Navigating the InternImage Model: A Deep Dive into its Encoder and Revolutionary DCNv3 Operator

By Andres G. Gomez

I. Article Purpose

I have written this article to dissect the intricacies of the InternImage model's encoder and core operator, DCNv3. Upon delving into the model's intricacies, I found that the paper lacked crucial details. Online resources explaining their method or the nuances of this model were also scarce. To bridge this gap, I delved into a thorough exploration of the code and engaged in numerous journal club presentations, along with discussions within my lab. Drawing from these experiences, I've compiled this explanation and tutorials, aiming to provide clarity and aid fellow enthusiasts in grasping the model's complexities. While not intended as a standalone resource, this article's purpose is to complement and bolster the explanations provided by the authors and their paper.

II. InternImage

The authors set out with a lofty ambition: to construct a unified model capable of accommodating various input data modules and combinations, and whose output can support any task and its combination [1]. The authors of InternImage argue that CNN-based models can rival or even outperform Vision Transformers (ViTs). Their work lays the foundation for a CNN-based model that seamlessly scales to accommodate large-scale parameters and data efficiently. Additionally, they introduce a groundbreaking operator, Deformable-Convolution-v3 (DCNv3), which adds a new dimension to the model's capabilities.

III. Encoder Architecture

To provide a comprehensive understanding of the InternImage core operator, it's beneficial to begin by examining the architecture of InternImage itself. As illustrated in *Figure 1*, we initiate the process by feeding an image of dimensions HxWx3 into the stem layer. This initial step yields a feature map characterized by C_1 channels and reduced dimensions of H/4 by W/4. Subsequently, this feature map is propagated through multiple InternImage blocks, conveniently categorized into stages 1 through 4.

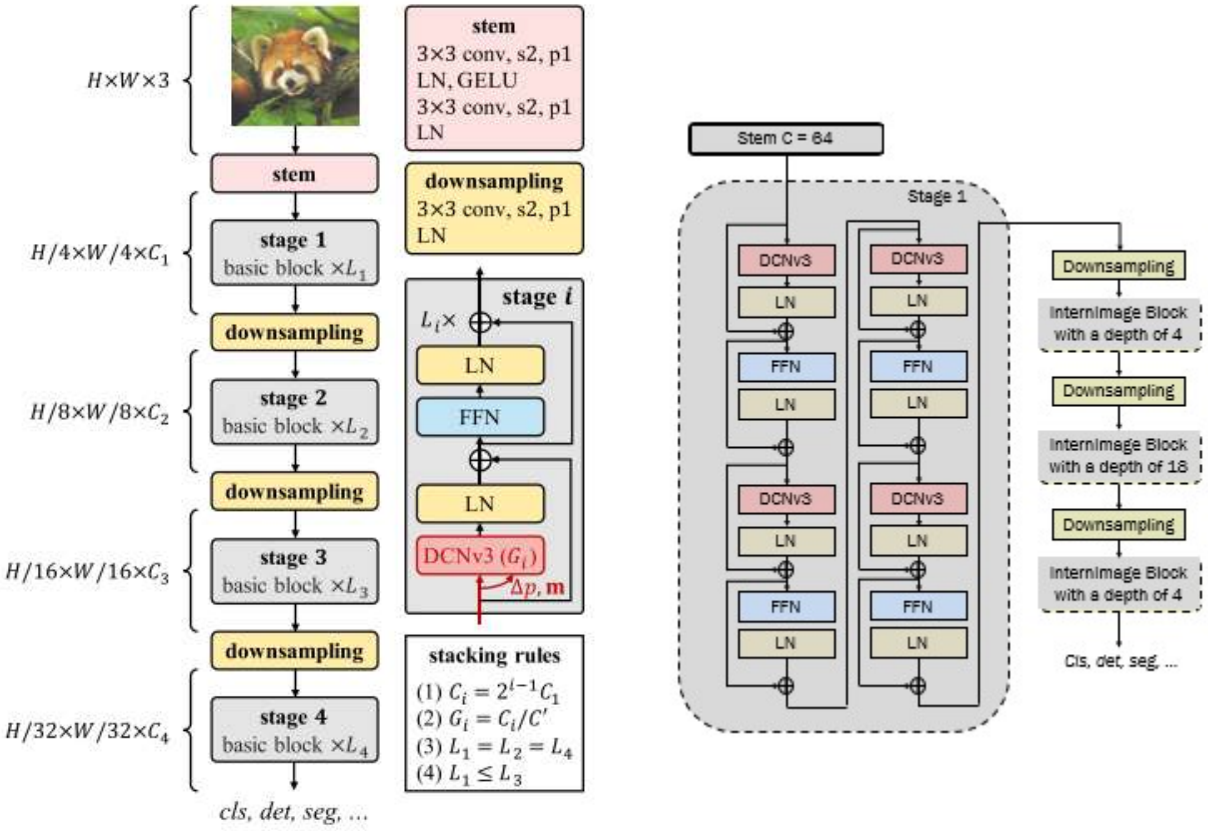

Figure 1: Overall architecture of the InternImage encoder. The core operator is DCNv3, and the basic block composes of layer normalization (LN) and feed-forward network (FFN), the stem and downsampling layers follows conventional CNN's designs, where "s2" and "p1" mean stride 2 and padding 1, respectively. Constrained by the stacking rules, only 4 hyperparameters (C1, C0 , L1, L3) can decide a model variant [1]. The right figure depicts the depth of the basic block.

In *Figure 1*, the deformable convolution operation is represented as DCNv3. Within each stage of the basic block, the input matrix undergoes convolution using this core operator. Subsequently, the process includes layer normalization, followed by summation with a skip connection, a feed-forward network, another layer normalization step, and finally summation with another skip connection. These operations can be iterated multiple times within a single stage, effectively increasing the number of parameters per stage. This iterative process enables the model to learn progressively intricate and abstract features.

The right image in *Figure 1* above illustrates the stacking and depth of these operations, which are regulated by parameter L_i. The depths of these stages and the model sizes are determined by the guidance provided by Tan et al [2]. While the default depths for each stage are set at [4, 4, 18, 4], they may vary depending on the parameter size of the model.

IV. DCNv3 core operator

The basic idea behind the DCNv3 core operator is to introduce learnable offsets to the regular grid sampling locations used in standard convolutional operations. These offsets effectively enhance the flexibility of the convolution operations, allowing them to adaptively adjust their receptive field to better capture spatially variant features in the input data. *Figure 2* illustrates the deformable convolution process, showcasing how the introduced offsets enable dynamic adjustments to the convolutional receptive field.

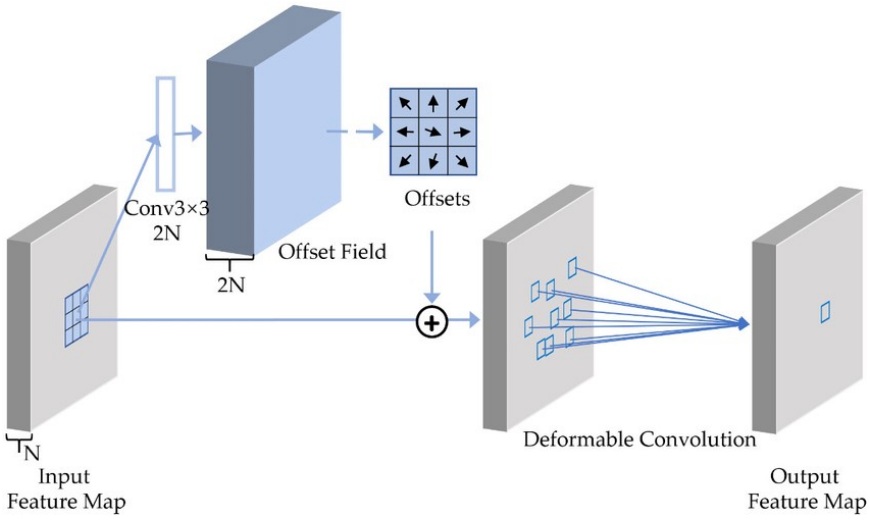

Figure 2: Deformable convolution principle, provided by Wang et al [3] .

The overall architecture behind the DCNv3 core operator is illustrated in *Figure 3*. The input feature map *x* is inputted into the operator. Depthwise convolution, normalization, and activation are subsequently applied to *x*. This process is followed by linear layers tasked with learning offset parameters Δp_{gk} and modulation scalars (filter values) m_{gk}. Here, *g* denotes the group to which the parameter belongs, and k corresponds to the k-th sampling point in the convolution filter. The utilization of groups facilitates the learning of distinct spatial aggregation patterns, thereby enhancing the robustness of features for downstream tasks.

The input feature map *x* is then linearly projected, resulting in *x'*. Utilizing the learned sampling offsets, modulation scalars, and the projected input, we perform deformable convolution and layer normalization to obtain an output feature map *y*. Subsequently, we have the option to adaptively combine this output with the linearly projected input or proceed directly with *y*. The last step involves linearly projecting the output.

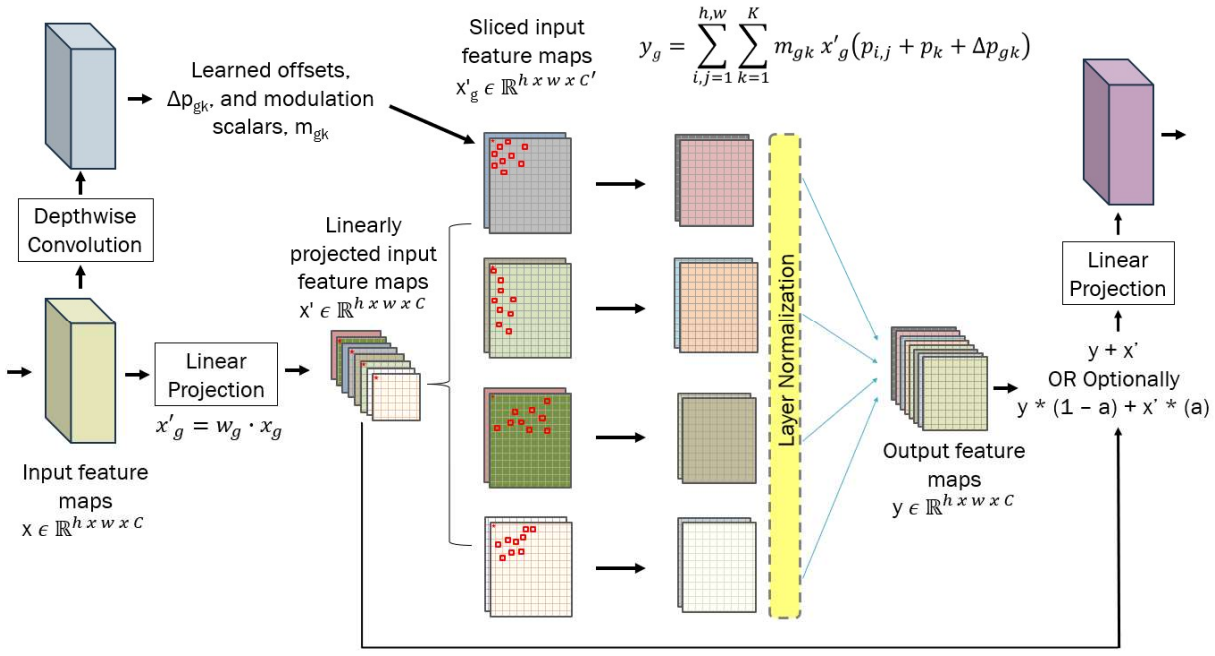

Figure 3: DCNv3 core operator. The input feature map and linearly projected input feature map are denoted by x and x', respectively. The location irrelevant projection weights of each group are denoted by w_g, m_{gk} denotes the modulation scalar of the k-th sampling point, Δp_{gk} denotes the offsets, and y_g denotes the output feature map of group g.

The paper offers the following equation for DCNv3, where G represents the total number of aggregation groups. For the g-th group, $w_g \in R^{C \times C'}$ denotes the location irrelevant projection weights of the group, where C' =C/G represents the group dimension. $m_{gk} \in R$ denotes the modulation scalar of the k-th sampling point in the g-th group, normalized by the SoftMax function along the dimension K. $x_g \in R^{C' \times H \times W}$ represents the sliced input feature map. Δp_{gk} is the offset corresponding to the grid sampling location p_k in the g-th group [1].

$$y(p_0) = \sum_{g=1}^{G} \sum_{k=1}^{K} w_g\, m_{gk}\, x_g(p_0 + p_k + \Delta p_{gk})$$

For clarity, we suggest some modifications to the above equation.

$$y(p_{i,j}) = \sum_{g=1}^{G} \sum_{k=1}^{K} m_{gk}(i,j)\, x'_g\left(p_{i,j} + p_k + \Delta p_{gk}(i,j)\right)$$

$$x'_g = w_g \cdot x_g$$

In this equation, it's important to note that for each point (i, j), there exists a corresponding location-aware modulation scalar m_{gk} and offset Δp_{gk}. Additionally, x_g' represents a linear projection of the input x_g. This notation not only clarifies the mathematical operations but also facilitates understanding when referring to the corresponding code.

V. Conclusion

By addressing gaps in existing documentation and drawing from my own analysis, I've aimed to provide a thorough grasp of InternImage, enhanced with tutorials and explanations intended to assist both enthusiasts and practitioners. While this article serves to complement the work of the original authors and their paper, it also reflects the collaborative efforts within the research community to propel advancements in deep learning. As the pursuit of knowledge continues, I remain committed to unraveling the complexities of emerging models and fostering a deeper understanding of their capabilities.

For an in-depth explanation of the author's DCNv3 code, please refer to my tutorial, which meticulously walks through nearly every line of code.

VI. References

[1] W. Wang et al., "Internimage: Exploring large-scale vision foundation models with deformable convolutions," 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2023. doi:10.1109/cvpr52729.2023.01385

[2] M. Tan, and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," International Conference on Machine Learning, 2019. doi:10.48550/arXiv.1905.11946

[3] S. Wang, X. Xia, L. Ye, and B. Yang, "Automatic detection and classification of steel surface defect using deep convolutional neural networks," Metals, vol. 11, no. 3, p. 388, Feb. 2021. doi:10.3390/met11030388