Cube Kernel: A Novel Approach to Enable Local Gradient Flow Across Channels in CNNs

Zhimeng He[1*], Yuwei Cai[1], Meiliu Wu[1], Xinyan Xian[1], Brian Barrett[1]

[1] School of Geographical and Earth Sciences, University of Glasgow, Glasgow, United Kingdom

Keywords: Cube Kernel; cross-channel feature fusion; gradient propagation; semantic segmentation; building rooftop extraction.

Abstract

Understanding inter-band and cross-channel relationships is essential for human color perception and object recognition. Yet, local gradients in standard convolutions are tied to fixed input–output channel pairs, and thus channels are fused by a dense, fully-coupled weight tensor: each output channel aggregates all input channels in a uniform way at every spatial location. This leads to heavy computation and does not exploit structured sparsity or selective local channel mixing. To overcome this limitation, we introduce Cube Kernel, a novel convolutional operator that introduces structured cross-channel groups into the local gradient. This design strengthens cross-channel feature fusion, improves optimization efficiency, and reduces computational overhead. Extensive building extraction experiments validate its effectiveness: Cube Kernel consistently outperforms standard convolutions and Involution when integrated into UNet, and replacing a single layer in DeepLabV3+, Swin-UNet, or UNet leads to consistent performance gains. Beyond serving as a lightweight plug-in module, Cube Kernel also scales effectively as a fundamental building block. A Cube-enhanced ConvNeXt variant, ConvNeXt-Cube, achieves state-of-the-art performance across all models (0.9095 IoU / 0.9535 F1 on WBD and 0.9133 IoU / 0.9547 F1 on WHU), demonstrating strong stackability and architectural potential. These results highlight a largely overlooked space in CNN design: enhancing cross-channel interaction at the gradient level. Cube Kernel offers a scalable and efficient alternative to deepen networks for channel mixing, laying a foundation for future advancements in convolutional architecture design.
The implementation and trained models will be made publicly available upon publication to facilitate reproducibility.

1. Introduction

Recent advances in unmanned aerial vehicle (UAV) platforms and onboard sensors have greatly improved access to very high-resolution imagery, enabling flexible data acquisition with fine spatial detail. This has led to the increasing use of deep learning for automated building extraction, achieving promising results (He et al., 2022) and supporting applications in resource management, environmental assessment and city planning (Bittencourt et al., 2024). However, conventional convolution-based networks still struggle to maintain geometric fidelity and precise rooftop delineation, especially when balancing accuracy and computational efficiency for large-scale deployment.

In response, researchers have proposed many innovative designs in model architecture, including skip connections in UNet and shortcut connections in ResNet (Targ et al., 2016), atrous spatial pyramid pooling (ASPP) in DeepLab (Lian et al., 2021), multi-scale branches in Inception (Szegedy et al., 2016), and incorporating multi-path strategies to enhance feature fusion (Guo and Gan, 2024). With these advancements, the capability of deep learning models has been extended to support complex feature processing for urban building segmentation. However, they still lack the ability to adaptively extract and fuse features across channels (Schrage, 2024), due to the unchanged mechanism behind the convolutional kernel.

In a typical convolution layer of these networks, the kernel captures features by optimizing its parameters through backpropagation (Cui et al., 2017), facilitating receptive field variation and diverse feature representation (Mairal et al., 2014). However, it only focuses on fixed channel specificity, limiting its capability to integrate cross-channel features (Lin et al., 2017). Despite deep networks with a large number of parameters, this structure often produces similar outputs across channels (Qiu et al., 2022). To address this limitation, SENets (Hu et al., 2018) introduced a Squeeze-and-Excitation (SE) module, which recalibrates the importance of feature channels by globally aggregating feature maps to compute channel-wise attention weights, and CBAM (Woo et al., 2018), which applies sequential spatial and channel attention. Self-attention, first introduced in the Transformer architecture (Vaswani, 2017), enables models to selectively focus on the most relevant features by capturing long-range dependencies and global context, and was later extended to vision tasks through Vision Transformers (ViTs). However, despite their strong capability in modelling global context, ViTs often struggle with dense prediction tasks such as segmentation: (1) decoders tend to lose fine-grained details, leading to over-smoothed outputs; (2) self-attention disrupts the hierarchical feature structure of convolutional neural networks (CNNs); and (3) its quadratic computational complexity makes ViTs expensive for high-resolution imagery, such as in medical or remote sensing applications. ShuffleNet (Zhang et al., 2017) explicitly partitions channels and introduces channel shuffle operations to enhance information exchange across groups. Additionally, Capsule Networks (Sabour et al., 2017) represent features as vectors and apply dynamic routing to capture part-whole relationships.

In this study, we propose the Cube Kernel, a novel cross-channel convolutional operator that enhances feature extraction and fusion for improved rooftop delineation in high-resolution imagery. Our key contributions include:

(1) **Cube Kernel as an Enhanced Convolutional Mechanism**: We present the Cube Kernel, which redesigns the convolutional process with a specialized grouping strategy to reconstruct the cross-channel kernel structure. This design strengthens cross-channel feature interaction, enables local gradients to propagate across channels, preserves geometric

fidelity in segmentation outputs, and achieves structured sparsity with lower FLOPs.

(2) **Consistent Performance Gains without Extra Cost**: The Cube Kernel improves rooftop segmentation accuracy by approximately +2% F1 across multiple architectures and datasets, including Swin-UNet, DeepLabV3+, and UNet, without increasing parameter counts or computational cost.

(3) **Scalable Architectural Design**: Beyond serving as a plug-in module, the Cube Kernel can be stacked to form a full hierarchical backbone. The proposed ConvNeXt-Cube architecture achieves superior performance on both large and small datasets with low computational cost, demonstrating that the Cube Kernel is not only an effective enhancement module, but also a scalable building block for high-performance segmentation networks.

2. Related Work

2.1 Grouped Convolution

Grouped convolution divides the input channels of a convolutional layer into multiple groups, each processed by separate filters. This reduces computational costs by allowing subsets of channels to be processed independently and improving efficiency without compromising feature representation (Liu et al., 2022a, Akay et al., 2021). An example in image segmentation of grouped convolution is its application in MobileNetV2, a lightweight deep neural network optimized for mobile and embedded devices that enables the model to balance efficiency, making it particularly effective for tasks requiring high-resolution segmentation on resource-constrained devices (Kanadath et al., 2021). In MobileNetV2, depthwise separable convolutions are used for splitting the standard convolution process (Tan and Le, 2019), and then used as the backbone of DeepLabV3+ (Wang et al., 2024).

2.2 Cross-channels Mechanisms in CNN

Channel-wise attention mechanisms have emerged as powerful tools for enhancing feature representations by modelling dependencies across channels, thus bolstering a network’s ability to capture intricate patterns and relationships in input data. Traditional CNNs, which process each channel independently, cannot often fully model these inter-channel dependencies, limiting their representational effectiveness. To address this limitation, various techniques have been developed, among which the Squeeze-and-Excitation (SE) Network (Hu et al., 2018) and Convolutional Block Attention Module (CBAM) (Woo et al., 2018) are notable. The SE Network introduces an efficient approach for channel attention through a “squeeze“ operation to capture global spatial information and an “excitation“ operation to recalibrate the significance of each channel (Hu et al., 2018). CBAM’s dual attention mechanism offers a more comprehensive attention framework, enabling networks to achieve enhanced performance across a range of vision applications by focusing on spatially and channel-wise critical features. This two-step process enables the SE Network to selectively emphasize essential feature channels, demonstrating marked improvements in various vision tasks.

2.3 Attention and Transformer-based Mechanisms

Involution (Li et al., 2021) departs from standard convolution by generating input-conditioned, position-specific weights that reweight features within each channel group (band), emphasizing intra-band relationships. This weighting mechanism has shown advantages over standard convolutions on tasks demanding fine detail and spatial coherence (He et al., 2024).

Swin Transformer(Liu et al., 2021), improves computational efficiency and spatial detail preservation by incorporating hierarchical feature maps and a shifted window mechanism, enabling multi-scale feature extraction while reducing memory overhead. These advancements allow transformer-based models like Polygon Transformer(Liang et al., 2020) to surpass CNNs in building rooftop delineation tasks.

Further improving upon these developments, BuildFormer(Wang et al., 2022) introduced a dual-path structure, achieving state-of-the-art results on the Massachusetts building dataset by balancing global context with fine-grained spatial details. LinFormer(Wang et al., 2020b), designed for long-range dependency modelling, enhances feature consistency across spatially distant regions, making it particularly effective for large-scale geospatial analysis.

Additionally, Swin-Unet(Cao et al., 2021), a fusion of Swin Transformer and U-Net, leverages hierarchical self-attention for multi-scale feature extraction and spatial preservation. This hybrid approach has demonstrated strong performance in medical image segmentation and remote sensing applications, underscoring the broader potential of transformer-based architectures beyond traditional CNNs.

3. Methodology of Cube Kernel

CNNs have achieved notable success in feature extraction; however, their ability to model cross-channel interactions remains fundamentally constrained. In standard convolution, each $k \times k$ kernel processes input channels independently, and the resulting feature maps are fused through a fixed element-wise summation. This fusion is non-learnable and uniform, forcing all channels to contribute at every spatial location, even when only a subset is relevant. Such rigid full-channel coupling restricts the flexibility of channel collaboration, leads to unnecessary computation, and often requires multiple stacked layers to achieve the selective cross-channel behaviour that a more adaptive mechanism could provide within a single layer.

To overcome these limitations, we introduce the Cube Kernel (Figure 1), a learnable cross-channel convolutional operator that performs feature fusion at the local-gradient level. Unlike standard convolutions that depend solely on global error signals during backpropagation, Cube Kernel explicitly restructures feature channels into a 3D cube representation, enabling localized cross-channel interaction. Within this cube, grouped channels exchange information through a convolutional sweep that propagates gradients across both spatial and channel dimensions. This mechanism establishes direct channel-to-channel coupling within local gradients, resulting in more efficient feature coordination and faster optimization across channels.

Local and global gradients in standard vs Cube convolution

This limitation can be better understood by examining the gradient computation in standard convolution. The gradient of the loss function L with respect to the convolution weight W is:

$$\frac{\partial L}{\partial W} = \sum_{i=1}^{C_{\text{out}}} \sum_{j=1}^{C_{\text{in}}} \underbrace{\frac{\partial L}{\partial Z_i}}_{\text{Global Gradient}} \cdot \underbrace{\frac{\partial Z_i}{\partial W_{i,j}}}_{\text{Local Gradient}} \tag{1}$$

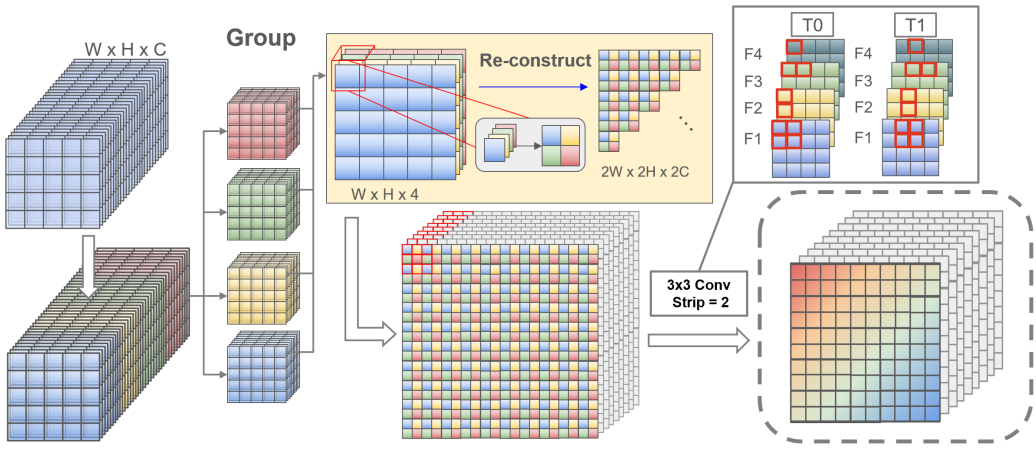

Figure 1. Cube Kernel: Grouping and Pixel Re-Construction Process

where $\frac{\partial L}{\partial Z_i}$ denotes the Global Gradient, providing the overall optimization direction across channels, and $\frac{\partial Z_i}{\partial W_{i,j}}$ represents the Local Gradient, describing how an individual weight contributes to the output. Critically, the Local Gradient is tied to a fixed input–output channel pair (i, j), meaning each weight update is influenced only by the corresponding input channel. As a result, standard convolution lacks a direct mechanism for selective cross-channel coordination at the local gradient level; any cross-channel dependency emerges only indirectly through accumulated Global Gradients over multiple layers.

To address this structural limitation, we propose the Cube Kernel, a learnable cross-channel convolutional mechanism that introduces channel interaction directly into the Local Gradient. The gradient computation becomes:

$$\frac{\partial L}{\partial W_{\text{Cube}}} = \sum_{i=1}^{C_{\text{out}}} \sum_{j \in G} \frac{\partial L}{\partial Z_i} \cdot \underbrace{\frac{\partial Z_i}{\partial W_{\text{Cube},i,j}}}_{\text{Local Gradient (Cross Channel)}} \tag{2}$$

where G denotes a learnable or predefined group of input channels (e.g., $G = C_{\text{in}}/4$). By allowing each output channel to receive Local Gradient contributions from a selected channel group rather than a single input channel, Cube Kernel enables direct and selective cross-channel coordination within a single layer (e.g. a 4D tensor $X_{\text{cube}} \in R^{N_{\text{group}} \times G \times H \times W}$).

A pseudo-code example of the gradient computation process is provided in Figure 2 to illustrate the sequential flow of backpropagation in standard convolution, with input $X \in R^{C_{\text{in}} \times H \times W}$, output $Y \in R^{C_{\text{out}} \times H' \times W'}$. The input activation X is a four-dimensional tensor consisting of all input channels and spatial locations. The output activation Y is the corresponding feature map produced after the convolution operation. The Global Gradient $\mathrm{d}Z$ is first computed and then multiplied with the input activations to obtain the Local Gradient $\mathrm{d}W$, reflecting the independent contribution of each channel pair.

Although cross-channel interaction can be improved by stacking more convolutional layers (Chollet, 2017, Yi et al., 2017), such approaches provide only indirect and inefficient channel collaboration. Additive fusion blends channels uniformly, often merging irrelevant features, while concatenation simply stacks channels without true interaction—both creating redundancy.

Step 1: Compute the Loss
Compute the loss: $Loss = \text{ComputeLoss}(A[L], Y)$
Step 2: Backpropagation
Initialize: $dA[L] = \text{ComputeGradientOfLoss}(A[L], Y)$
Compute Gradients:
for each layer l from L to 1:
Compute global gradient: $dZ[l] = dA[l] \times$ Activation Derivative
Compute local gradient: $dW[l] = dZ[l] \times A[l-1]$
Compute bias gradient: $db[l] = \text{Sum}(dZ[l])$
Propagate gradient to previous layer: $dA[l-1] = dZ[l] \times W[l]$
Step 3: Update Parameters
for each layer l from 1 to L:
Update weights: $W[l] = W[l] - \text{LearningRate} \times dW[l]$
Update biases: $b[l] = b[l] - \text{LearningRate} \times db[l]$

Figure 2. Pseudo-code for Gradient Computation

This also explains why deeper networks sometimes perform better: cross-channel information is accumulated gradually rather than learned directly.

In contrast, Cube Kernel enables direct and selective cross-channel coordination during backpropagation. Its parameters are updated using Local Gradients that integrate information from multiple channels, with the Global Gradient modulating their interaction. Since different channels encode related yet distinct cues (e.g., edges, textures, regions), Cube Kernel refines local features more precisely by exploiting subtle inter-channel differences, substantially enhancing representational expressiveness.

Cube Kernel forward propagation

Given an input feature map $X \in R^{C_{\text{in}} \times H \times W}$, we assign to each output channel i a channel group $G(i) \subset \{1, \ldots, C_{\text{in}}\}$ of fixed size $|G(i)| = G$ (e.g., $G = C_{\text{in}}/4$). The pre-activation of the Cube Kernel at spatial location (h, w) is then defined as

$$Z_i(h, w) = \sum_{j \in G(i)} \sum_{u,v=1}^{K} W_{\text{Cube},i,j,u,v}\, X_j(h+u-\delta,\ w+v-\delta), \tag{3}$$

where W_{Cube} denotes the learnable Cube Kernel weights, K is the kernel size, and $\delta = \lfloor K/2 \rfloor$ is the padding offset. The output feature is obtained as $Y_i(h, w) = \phi\big(Z_i(h, w)\big)$ with a non-linear activation $\phi(\cdot)$. This formulation restricts each output channel to interact with only G input channels while still enabling learnable cross-channel fusion within each group, consistent with the structured sparsity and local-gradient behaviour described above.

Structured Sparsity

By introducing structured sparsity, it promotes more efficient learning of diverse feature combinations. This results in a higher-rank factorization, allowing the network to capture more intricate

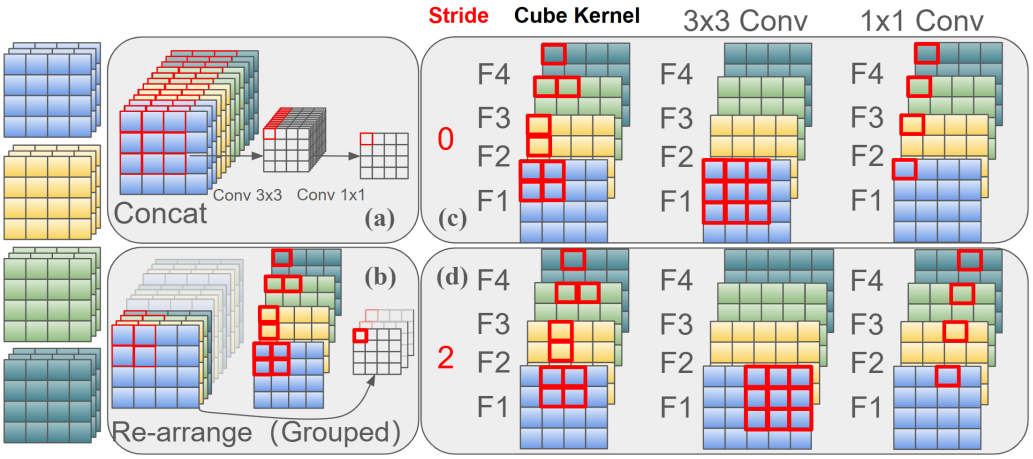

Figure 3. Receptive Field Overview. This figure illustrates the receptive fields of (a) 12 iterations of a standard 3×3 kernel; (b) a single Cube Kernel operation; and (c, d) the Cube Kernel compared with 3×3 and 1×1 kernels at time $T0$ and $T2$.

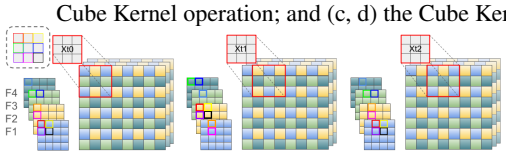

Figure 4. Cross-channel Feature Fusion by Cube Kernel.

patterns and approximate a richer set of functions compared to standard convolution with the same number of layers. As shown in Figure 4, displays a 3×3 Cube Kernel on the top left corner, which assembles features across channels at each time stamp based on a specialized grouping strategy, with each coloured cell corresponding to a specific position of each channel. Next, on each generated feature fusion layer, The primary feature selection of the kernel (determined by the highest pixel count from a specific feature) will vary depending on the stride (e.g., $Xt1$'s yellow cells versus $Xt0$' and $Xt2$'s light blue cells). The kernel's adaptive can enhance feature extraction by maintaining the primary features over time when stride. When the stride is set to 2, this primary feature remains consistent across layers. This consistency is particularly beneficial for tasks that require stable feature representation in complex environment.

While a single Cube Kernel layer supports 4-layer interactions, stacking multiple layers progressively expands the effective receptive field along the channel dimension (Figure 3). Combined with spatial convolutions, repeated Cube Kernel layers enable global cross-channel feature aggregation, capturing richer dependencies and improving representation learning. This makes Cube Kernel suitable for extraction tasks requiring fine spatial detail, efficient gradient propagation, and structured feature fusion—areas where transformers often struggle.

FLOPs Reduction

Local Channel Interaction: Each output channel interacts with only G input channels instead of all C_{in}, where $G = \frac{C_{\text{in}}}{4}$. Grouped Convolutions: The second convolution operates in a grouped manner with dilation. The Cube Kernel FLOPs are calculated as:

$$\text{FLOPs}_{\text{Cube}} = 2 \times \left(\frac{C_{\text{in}}}{4}\right) \times C_{\text{out}} \times K^2 \times H \times W \quad (4)$$

Thus, each Cube Kernel layer only requires 25% of the computations compared to a standard convolution layer.

Gradient Propagation and Optimization Stability

In addition to its advantages in feature representation, the Cube Kernel also contributes to more stable optimization dynamics during training. Conventional layers often suffer from uneven gradient propagation across channels, which leads to anisotropic curvature in the loss landscape and unstable parameter updates. By introducing structured inter-channel fusion, the Cube Kernel alleviates this issue and enables more consistent gradient flow.

From a learning dynamics perspective, the Cube Kernel facilitates more uniform backpropagation through both spatial and channel dimensions. The enhanced coupling between channels promotes balanced parameter updates, effectively minimizing redundant oscillations in the optimization trajectory. Consequently, the model achieves faster convergence and better generalization across datasets of different scales (e.g., WHU vs. WBD), confirming that cross-channel gradient consistency plays a crucial role in stable and efficient deep network optimization.

Overall, standard convolution relies on global gradient aggregation across layers to achieve cross-channel coupling, making such interaction indirect, inefficient, and spatially coarse. Cube Kernel introduces a new optimization pathway by enabling local-gradient-level cross-channel coordination, allowing channels to interact directly and selectively during backpropagation. By encoding cross-channel relationships into the kernel design itself, Cube Kernel effectively bridges the gap between spatially local convolutions and globally receptive transformer-style feature mixing, offering an efficient, scalable, and expressive alternative for modern deep learning architectures. This design not only enhances fine-grained feature modeling but also enables more stable optimization dynamics, faster convergence, and stronger generalization across datasets.

4. Extraction Experiments

All models are trained for 100 epochs with a batch size of 8 using Adam (LR = 1×10, BCE loss). Training is performed on a NVIDIA A100 GPU (32 GB, CUDA 12.6). All Cube variants and baselines share identical training schedules and augmentations.

The datasets used for this section are the Waterloo Building Dataset (WBD)(He et al., 2021) and the Wuhan University (WHU) Building Dataset (Ji et al., 2019)(Figure 5). Both WHU and WBD follow their official train/validation/test splits.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-XI-3-2026-109-2026 | © Author(s) 2026. CC BY 4.0 License.

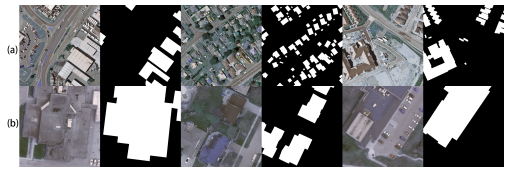

Figure 5. (a) WHU and (b) WBD

4.1 Effect of Kernel Stride Configuration

To evaluate the influence of kernel stride on performance, Cube Kernel with stride values of 1 and 2, as well as a standard convolutional kernel, were integrated into the final stage of UNet (Ronneberger et al., 2015) (Figure 6(b)) and tested on the WBD dataset.

Table 1. Comparison of Cube Kernel with Different Strides

Config	IoU	Accuracy	Recall	F1 Score	Kappa	Precision
Normal Kernel	0.7990	0.9725	0.9205	0.8984	0.8896	0.8784
Stride=1	0.8394	0.9847	0.9262	0.9127	0.9043	0.8996
Stride=2	**0.8607**	**0.9867**	**0.9289**	**0.9252**	**0.9178**	**0.9254**

Across all evaluation metrics, the stride = 2 configuration achieved the best performance (Table 1). Both Cube Kernel variants outperformed the standard convolution, and using a stride of 2 yielded absolute IoU improvements of +0.04 and +0.06 over the stride = 1 and normal kernel settings, respectively. These findings highlight the efficacy of using a stride of 2 in the Cube Kernel for accurate segmentation while preserving primary features within the same layer serves as a boosting mechanism for feature analysis. Consequently, all subsequent experiments adopt a stride setting of 2.

4.2 Ablation Experiments

Single-Module Placement Analysis

To further assess the joint effects of the two cross-channel modules, we designed four combination configurations, as illustrated in Figure 6: (a) Cube Kernel + UNet, (b) UNet + Cube Kernel, (c) UNet + Involution (Li et al., 2021), and (d) Involution + UNet.

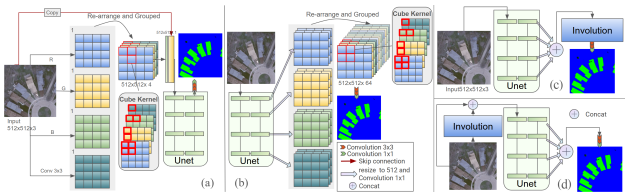

Figure 6. Architecture of the Cube Kernel Ablation Experiments

As shown in Table 2 and in Figure 8, the placement of Cube Kernel within the UNet architecture has a notable impact on performance. Among all configurations, “UNet + Cube” achieved the highest scores across most metrics (IoU = 0.8607, F1 = 0.9252, Kappa = 0.9178), indicating that applying Cube Kernel at the final stage is more effective for refining high-level feature representations before prediction. In contrast, “Cube + UNet” showed weaker performance, suggesting that early integration provides less benefit. For Involution, “Involution + UNet” achieved moderate results, whereas “UNet + Involution” led to a substantial drop (IoU = 0.7054), indicating that Involution is highly sensitive to its placement and may disrupt feature encoding when positioned at later stages of the network.

Joint-Module Interaction Analysis

To evaluate the combined effects of these two modules on network performance, we designed two experiments in Figure 7: (a) Involution + Cube-Kernel + UNet; and (b) Involution + UNet + Cube-Kernel).

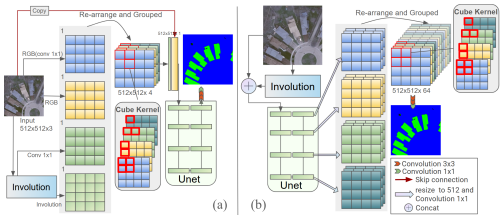

Figure 7. Architecture of the Combination Experiments

Furthermore, the combined configurations did not yield complementary gains. Both hybrid settings underperformed compared to using Cube Kernel alone, suggesting that simultaneous application of the two modules offers no additive benefit and may negatively affect performance stability. This outcome implies that the two mechanisms may not be fully compatible when applied together within the same architecture.

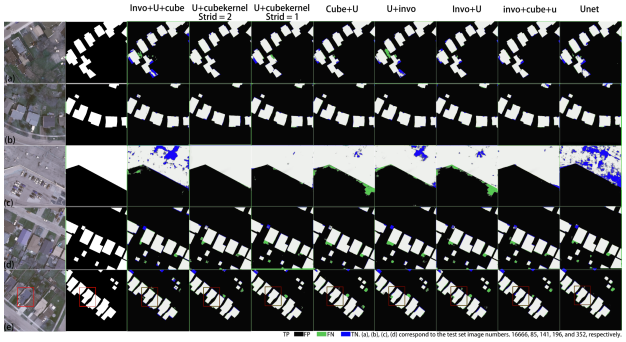

Figure 8. Result of Ablation Experiments

Table 2. The Cube Kernel and Involution Ablation Experiments

Configuration	IoU	Accuracy	Recall	F1 Score	Kappa	Precision
UNet + Cube	**0.8607**	**0.9867**	0.9289	**0.9252**	**0.9178**	**0.9254**
Cube + UNet	0.8224	0.9826	0.9018	0.9025	0.8930	0.9033
Involution + UNet	0.8380	0.9847	**0.9351**	0.9119	0.9034	0.8897
UNet + Involution	0.7054	0.9659	0.7550	0.8272	0.8085	0.9148
Invo+UNet+Cube	0.7762	0.9779	0.8873	0.8740	0.8619	0.8611
Invo+Cube+UNet	0.8083	0.9810	0.8876	0.8940	0.8835	0.9005

4.3 Comparative Experiments

To evaluate the effectiveness and generalizability of the proposed Cube Kernel, we conducted comparative experiments on both the WHU and WBD datasets. These datasets pose different levels of difficulty, with WBD featuring larger scene diversity and more complex building structures, while WHU contains smaller and more homogeneous regions. We selected UNet, DeepLabv3+, and Swin-UNet as baseline models and replaced all standard convolutions in their decoder modules with Cube Kernel layers. Performance was assessed using IoU, Recall (Rec), Precision (Prec), F1-score (F1), number of Parameters (Params), and FLOPs to jointly evaluate accuracy and efficiency.

In DeepLabv3+-Cube, Swin-UNet-Cube, and Swin Transformer-Cube, the standard convolution layers at the highlighted positions in Figure 9 were replaced with Cube Kernel layers.

The results in Table 3 and in Figure 10 show that Cube Kernel consistently improves segmentation accuracy across all three

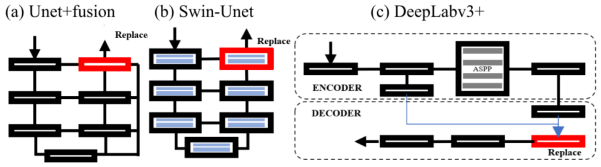

Figure 9. Locations of Convolution Layers Replaced by Cube Kernel Across Different Architectures

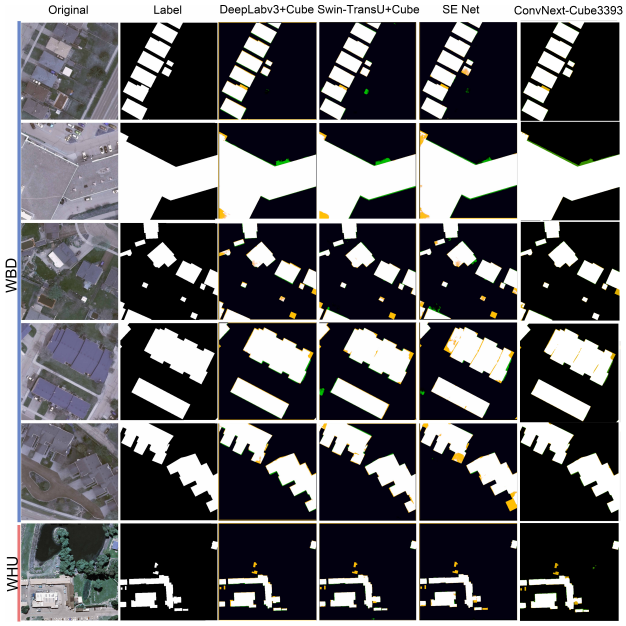

Figure 10. Result of Comparative Experiments

baseline models. Notably, across both datasets and all three baselines (UNet, DeepLabV3+, and Swin-UNet), Cube Kernel yields an improvement of approximately 2%.

Transformer-based models also benefit significantly from Cube Kernel. Swin-UNet_Cube improves IoU from 0.8089 to 0.8726 and F1 from 0.9002 to 0.9360 on WBD, suggesting that Cube Kernel complements self-attention by enhancing local spatial feature modeling. It effectively bridges CNN-style local feature extraction with transformer-based global context modeling, enabling more discriminative feature fusion across architectures.

Cube Kernel not only improves accuracy but also reduces model complexity. For example, Swin-UNet_Cube reduces the parameter count from 42.03M to 29.8M and FLOPs from 192.7G to 100.58G, making it a lightweight yet high-performing alternative. Across all Cube Kernel variants, a consistent improvement in Precision is observed, indicating fewer false positives—particularly valuable in visually ambiguous regions where artificial surfaces resemble rooftops.

Overall, the results demonstrate that Cube Kernel enhances both convolutional and transformer-based models, offering a versatile, scalable, and efficient component for high-performance image segmentation.

Beyond Plug-and-Play Integration

Beyond plug-and-play usage, we further explore whether Cube Kernel can scale into a full hierarchical architecture rather than serving solely as a refinement module. To this end, we construct a Cube-based encoder–decoder network, termed ConvNeXt-Cube (show in Figure 11). We adopt a ConvNeXt-Tiny backbone with a stage configuration of [3, 3, 9, 3] and channel dimensions [96, 192, 384, 768]. The model adopts a ConvNeXt (Liu et al., 2022b) encoder initialized with pretrained weights, while all convolution layers in both the encoder and the symmetric decoder are replaced with Cube Kernel Blocks. This design preserves the ConvNeXt 3–3–9–3 hierarchical depth while enabling Cube Kernel to act as the core computational operator throughout the network.

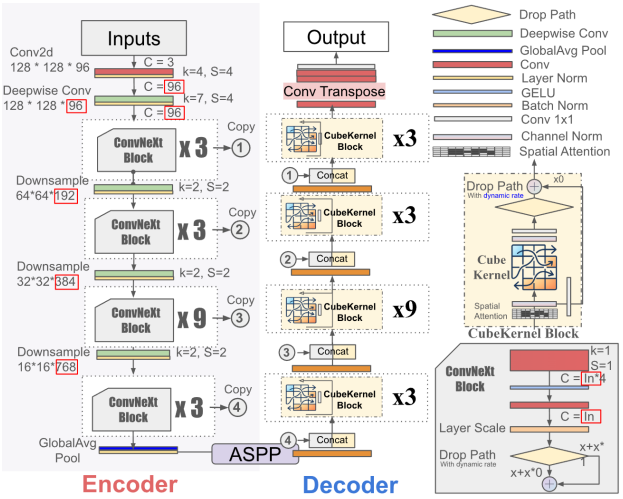

Figure 11. Structure of ConvNeXt-Cube Network

As shown at the bottom of Table 3, ConvNeXt-Cube achieves the best overall performance among all evaluated models, reaching 0.9095 IoU / 0.9535 F1 on WBD and 0.9133 IoU / 0.9547 F1 on WHU, while maintaining competitive model complexity (45.35M parameters, 48.41G FLOPs). These results confirm that Cube Kernel is not only effective as a plug-in module but can also be stacked to form deeper hierarchical networks, serving as a scalable building block for next-generation CNN architectures.

5. Discussion

Involution is a channel-interaction operator that enhances a network’s ability to emphasize task-relevant regions rather than performing learnable feature fusion. As illustrated in Figure 12(a), it selectively highlights salient areas of the input, functioning as a feature filter that suppresses ambiguous responses and retains high-confidence cues. In contrast, Cube Kernel performs learnable cross-channel fusion, enabling richer feature interaction and more adaptive representation learning. Such fusion allows network depth to be partially replaced by intelligent gradient-level feature integration rather than adding more convolution blocks.

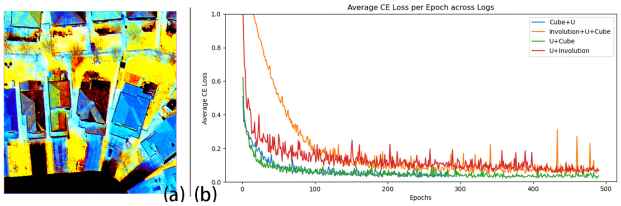

Figure 12. (a) Output of Involution Preview, and (b) Loss curves of “Cube Kernel+ UNet”, “UNet + Cube Kernel”, “Involution + UNet+ Cube Kernel”, and “UNet + Involution” in WBD.

As shown in Figure 12(b), models incorporating Cube Kernel exhibit smoother loss convergence, whereas configurations combining Involution and Cube Kernel show early loss saturation, reflecting optimisation conflicts. These observations reveal a

Table 3. Performance Comparison of Different Models on WBD and WHU Datasets

	WBD (Larger and Complex)				WHU (Smaller and Less Complex)				Model Complexity	
Model	IoU	Rec	F1	Prec	IoU	Rec	F1	Prec	Params	FLOPs
Fast SCNN (Poudel et al., 2019)	0.6300	0.7010	0.6540	0.6780	0.6240	0.6204	0.6503	0.6790	1.1	38.65
Mask R-CNN (He et al., 2017)	0.7060	0.7010	0.7850	0.8920	0.7662	0.7780	0.8120	0.8500	44	370.99
HRNet-W48 (Wang et al., 2020a)	0.7660	0.8170	0.8580	0.9250	0.8564	0.9285	0.9207	0.9169	65	1900
SEnet (Hu et al., 2018)	0.7905	0.8288	0.8830	0.9248	0.8670	0.9024	0.9288	0.9567	31.26	385.85
UNet (Ronneberger et al., 2015)	0.7775	0.9104	0.8748	0.8419	0.7938	0.8502	0.8856	0.9228	27.22	364
UNet + Conv Kernel	0.7990	0.9205	0.8984	0.8784	0.8274	0.8680	0.8973	0.9233	27.21	402.8
UNet + Cube Kernel (Ours)	0.8607	0.9289	0.9252	0.9254	0.8665	0.8984	0.9284	0.9618	27.14	315.76
DeepLabv3+ (Chen et al., 2018)*	0.8543	0.8857	0.9214	0.9602	0.8617	0.8943	0.9257	0.9594	41.2	178.72
DeepLabv3+Cube (Ours)*	0.8903	**0.9468**	0.9419	0.9401	0.8693	0.9169	0.9421	0.9546	40	105
Swin-Unet (Cao et al., 2021)	0.8089	0.8705	0.9002	0.9295	0.8500	0.9269	0.9189	0.9110	42.03	192.7
Swin-UCube (Ours)	0.8726	0.8987	0.9360	0.9529	0.8859	**0.9395**	0.9472	0.9521	29.8	100.58
ConvNeXt-Cube	**0.9095**	0.9422	**0.9535**	**0.9633**	**0.9133**	0.9341	**0.9547**	**0.9761**	45.35	48.41

Rec = Recall, F1 = F1 Score, Prec = Precision, Params = Parameters (M), FLOPs = Floating Point Operations of 3×512×512(G).

clear functional distinction: Involution acts as a selective feature enhancer, yielding higher Precision but lower Recall, while Cube Kernel provides balanced cross-channel fusion with stable convergence. Feature visualisations in Figure 13 show that when Involution precedes Cube Kernel, the inputs are semantically inconsistent across scales, limiting fusion effectiveness. This suggests that channel-interaction modules should not be stacked without coordination.

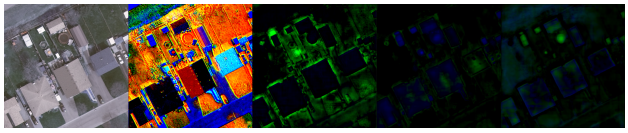

Figure 13. Visualizes the input features for Cube Kernel: (a) the original image, (b) the feature extracted by Involution, which serves as the main feature for Cube Kernel after the skip connection, and (c), (d), and (e) the features from the 2nd, 3rd, and 4th levels of UNet.

Beyond plug-in usage, Cube Kernel also proves scalable as an architectural primitive. Its consistent gains across CNN and Transformer models—without architecture-specific tuning—and its success as the backbone of ConvNeXt-Cube highlight strong generalisability. This indicates potential for broader perception tasks such as large-scale scene parsing, and multi-modal fusion. Moreover, the results suggest that intelligent gradient-level channel fusion can partially replace depth, offering a more efficient alternative to stacking convolution blocks.

In future work, Cube Kernel could be integrated into different network architectures, where its robust feature fusion capabilities may further boost performance. The code is designed with modularity and ease of integration, enabling seamless application across diverse architectures and suggesting numerous potential applications.

6. Conclusion

In this study, we propose Cube Kernel, a lightweight and efficient approach for CNN-based image feature fusion. Cube Kernel restructures the convolutional kernel to enable direct local gradient computation across channels, restoring convolution’s sensitivity to inter-band interactions while maintaining minimal computational overhead. Ablation experiments on UNet using the Waterloo Building Dataset (WBD) demonstrated that Cube Kernel outperformed in feature fusion. Further, comparative experiments integrating Cube Kernel into DeepLabv3+ and Swin-Unet on the WHU and WBD datasets showed significant improvements across all three models. These gains demonstrate that Cube Kernel enhances feature representation in both CNN- and Transformer-based networks, complementing global self-attention with stronger local feature fusion. Furthermore, the Cube-based ConvNeXt-Cube demonstrates that Cube Kernel can be effectively stacked into deeper hierarchical networks. By replacing all convolutional layers with Cube Kernel blocks, ConvNeXt-Cube achieved the highest performance on both datasets (0.9095 IoU / 0.9535 F1 on WBD and 0.9133 IoU / 0.9547 F1 on WHU), confirming that Cube Kernel is not only an effective refinement module but also a scalable and stackable architectural building unit.

The adaptability of Cube Kernel underscores its potential as a versatile solution for multi-scale feature fusion in segmentation tasks. Future work will focus on advancing network designs to further exploit Cube Kernel’s capabilities, and we expect it to inspire new developments in cross-channel interaction mechanisms within convolutional neural networks.

References

Akay, M., Du, Y., Sershen, C. L., Wu, M., Chen, T. Y., Assassi, S., Mohan, C., Akay, Y. M., 2021. Deep learning classification of systemic sclerosis skin using the MobileNetV2 model. IEEE Open Journal of Engineering in Medicine and Biology, 2, 104–110.

Bittencourt, J. C. N., Costa, D. G., Portugal, P., Vasques, F., 2024. A Survey on Adaptive Smart Urban Systems. IEEE Access.

Cao, H., Wang, Y., Chen, J., Jiang, D., Zhang, X., Tian, Q., Wang, M., 2021. Swin-unet: Unet-like pure transformer for medical image segmentation.

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H., 2018. Encoder-decoder with atrous separable convolution for semantic image segmentation. Proceedings of the European conference on computer vision (ECCV), 801–818.

Chollet, F., 2017. Xception: Deep learning with depthwise separable convolutions. Proceedings of the IEEE conference on computer vision and pattern recognition, 1251–1258.

Cui, Y., Zhou, F., Wang, J., Liu, X., Lin, Y., Belongie, S., 2017. Kernel pooling for convolutional neural networks. Proceedings of the IEEE conference on computer vision and pattern recognition, 2921–2930.

Guo, Z., Gan, H., 2024. Cpp-net: Embracing multi-scale feature fusion into deep unfolding cp-ppa network for compressive sensing. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 25086–25095.

He, H., Jiang, Z., Gao, K., Narges Fatholahi, S., Cai, Y., Tan, W., Hu, B., Qing, L., Xu, H., Li, J., 2021. Waterloo Building Dataset.

He, K., Gkioxari, G., Dollár, P., Girshick, R., 2017. Mask r-cnn. Proceedings of the IEEE international conference on computer vision, 2961–2969.

He, Z., Cai, Y., He, H., Xian, X., Barrett, B., 2024. An enhanced trans-involution network for building footprint extraction from high resolution orthoimagery. IGARSS 2024-2024 IEEE International Geoscience and Remote Sensing Symposium, IEEE, 10030–10034.

He, Z., DING, H., AN, B., 2022. E-Unet: a atrous convolution-based neural network for building extraction from high-resolution remote sensing images. Acta Geodaetica et Cartographica Sinica, 51(3), 457. http://xb.chinasmp.com/EN/abstract/article$_1$2701.$shtml$.

Hu, J., Shen, L., Sun, G., 2018. Squeeze-and-excitation networks. Proceedings of the IEEE conference on computer vision and pattern recognition, 7132–7141.

Ji, S., Wei, S., Lu, M., 2019. Fully Convolutional Networks for Multisource Building Extraction From an Open Aerial and Satellite Imagery Data Set. IEEE Transactions on Geoscience and Remote Sensing, 57(1), 574-586.

Kanadath, A., Jothi, J. A. A., Urolagin, S., 2021. Histopathology image segmentation using mobilenetv2 based u-net model. 2021 International Conference on Intelligent Technologies (CONIT), IEEE, 1–8.

Li, D., Hu, J., Wang, C., Li, X., She, Q., Zhu, L., Zhang, T., Chen, Q., 2021. Involution: Inverting the inherence of convolution for visual recognition. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 12321–12330.

Lian, X., Pang, Y., Han, J., Pan, J., 2021. Cascaded hierarchical atrous spatial pyramid pooling module for semantic segmentation. Pattern Recognition, 110, 107622.

Liang, J., Homayounfar, N., Ma, W.-C., Xiong, Y., Hu, R., Urtasun, R., 2020. Polytransform: Deep polygon transformer for instance segmentation. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 9131–9140.

Lin, T.-Y., RoyChowdhury, A., Maji, S., 2017. Bilinear convolutional neural networks for fine-grained visual recognition. IEEE transactions on pattern analysis and machine intelligence, 40(6), 1309–1322.

Liu, T., Wang, S., Liu, Y., Quan, W., Zhang, L., 2022a. A lightweight neural network framework using linear grouped convolution for human activity recognition on mobile devices. The Journal of Supercomputing, 1–21.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B., 2021. Swin transformer: Hierarchical vision transformer using shifted windows. Proceedings of the IEEE/CVF international conference on computer vision, 10012–10022.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., Xie, S., 2022b. A convnet for the 2020s.

Mairal, J., Koniusz, P., Harchaoui, Z., Schmid, C., 2014. Convolutional kernel networks. Advances in neural information processing systems, 27.

Poudel, R. P., Liwicki, S., Cipolla, R., 2019. Fast-scnn: Fast semantic segmentation network. arXiv preprint arXiv:1902.04502.

Qiu, Y., Liu, Y., Chen, Y., Zhang, J., Zhu, J., Xu, J., 2022. A2SPPNet: Attentive atrous spatial pyramid pooling network for salient object detection. IEEE Transactions on Multimedia, 25, 1991–2006.

Ronneberger, O., Fischer, P., Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation. International Conference on Medical image computing and computer-assisted intervention, Springer, 234–241.

Sabour, S., Frosst, N., Hinton, G. E., 2017. Dynamic routing between capsules. Advances in neural information processing systems, 30.

Schrage, L., 2024. Visual AI for Sustainable Urban Development Computer Vision and Machine Learning Applications for Climate and Social Impact. PhD thesis, Massachusetts Institute of Technology.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., 2016. Rethinking the inception architecture for computer vision. Proceedings of the IEEE conference on computer vision and pattern recognition, 2818–2826.

Tan, M., Le, Q. V., 2019. Mixconv: Mixed depthwise convolutional kernels. arXiv preprint arXiv:1907.09595.

Targ, S., Almeida, D., Lyman, K., 2016. Resnet in resnet: Generalizing residual architectures. arXiv preprint arXiv:1603.08029.

Vaswani, A., 2017. Attention is all you need. Advances in Neural Information Processing Systems.

Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., Liu, W., Xiao, B., 2020a. Deep high-resolution representation learning for visual recognition.

Wang, L., Fang, S., Meng, X., Li, R., 2022. Building extraction with vision transformer. IEEE Transactions on Geoscience and Remote Sensing, 60, 1–11.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., Ma, H., 2020b. Linformer: Self-attention with linear complexity. arXiv preprint arXiv:2006.04768.

Wang, Y., Yang, L., Liu, X., Yan, P., 2024. An improved semantic segmentation algorithm for high-resolution remote sensing images based on DeepLabv3+. Scientific Reports, 14(1), 9716.

Woo, S., Park, J., Lee, J.-Y., Kweon, I. S., 2018. Cbam: Convolutional block attention module. Proceedings of the European conference on computer vision (ECCV), 3–19.

Yi, S., Ju, J., Yoon, M.-K., Choi, J., 2017. Grouped convolutional neural networks for multivariate time series. arXiv preprint arXiv:1703.09938.

Zhang, X., Zhou, X., Lin, M., Sun, J., 2017. Shufflenet: An extremely efficient convolutional neural network for mobile devices.