split into training (60%), validation (20%), and test (20%) sets.Itrained the model using the Adam optimizer with a learning rate of 5e-5 and a batch size of 8 for 30 epochs with early stopping based on validation loss.

Algorithm 1 Two-Stage Gating Mechanism

```
1:  procedure GATEFUNCTION($\mathbf{B}$)
2:      contains_target ← False
                                ▷ Stage 1: Target Detection (High Recall)
3:      for each channel $c$ and threshold $\theta_c$ in target detection set do
4:          $\mu_c \leftarrow$ MeanActivation($\mathbf{B}, c$)
5:          if $(c$.target_higher$\wedge \mu_c > \theta_c) \vee (\neg c$.target_higher$\wedge \mu_c < \theta_c)$ then
6:              contains_target ← True
7:              break
8:          end if
9:      end for
10:     if ¬contains_target then
11:         return False
12:     end if
                                ▷ Stage 2: False Alarm Filtering
13:     is_false_alarm ← True
14:     for each channel $c$ and threshold $\gamma_c$ in false alarm set do
15:         $\mu_c \leftarrow$ MeanActivation($\mathbf{B}, c$)
16:         if $\mu_c \leq \gamma_c$ then
17:             is_false_alarm ← False
18:             break
19:         end if
20:     end for
21:     return contains_target ∧ ¬is_false_alarm
22: end procedure
```

For the bottleneck feature analysis,Iprocessed the entire training set to extract bottleneck representations and categorized patches based on their ground-truth masks.Idefined "building patches" as those containing at least 0.5% building pixels.Ianalyzed channel activations to identify the most discriminative channels for building detection and false alarm filtering.

For the first stage (target detection),Iselected 10 channels (937, 117, 640, 227, 996, 676, 135, 988, 534, 774) with optimal thresholds determined through ROC analysis. For the second stage (false alarm filtering),Iidentified 3 channels (135, 700, 996) that strongly activate for certain non-building classes, with a threshold of 1.0 for robust filtering.

4.2 Evaluation Metrics

I evaluated BGS-Neton the following metrics:

- **Segmentation Quality**: Per-class IoU, with a focus on the Building class
- **Decoder Execution Rate**: Percentage of patches that trigger decoder execution
- **Building Detection Recall**: Percentage of building-containing patches correctly identified
- **Building Detection Precision**: Percentage of decoder-executed patches that actually contain buildings

I also performed detailed analysis of cases where the gating mechanism missed buildings or triggered unnecessary decoder executions.

5 Results and Analysis

5.1 Segmentation Performance

BGS-Netachieves a Building IoU of 0.6981 compared to 0.7145 for the baseline U-Net, representing a modest 2.3% relative decrease in segmentation quality. This demonstrates that conditional decoder execution maintains competitive segmentation performance while substantially reducing computation.

Table 2 compares the segmentation performance of the baseline U-Net and BGS-Neton the Building class, which is my primary target.

TABLE 2: Building Segmentation Performance

Model	Building IoU
Vanilla U-Net	0.7145
BGS-Net	0.6981

Importantly, the IoU is calculated only for patches where buildings are present, ensuring thatImeasure segmentation quality on relevant regions. The slight decrease in IoU is primarily attributable to the small percentage of building patches (5.65%) that are missed by the gating mechanism and receive default (all-background) segmentation.

5.2 Computational Efficiency

The computational benefits of BGS-Netare substantial, with a 56.62% reduction in decoder computations. Table 3 summarizes the key efficiency metrics.

TABLE 3: Computational Efficiency Results

Metric	Value
Total test patches	1,602
Patches containing buildings	230 (14.36%)
Patches with decoder execution	695 (43.38%)
Decoder computation reduction	56.62%

The overhead introduced by the gating mechanism is negligible, requiring approximately 10,251 operations per patch, which represents about 0.0007% of a single decoder convolution (which typically requires a few billion operations).

5.3 Detection Performance Analysis

Analysis of the missed building cases revealed that these patches typically contain very small buildings (average building coverage of 1.45%), which explains why they might be difficult to detect from bottleneck features.

6 Comparative Advantages of BGS-Net

BGS-Netoffers several key advantages over alternative approaches to efficient semantic segmentation:

6.1 Architectural Efficiency

Separate Encode vs. Decode Optimization: BGS-Netuniquely targets the decoder component, which typically accounts for 65-70% of the computation in U-Net architectures. By always executing the lightweight encoder and bottleneck stages, my approach ensures that critical feature extraction occurs for every patch while selectively applying the more costly upsampling and skip connection operations.

Negligible Inference Overhead: The gating mechanism adds minimal computational overhead (approximately 0.0007% of a single decoder convolution), making it highly efficient even for real-time applications.

6.2 Training Efficiency

No Full-Model Retraining: Unlike knowledge distillation or model compression techniques, BGS-Netrequires no retraining of the base segmentation model. The threshold determination process takes only minutes rather than the hours or days required for full model training.

Rapid Adaptation to New Target Classes: The bottleneck analysis can be quickly reapplied to optimize for different target classes within the existing segmentation model, making BGS-Nethighly adaptable to changing application requirements.

6.3 Comparison with Alternative Approaches

Preservation of Segmentation Quality: Unlike knowledge distillation approaches that typically sacrifice some accuracy for efficiency, BGS-Netmaintains near-identical segmentation quality for regions containing target objects, with only a 2.3% relative decrease in Building IoU.

Advantages over Trainable Early-Exit Mechanisms: Compared to multi-exit segmentation networks that require additional parameters and training, BGS-Netintroduces no new parameters and works with pre-trained models without modification.

Complementarity with Existing Methods: BGS-Netcan be combined with other optimization techniques such as pruning, quantization, or architectural improvements, as it operates at the execution strategy level rather than modifying the model itself.

7 Discussion

7.1 Strengths and Limitations

The primary strength of BGS-Netis its ability to significantly reduce computation for semantic segmentation while maintaining high-quality results for regions of interest. This makes it particularly valuable for applications processing large-scale satellite imagery where computational resources are limited or where processing time is a concern.

However, the approach has several limitations that should be considered:

- **Target Sparsity Requirement:** The efficiency gains are most significant when target objects occupy a small fraction of the total image area. For scenes with dense target distribution, the computational benefits would be reduced.
- **Dependency on Bottleneck Representation:** The effectiveness of the gating mechanism relies on the discriminative power of the bottleneck features. If the base model has a poorly structured bottleneck representation, the gating performance may be suboptimal.

7.2 Generalizability

While my experiments focused on building segmentation in satellite imagery, the approach is generalizable to other segmentation tasks where targets are sparsely distributed. Potential applications include:

- Medical image segmentation, where pathological regions often occupy small portions of scans
- Agricultural monitoring, where crop disease or specific plant types may be sparsely distributed
- Autonomous driving perception, where certain object classes (e.g., pedestrians) may appear infrequently

The mathematical framework for bottleneck analysis could be applied to any encoder-decoder architecture with a well-defined bottleneck, not limited to U-Net variants.

8 Conclusion and Future Work

In this paper,Iintroduced the Bottleneck-Gated Segmentation Network (BGS-Net), an efficient approach to semantic segmentation that selectively activates the decoder component based on bottleneck feature analysis. My approach achieves a 56.62% reduction in decoder computation while maintaining competitive segmentation quality, with a Building IoU of 0.6981 compared to 0.7145 for the standard U-Net.

The key innovation of BGS-Netis a mathematically-grounded two-stage gating mechanism that combines high recall for target detection with effective filtering of false alarms. This approach requires no retraining of the base model and introduces negligible inference overhead, making it particularly suitable for large-scale satellite image processing.

Future work could explore several promising directions:

- **Multi-Class Gating:** Extending the approach to simultaneously optimize for multiple target classes
- **Adaptive Thresholding:** Developing dynamic thresholding strategies that adapt to different scenes or conditions
- **Integration with Other Optimizations:** Combining BGS-Netwith model compression techniques for further efficiency gains
- **Application to Video Segmentation:** Leveraging temporal continuity in video data to enhance gating decisions

BGS-Netdemonstrates that significant computational efficiency can be achieved without substantial sacrifices in segmentation quality by intelligently leveraging the information already present in the network's bottleneck representation. This approach represents a step toward making deep learning-based semantic segmentation more practical for resource-constrained applications and large-scale processing scenarios.

Acknowledgments

Thankful to the creators of LandCoverAI dataset for high resolution satellite imagery.

References

[1] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015, pp. 234-241.

[2] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431-3440.

[3] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834-848, 2017.

[4] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, “Pyramid scene parsing network,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2881-2890.

[5] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao, “Deep high-resolution representation learning for visual recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 10, pp. 3349-3364, 2020.

[6] I. Demir, K. Koperski, D. Lindenbaum, G. Pang, J. Huang, S. Basu, F. Hughes, D. Tuia, and R. Raskar, “DeepGlobe 2018: A challenge to parse the earth through satellite images,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 172-181.

[7] A. Van Etten, D. Lindenbaum, and T. M. Bacastow, “SpaceNet: A remote sensing dataset and challenge series,” *arXiv preprint arXiv:1807.01232*, 2018.

[8] S. Han, J. Pool, J. Tran, and W. Dally, “Learning both weights and connections for efficient neural network,” in *Advances in Neural Information Processing Systems*, 2015, pp. 1135-1143.

[9] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, “Quantization and training of neural networks for efficient integer-arithmetic-only inference,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2704-2713.

[10] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” *arXiv preprint arXiv:1503.02531*, 2015.

[11] S. Teerapittayanon, B. McDanel, and H. T. Kung, “BranchyNet: Fast inference via early exiting from deep neural networks,” in *2016 23rd International Conference on Pattern Recognition (ICPR)*, 2016, pp. 2464-2469.

[12] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Q. Weinberger, “Multi-scale dense networks for resource efficient image classification,” in *International Conference on Learning Representations*, 2017.

[13] Y. Bengio, N. Léonard, and A. Courville, “Estimating or propagating gradients through stochastic neurons for conditional computation,” *arXiv preprint arXiv:1308.3432*, 2013.

[14] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, “Outrageously large neural networks: The sparsely-gated mixture-of-experts layer,” in *International Conference on Learning Representations*, 2017.

[15] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in *Advances in Neural Information Processing Systems*, 2015, pp. 91-99.

[16] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2961-2969.

[17] A. Howard, M. Sandler, G. Chu, L. C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, “Searching for MobileNetV3,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1314-1324.

[18] A. Boguszewski, D. Batorski, N. Ziemba-Jankowska, T. Dziedzic, and A. Zambrzycka, “LandCoverAI: Dataset for Automatic Mapping of Buildings, Woodlands, Water and Roads from Aerial Imagery,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 1102-1110.