Summer Internship Project
Report

A Study on Enhanced Convolutional Methods for Image Segmentation Architectures

Submitted by

Mohd Yusuf
20EE10043

Under the guidance of

Professor Adway Mitra and Priyanka Goyal(PMRF Research Scholar)

Department of Center of Excellence in Artificial Intelligence
Indian Institute of Technology Kharagpur
Address

Summer Internship 2023

Contents

1 Objective

The objective of this project is to conduct a comprehensive study and evaluation of various Image semantic segmentation architectures, including Unet, AttenUnet, Unet++ and CapsuleNet, by replacing their standard convolutions with ten different types of convolutions. The ten convolution types that will be explored are:

1. Standard Convolution
2. Spatially Separable Convolution
3. Gaussian dynamic Convolution
4. Deformable Convolution
5. Adaptive deformable Convolution
6. Standard Asymmetric Convolution
7. Asymmetric Convolution
8. Gaussian Dynamic Asymmetric Convolution
9. Deformable Asymmetric Convolution
10. Adaptive Deformable Asymmetric Convolution

The primary goal is to investigate the impact of these novel convolutional techniques on the performance, efficiency, and robustness of these architectures for various image processing tasks, particularly focusing on image segmentation and object detection.

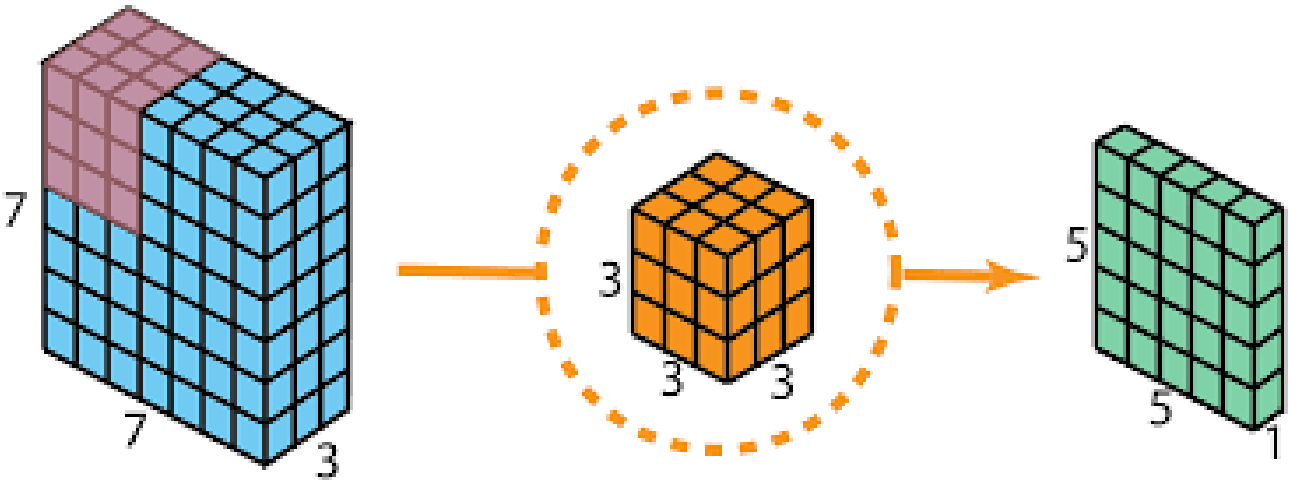
Through this project, we aim to identify which combinations these architectures and convolution types yield the most promising results in terms of accuracy, speed, and generalization capabilities. Furthermore, we seek to understand the underlying principles that contribute to the improved performance achieved by the enhanced models.

The findings from this study will not only contribute to the advancement of convolutional neural networks for image processing tasks but also offer valuable insights into selecting suitable architectural variations for specific applications and datasets. This research will serve as a stepping stone for future developments in the field of convolutional neural networks and their applications in computer vision and artificial intelligence.

2 Convolutions

We will explore ten different types of convolutions and their integration into U-Net architectures. These convolutions will be analyzed in the context of image processing tasks, with a particular emphasis on image segmentation and object detection.

2.1 Standard Convolution

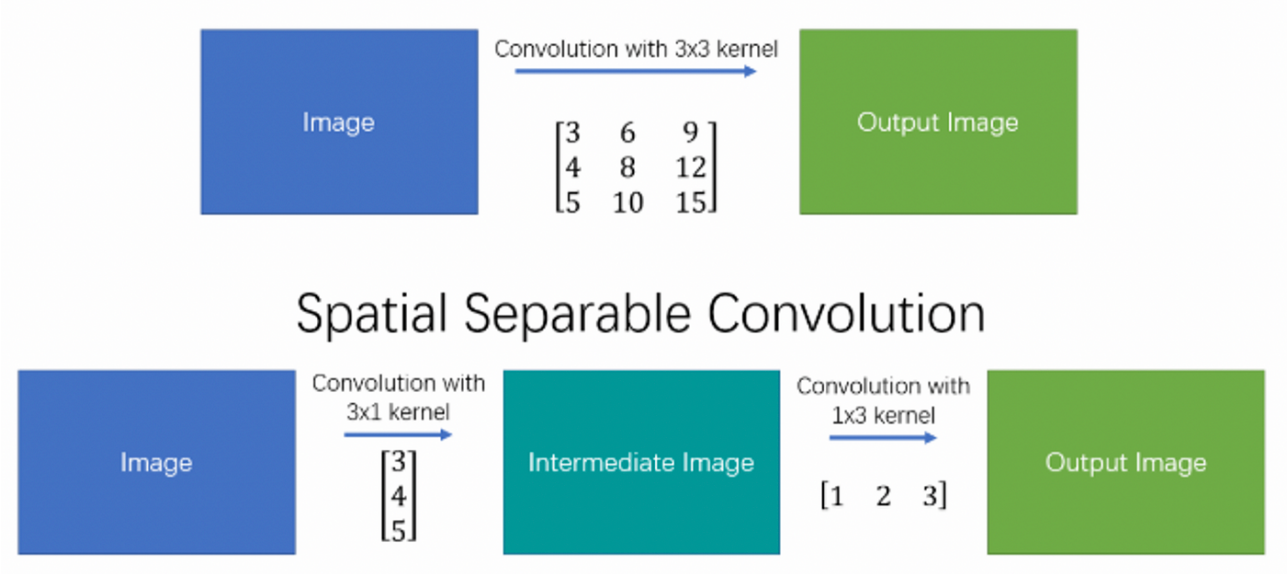


Standard 2D convolution is a mathematical operation where a small matrix, called a kernel, moves across a larger matrix, the image or input, calculating element-wise products at each position. This generates an output, also known as a feature map, reflecting local interactions between the input and the kernel.

Standard 2D convolution is essential for various image processing tasks like edge detection, blurring, sharpening, and feature extraction. The kernel used determines the specific impact on the input. For example, a kernel with positive values blurs an image, while one with positive and negative values is useful for edge detection.

2.2 Spatially Separable Convolution

Spatially separable convolution simplifies 2D kernels into two 1D ones (width and height). This lowers parameters and speeds up processing. For example, a 3x3 kernel becomes 3x1 and 1x3, reducing parameters from 9 to 6. The input image is convolved first vertically, then horizontally. Output matches the original kernel. Note: Not all kernels can be separated due to potential info loss or altered outputs, limiting practical use.



2.3 Gaussian dynamic Convolution

Gaussian dynamic convolution (GDC) uses adaptive receptive fields, adjusting the input image's influence on the convolutional layer output dynamically. Unlike traditional convolution, GDC employs Gaussian distribution offsets for spatial sampling, enabling the kernel to adapt its shape and size based on the input image. This flexibility captures relevant information across contexts and scales, making GDC suitable for segmentation. Essentially, GDC is a versatile, dynamic convolution method that molds itself to the input image characteristics.

For a standard 3×3 convolutional kernel, a feature map is obtained by sliding it over the input layer. Assuming the coordinates of the center of the convolutional kernel are represented by $d = (x, y)$, the coordinates of positions in the remaining eight directions can be represented as $d + \Delta_i \odot s_i$, where $s_i \in \{\langle -1, -1\rangle, \langle -1, 0\rangle, \ldots, \langle 1, 1\rangle\}$, and $\Delta = \langle 1, 1\rangle$. The expression is given as:

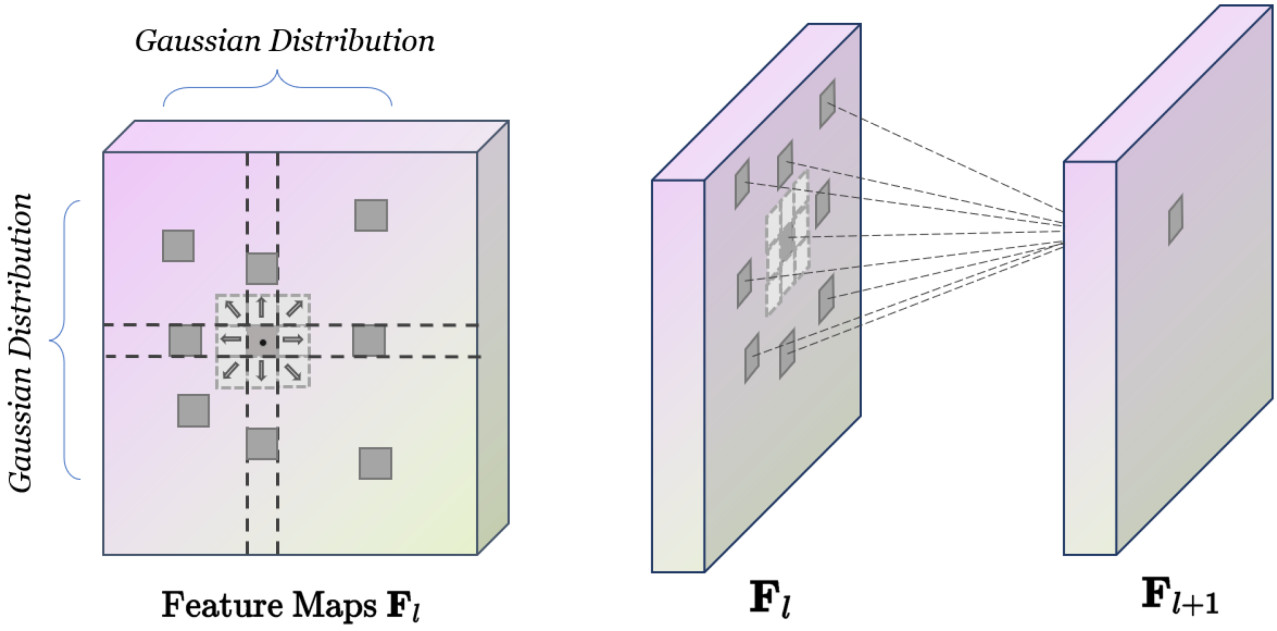
$$\mathbf{d}_i = \mathbf{d} + \Delta_i \odot \mathbf{s}_i, \quad where \quad \Delta = \langle 1, 1\rangle. \tag{1}$$

For example, the feature vector in the lower-left corner can be expressed as:

$$\langle x, y\rangle + \langle -1, -1\rangle \odot \langle 1, 1\rangle = \langle x - 1, y - 1\rangle. \tag{2}$$

We illustrate a toy example of Gaussian Dynamic Convolution (GDConv) in Figure **??**. We first fix the center weights and then assign a random offset Δ to each direction. The offset is obtained from a two-dimensional Gaussian distribution with standard deviation Σ. The Gaussian distribution equation is:

$$Gaussian(0, \Sigma) = \frac{1}{\sqrt{2\pi}\Sigma} \exp\left(-\frac{x^2}{2\Sigma^2}\right). \tag{3}$$



2.4 Deformable Convolution

Deformable convolution is an innovative technique that enables convolutional filters to adjust to object shapes and poses in input images by using learnable offsets. This allows filters to match object contours and capture details effectively.

The underlying working principle of deformable convolution can be summarized as follows:

1. Initially, a convolution generates offset fields from input features.
2. Offset fields modify sampling locations, allowing for bilinear interpolation.
3. Another convolution on deformed features produces the output map, with shared filters.

Deformable convolution enhances standard convolution by dynamically adjusting sampling locations based on input features. This flexibility improves filter versatility and expression, enabling effective handling of object shape and pose variations.

The convolution consists of two steps:

1) sampling using a regular grid R over the input feature map x;
2) summation of sampled values weighted by w. The grid R defines the receptive field size and dilation. For example, $R = \{(-1,-1),(-1,0),\ldots,(0,1),(1,1)\}$ defines a 3×3 kernel with dilation 1.

For each location p_0 on the output feature map y, we have

$$y(p_0) = \sum_{p_n \in R} w(p_n) \cdot x(p_0 + p_n), \quad (1)$$

where p_n enumerates the locations in R.

In deformable convolution, the regular grid R is augmented with offsets $\{\Delta p_n \mid n = 1, ..., N\}$, where $N = |R|$. Eq. (1) becomes

$$y(p_0) = \sum_{p_n \in R} w(p_n) \cdot x(p_0 + p_n + \Delta p_n), \quad (2)$$

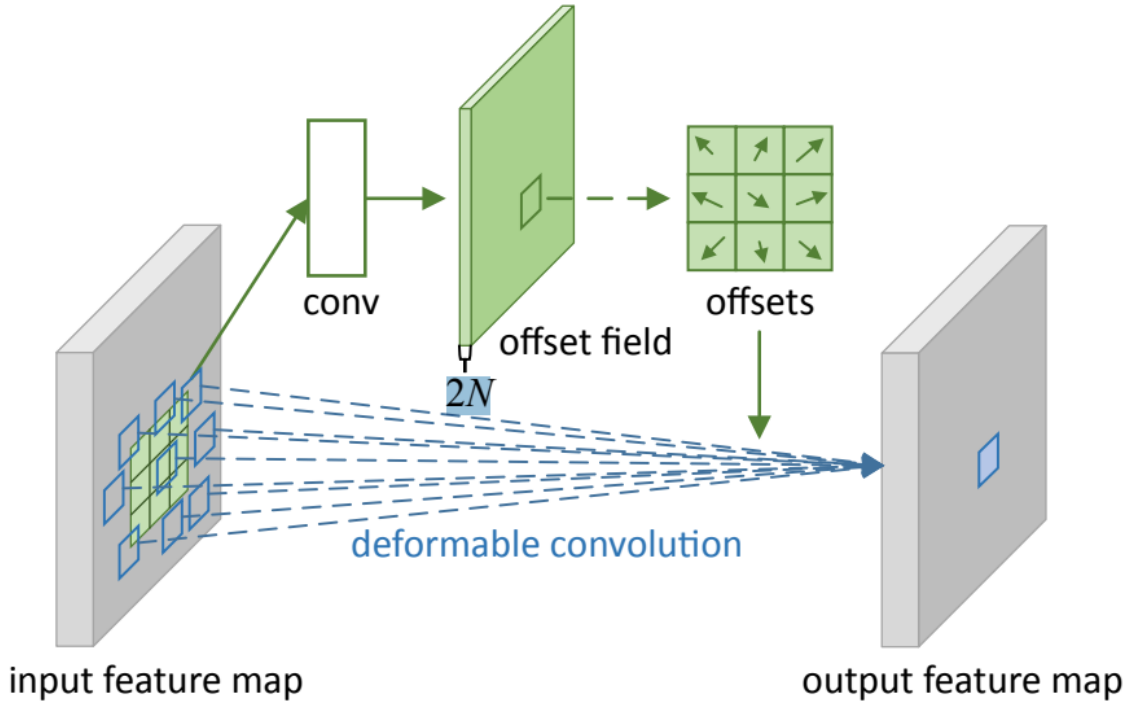
Now, the sampling is on the irregular and offset locations $p_n + \Delta p_n$. As the offset Δp_n is typically fractional, Eq. (2) is implemented via bilinear interpolation as

$$x(p) = \sum_q G(q, p) \cdot x(q), \quad (3)$$

where p denotes an arbitrary (fractional) location ($p = p_0 + p_n + \Delta p_n$ for Eq. (2)), q enumerates all integral spatial locations in the feature map x, and $G(\cdot, \cdot)$ is the bilinear interpolation kernel. Note that G is two-dimensional and is separated into two one-dimensional kernels as

$$G(q, p) = g(q_x, p_x) \cdot g(q_y, p_y), \quad (4)$$

where $g(a, b) = \max(0, 1 - |a - b|)$. Eq. (3) is fast to compute as $G(q, p)$ is non-zero only for a few qs.



In Figure, offsets are derived using a convolutional layer with the same spatial resolution and dilation as the current layer (e.g., also 3×3 with dilation 1). The output offsets match the input's spatial resolution, with a channel dimension of $2N$ representing N 2D offsets. Training involves learning both the output feature kernels and offsets simultaneously.

2.5 Adaptive deformable Convolution

Adaptive deformable convolution adjusts filters to input shape by adding learnable offsets to grid sampling. This enhances precision, capturing finer context information.

The method, called NC-A-DCN (Non-Local Context-Aware Deformable Convolutional Networks), utilizes non-local operations to capture long-range dependencies among input features. This enhances performance in computer vision tasks like object detection, semantic segmentation, and pose estimation by making the offsets adaptive to both local and global context.

To enhance adaptive learning of deformable convolution, we refine it spatially, in terms of channel attention, and their interdependency. This results in the new version, called adaptive deformable convolution (a-dconv). We assume images exhibit linear feature map characteristics, with pixels at equal distances from the receptive field center having similar impact, indicating gradual local image content changes. To address this, we introduce adaptive dilation factors to model this effect. In Fig. 2(a) and (b), we illustrate how offset point movements can be decomposed. For example, the three middle slice points' movements are divided into $s_k \cdot p_k$ and D_{p_k}. Notably, $s_k \cdot p_k$ describes the distance from the center and remains consistent for these three points, forming a single phase as seen in Fig. 2(c). We name s_k as the phase distance, influencing the channel weight m_k. In summary, channel weights are depressed for distant sampling points and aggravated for closer ones.

With a convolutional kernel size of N which has $K = N^2$ sampling locations (e.g., 9 sampling locations in a 3×3 convolution), we let $w_k, p_k, D_{p_k} = f(D_{a_{ij}}, D_{b_{ij}})$ and D_{m_k} refer to the weight, hand-picked offset, learnable offset, and modulation scalar for the kth location. $f(D_{a_{ij}}, D_{b_{ij}})$ locates the position of sampling points in the sampling grid or spatial dimension. The refined deformable convolution can be formulated as:

$$y(p) = \sum_{k=1}^{K} w_k \cdot x(p + s_k \cdot p_k + D_{p_k}) \cdot (1 - s_k) \cdot D_{m_k},$$

where $s_k \in R^N$ is the adaptive dilation factor that contains the general distance information of sampling locations. $x(p)$ and $y(p)$ denote the feature representations at location $p = (a_{ij}, b_{ij})$ from the input feature maps x and that from output feature maps y.

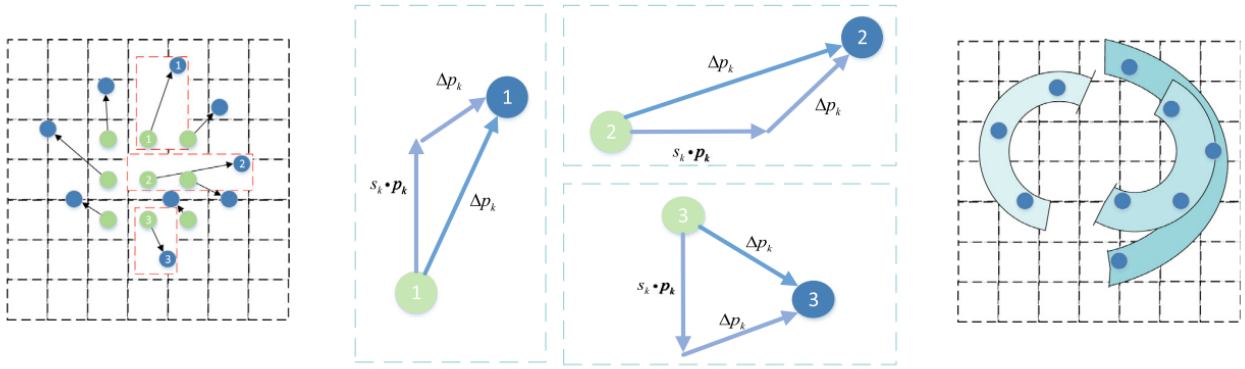
The final location in the input $(p + s_k \cdot p_k + D_{p_k}) = (a_{ij}, b_{ij})$ after adding the offset can be denoted as:

$$a_{ij} = a_{0_{ij}} + s_{ij} \cdot d_{ij} + D_{a_{ij}},$$
$$b_{ij} = b_{0_{ij}} + s_{ij} \cdot d_{ij} + D_{b_{ij}},$$

where $i \in \left\{-\frac{N}{2}, ..., \frac{N}{2}\right\}$ and $j \in \left\{-\frac{N}{2}, ..., \frac{N}{2}\right\}$ locate the integral coordinate in the kernel grid. s_{ij} is an adaptive dilation vector for the position (i, j), and d_{ij} is the predefined dilation rate for position (i, j). Since s_k and D_{p_k} are fractional, $x(p)$ is computed by bilinear interpolation.

In our Adaptive Deformable ConvNet, we use a 3×3 deformable convolution, where D_{p_k}, s_k, and D_{m_k} are defaulted to $0, 1$, and 1 respectively.

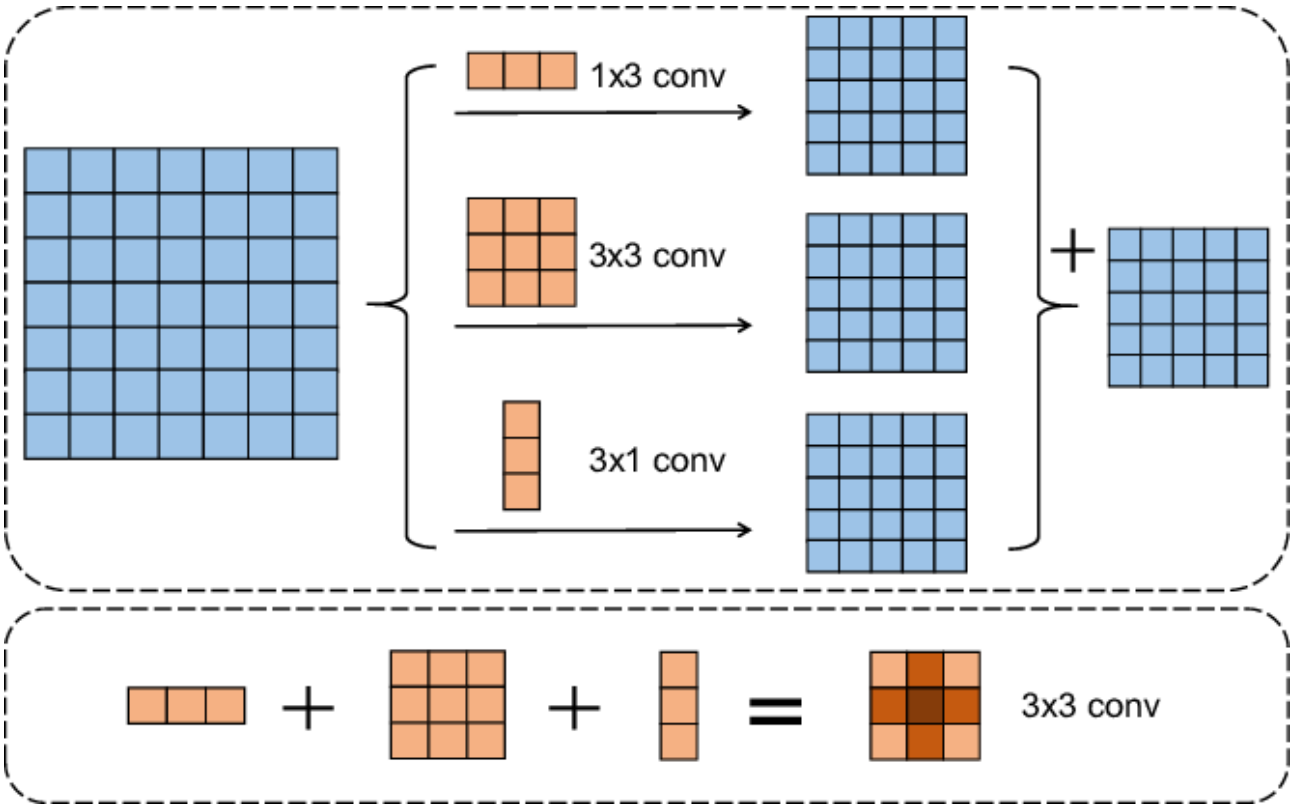
In modulated deformable convolution , the spatial attention (spatial offset) and channel attention (modulation scalar) mechanisms are independent, connected in a parallel way. In our approach, we introduce the adaptive dilation factor to interact the spatial and channel attention parts. Consequently, the modulation module becomes sensitive to the distance. The feature, spatial, and channel parts cooperatively learn relevant image content while excluding irrelevant content, as they interact with each other during the learning process.



2.6 Standard Asymmetric Convolution

Asymmetric convolution improves CNN kernel efficiency by using 1D convolutions instead of 2D squares. It breaks a square kernel into two 1D kernels (horizontal and vertical) for better spatial information, with fewer parameters and computations. For example, a 3x3 square kernel has 9 parameters, while two 1D kernels of size 3 have only 6 parameters.

ACB (Asymmetric Convolution Block) consists of two branches: one for square convolution and the other for asymmetric convolution followed by another square convolution. Their outputs are added to produce the final result. After training, ACB can revert to the original architecture by merging the two branches into a single square convolution.



The asymmetric convolution can be decomposed into two 1D convolutions, one horizontal and one vertical, as follows:

$$y_{i,j} = \sum_{m=0}^{k-1}\sum_{n=0}^{k-1} w_{m,n}x_{i+m,j+n} = \sum_{m=0}^{k-1}\left(\sum_{n=0}^{k-1} w_{m,n}\right)x_{i+m,j} + \sum_{n=0}^{k-1}\left(\sum_{m=0}^{k-1} w_{m,n}\right)x_{i,j+n} - \sum_{m=0}^{k-1}\sum_{n=0}^{k-1} w_{m,n}x_{i,j} \tag{4}$$

The output channels ratio affects the results of ACB in several ways:

1. Output channel ratios affect representation power. Higher ratios capture richer input features, while lower ratios offer reduced power.

2. Output channel ratios impact computational complexity. High ratios increase parameters and load; low ratios decrease them.

3. Output channel ratios affect generalization. High ratios risk overfitting complex data, low ratios may underfit.

The output channels ratio is vital for ACB's performance, balancing representation, complexity, and generalization. In Standard Asymmetric Convolution, square Convolution, horizontal, and vertical 1D Convolutions each get one-third of the output channels with a ratio of 0.33:0.33:0.33. Output channels for each convolution are calculated by multiplying the ratio by the total number of channels and rounding down to the nearest whole number.

2.7 Asymmetric Convolution

This convolution type, similar to the standard 2D convolution, allows for adjustable ratios (currently set to 0.5:0.33:0.33:0.17) to meet specific requirements. It enhances representation power compared to vertical and horizontal convolutions, benefiting tasks like object recognition and scene understanding.

Reducing computational complexity in square convolution yields fewer parameters and lower resource demands, resulting in quicker training, reduced memory consumption, and energy-efficient inference. However, it may compromise generalization, particularly in tasks needing localized details like edge detection or super-resolution.

2.8 Gaussian Dynamic Asymmetric Convolution

By substituting square convolution with Gaussian dynamic convolution while maintaining a 0.5:0.33:0.33:0.17 ratio for vertical and horizontal convolutions, we obtain a variant of ACB called GDAC (Gaussian Dynamic Asymmetric Convolution).

Effects of this change:

1. GDAC gains increased representation power, allowing it to learn more diverse features from input maps compared to ACB. This enhancement can boost performance in tasks like image segmentation or style transfer.

2. GDAC becomes more computationally complex, involving more parameters and operations than ACB. This may lead to longer training times, higher memory usage, and increased inference time and energy consumption.

3. GDAC's generalization ability remains unaffected, retaining the same output channel ratio as ACB. This ensures a balance between overfitting and underfitting on various tasks.

2.9 Deformable Asymmetric Convolution

Substituting square convolution with deformable convolution while maintaining the same ratio (0.5:0.33:0.33:0.17) gives rise to Deformable Asymmetric Convolution (DAC), a variant of ACB. DAC is akin to ACB but employs deformable convolution in the first branch.

The effects of this substitution are as follows:

1. DAC gains increased representation power, adapting dynamically to object geometric variations and learning more diverse features from input feature maps compared to ACB. This enhancement can boost DAC's performance in tasks demanding precise alignment, like object detection or instance segmentation.

2. DAC experiences heightened computational complexity due to more parameters and operations compared to ACB. This may result in longer training times, increased memory consumption, and longer inference times with higher energy consumption for DAC.

3. DAC's generalization ability remains unchanged, as it maintains the same output channel ratio as ACB, striking a balance between overfitting and underfitting across various tasks.

2.10 Adaptive Deformable Asymmetric Convolution

By replacing square convolution with Adaptive Deformable Asymmetric Convolution (ADAC) while maintaining the same ratio (0.5:0.33:0.33:0.17), we create a distinct version of ACB, referred to as ADAC.

ADAC is similar to ACB but uses Adaptive Deformable Asymmetric Convolution instead of standard 2D convolution for the first branch.

The effects of this change are as follows:

1. ADAC increases representation power by adapting to geometric object variations and capturing finer, asymmetric features in input feature maps, potentially improving performance in tasks requiring precise alignment, handling complex information, such as object detection, instance segmentation, or style transfer.

2. ADAC introduces higher computational complexity due to more parameters and operations compared to ACB, resulting in longer training times, increased memory consumption, and higher inference time and energy usage.

3. ADAC maintains the same output channel ratio as ACB, preserving its generalization ability across various tasks, balancing between overfitting and underfitting.

3 Image Segmentation Architectures

3.1 UNet

U-Net, initially designed for biomedical image segmentation, consists of two main parts: a contracting path with convolutional layers, ReLU activation, and max pooling, and an expansive path with upsampling, convolution, and skip connections through concatenation. Its final 1x1 convolution layer maps features to classes.

U-Net is highly effective in various semantic segmentation tasks, especially in medical image processing, denoising, super-resolution, and image generation. Its success has led to the development of variants like UNet++, ResUNet, and Attention U-Net, making it a key architecture in deep learning for image segmentation.

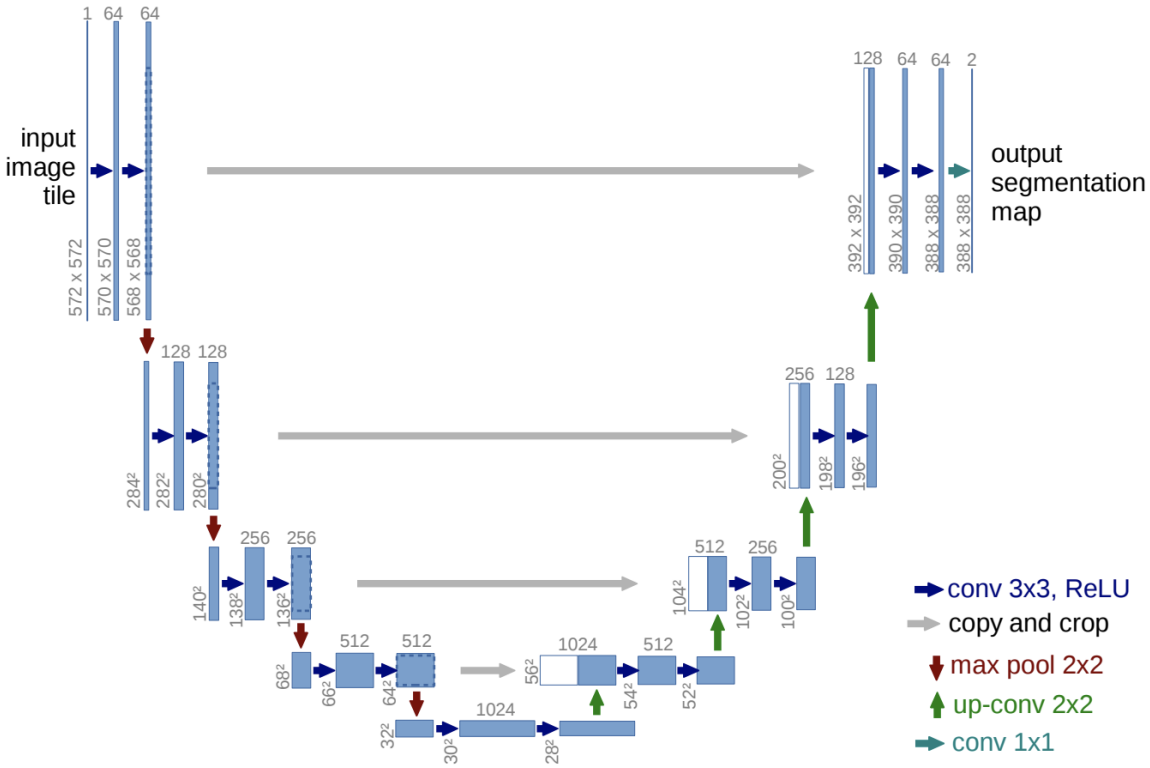
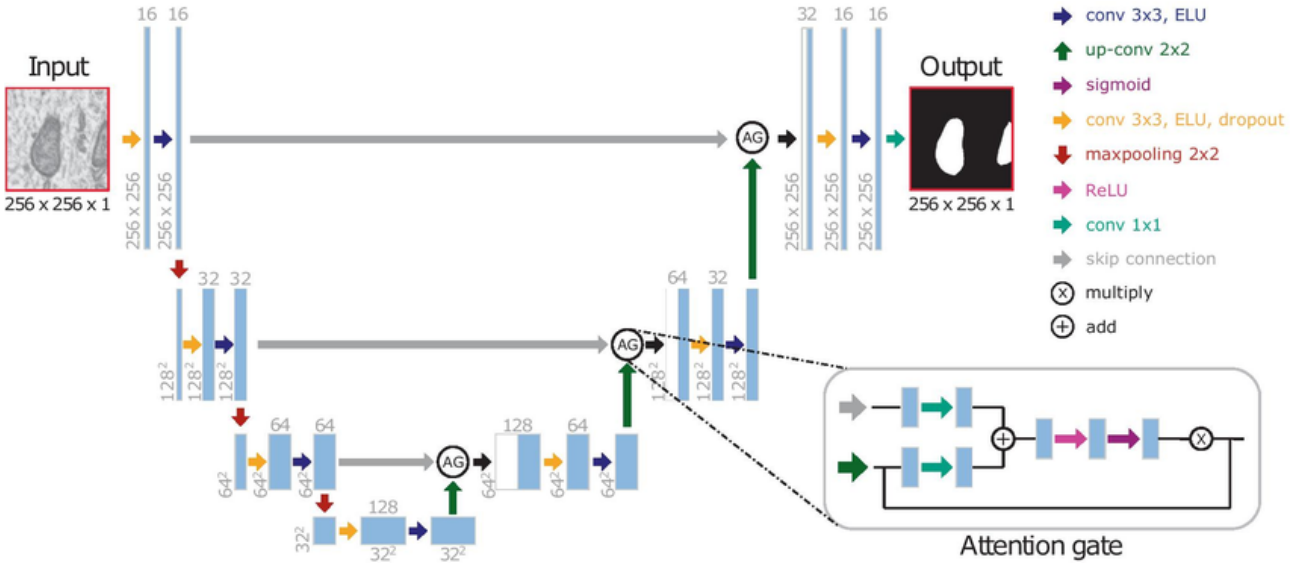


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

3.2 Attention-Unet

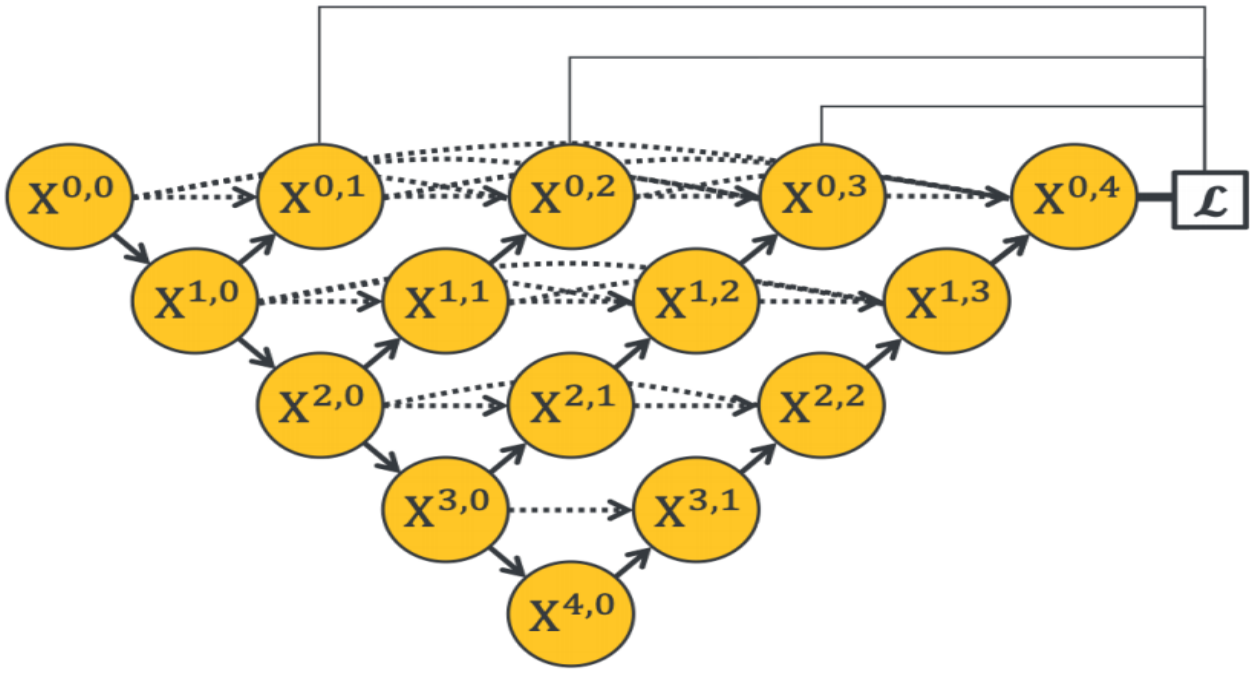
Attention U-Net is a neural network model for semantic segmentation, widely used in medical imaging, autonomous driving, and scene analysis. It extends the U-Net architecture by incorporating attention gates into skip connections. These attention gates dynamically focus on target structures and suppress irrelevant regions in input images, improving segmentation

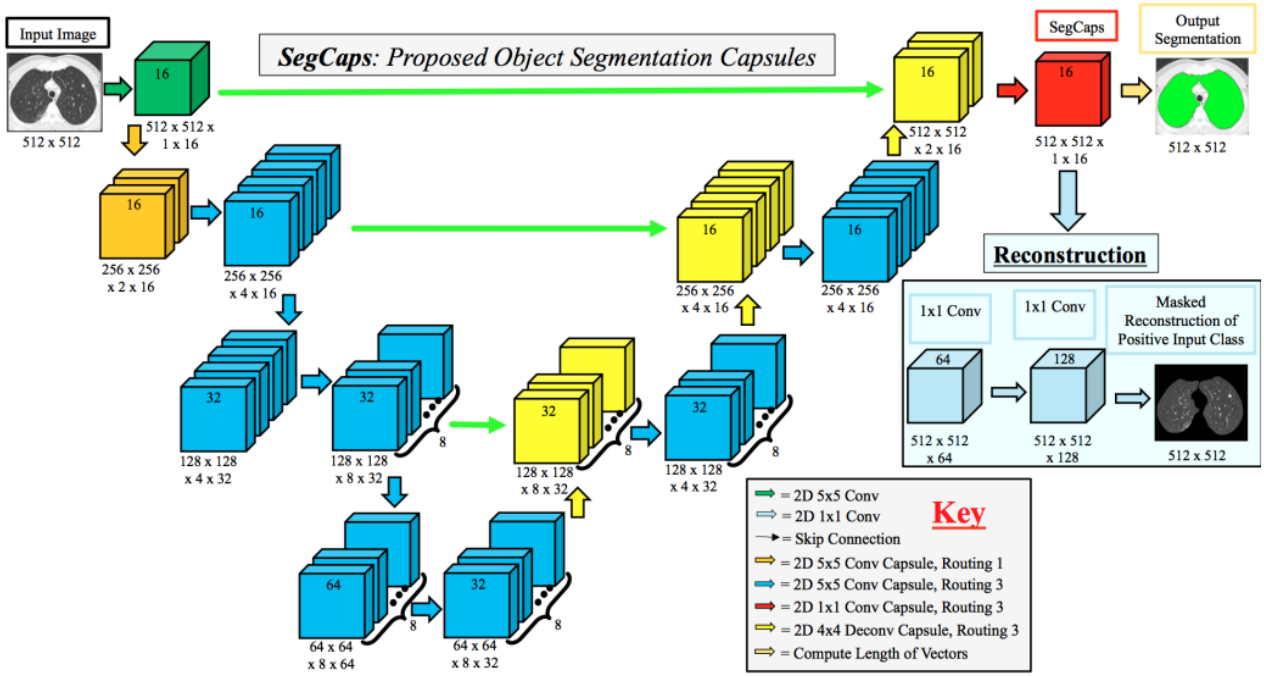
precision without additional modules or networks. This adaptive attention makes it valuable for capturing intricate patterns in segmentation tasks.



3.3 Unet++

UNet++ enhances semantic segmentation using dense convolutional blocks and deep supervision within skip connections. It improves upon the U-Net model by strategically placing dense convolutional blocks to boost feature processing and introducing deep supervision with auxiliary outputs in the expansive path. These innovations optimize learning, improving gradient flow and training stability. This is crucial in applications like medical image analysis, autonomous driving, and scene understanding.





3.4 Capsule-Net

CapsuleNet, a novel neural network, utilizes capsules to encode entity presence and properties in images. It addresses CNN limitations: loss of spatial data, data dependency, and lack of viewpoint generalization. Comprising three key components – convolutional layer (low-level feature extraction), primary capsule layer (transforms features into pose capsules), and digit capsule layer (groups capsules using dynamic routing) – CapsuleNet excels in image recognition. It employs a reconstruction network to prevent overfitting, achieving top results on MNIST, smallNORB, and CIFAR10 datasets, while demonstrating robustness to transformations and adversarial attacks. A promising approach for capturing hierarchical visual relationships.

4 Conclusion

The results of this study reveal the profound impact of convolutional technique selection on the performance of semantic segmentation architectures, including Unet, Attention Unet, Unet++, and Capsule-Net. Notably, Unet with Gaussian dynamic convolution consistently demonstrated superior accuracy across various image processing tasks. Attention Unet and Unet++, on the other hand, showcased their strengths when combined with specific convolution types. Attention Unet, when paired with adaptive deformable convolution, exhibited remarkable efficiency in object detection tasks. Unet++ paired with standard asymmetric convolution demonstrated robust performance in terms of both accuracy and speed. These findings underscore the significance of tailoring convolutional techniques to the unique characteristics of different architectures. Moreover, the research illuminates the intricate synergy between convolutional methods and architectural designs, offering valuable guidance for optimizing neural networks in computer vision applications.