

# Vision Transformers

Docentes:

Esp. Abraham Rodriguez - FIUBA

Mg. Oksana Bokhonok - FIUBA

# Programa de la materia

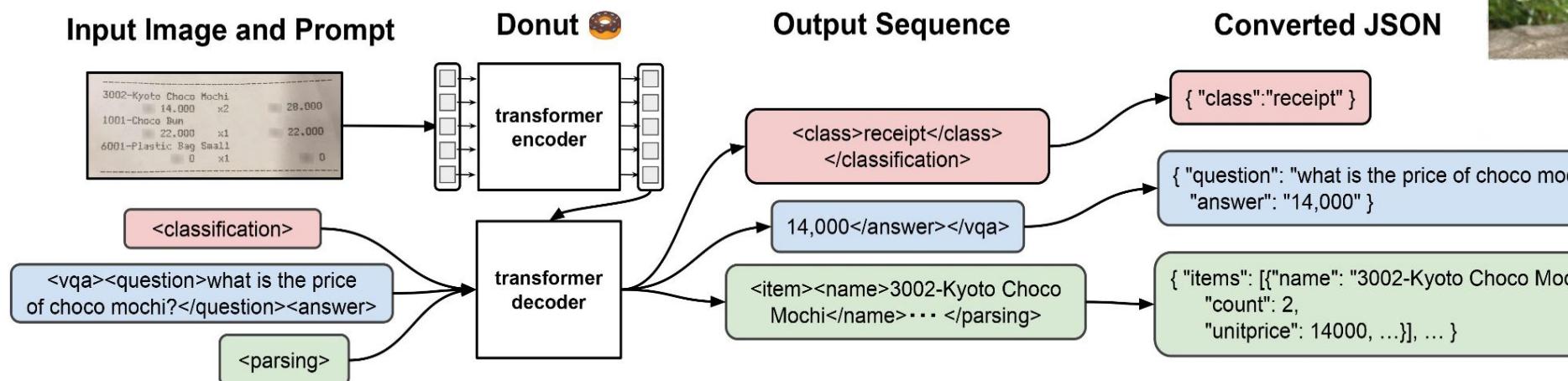
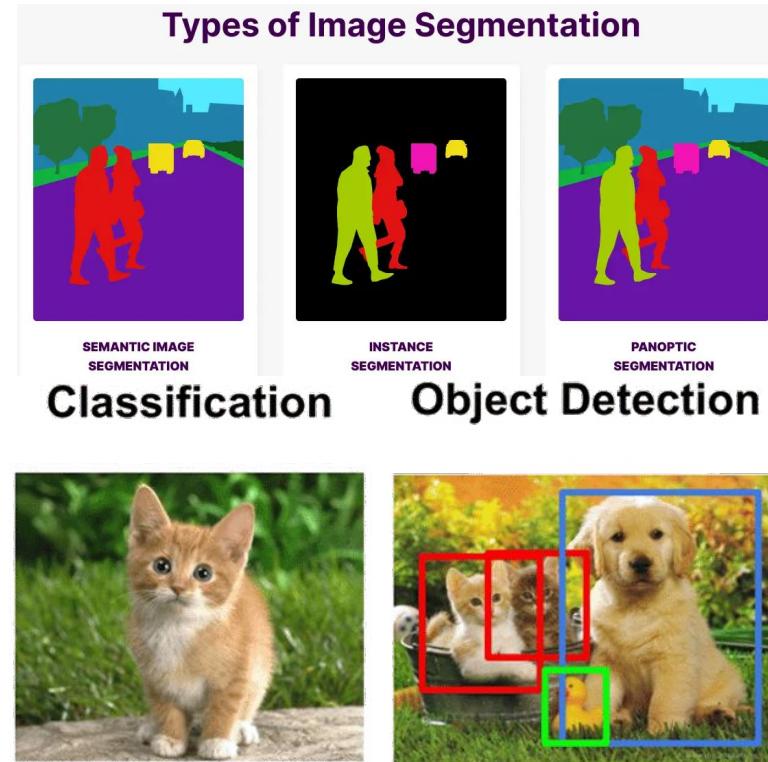
1. Arquitectura de Transformers e imágenes como secuencias.
2. **Arquitecturas de ViT y el mecanismo de Attention.**
3. Ecosistema actual, Huggingface y modelos pre entrenados.
4. GPT en NLP e ImageGPT.
5. Modelos multimodales: combinación de visión y lenguaje
6. Segmentación con SAM y herramientas de auto etiquetado multimodales.
7. OCR y detección con modelos multimodales.
8. Presentación de proyectos.

# ViT vs CNN: Comparativa y beneficios de enfoques híbridos (ViT+CNN)

Característica	ViT	CNN (Convolutional Neural Networks)	Híbrido ViT+CNN
<b>Estructura</b>	Basado en <b>attention</b> . Sin operación de convolución.	Basado en <b>convoluciones</b> para la extracción de características.	Combinación de <b>convolución</b> y <b>self-attention</b> .
<b>Procesamiento de datos</b>	Procesa la imagen como una <b>secuencia</b> de parches.	Procesa la imagen usando <b>filtros</b> convolucionales.	Usa CNN para características <b>locales</b> y ViT para relaciones <b>globales</b> .
<b>Ventajas</b>	<ul style="list-style-type: none"> <li>Captura relaciones <b>globales</b> entre píxeles.</li> <li><b>Escalabilidad</b> con datos grandes.</li> </ul>	Excelente para capturar características <b>locales</b> .	Captura características <b>locales</b> y <b>globales</b> .
<b>Desventajas</b>	Necesita una alta cantidad <b>más datos</b> para entrenar.	Dificultad para capturar relaciones globales a gran escala.	Mayor complejidad y demanda computacional.
<b>Necesidad de datos</b>		<b>Menor cantidad de datos</b> en comparación con los ViT.	Depende del balance entre la parte CNN y la parte ViT.
<b>Ejemplos</b>	Diagnóstico médico como detección de patologías en una radiografía.	Reconocimiento facial, donde se debe identificar características y detalles faciales.	Vehículos autónomos para identificar y clasificar objetos en tiempo real.

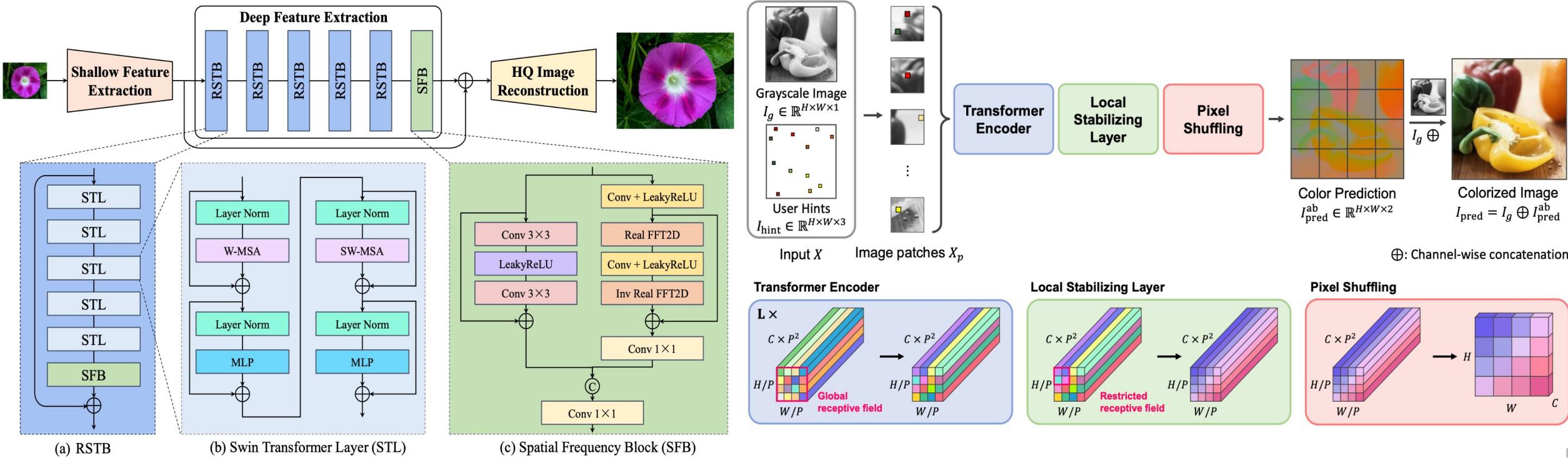
# Aplicaciones de ViT: Casos de Uso

Aplicación	Ejemplos de Uso
<b>Tareas de Reconocimiento</b>	<ul style="list-style-type: none"> <li><b>Clasificación de imágenes:</b> Categorizar imágenes (e.g., especies de animales).</li> <li><b>Detección de objetos:</b> Identificación de objetos en imágenes.</li> <li><b>Segmentación:</b> Dividir imágenes en segmentos significativos.</li> </ul>
<b>Tareas Multimodales</b>	<ul style="list-style-type: none"> <li><b>Respuesta a preguntas visuales:</b> Responder preguntas sobre el contenido de una imagen.</li> <li><b>Razonamiento visual:</b> Realizar inferencias basadas en imágenes.</li> <li><b>Visual grounding:</b> Asociar texto con regiones específicas de una imagen.</li> </ul>

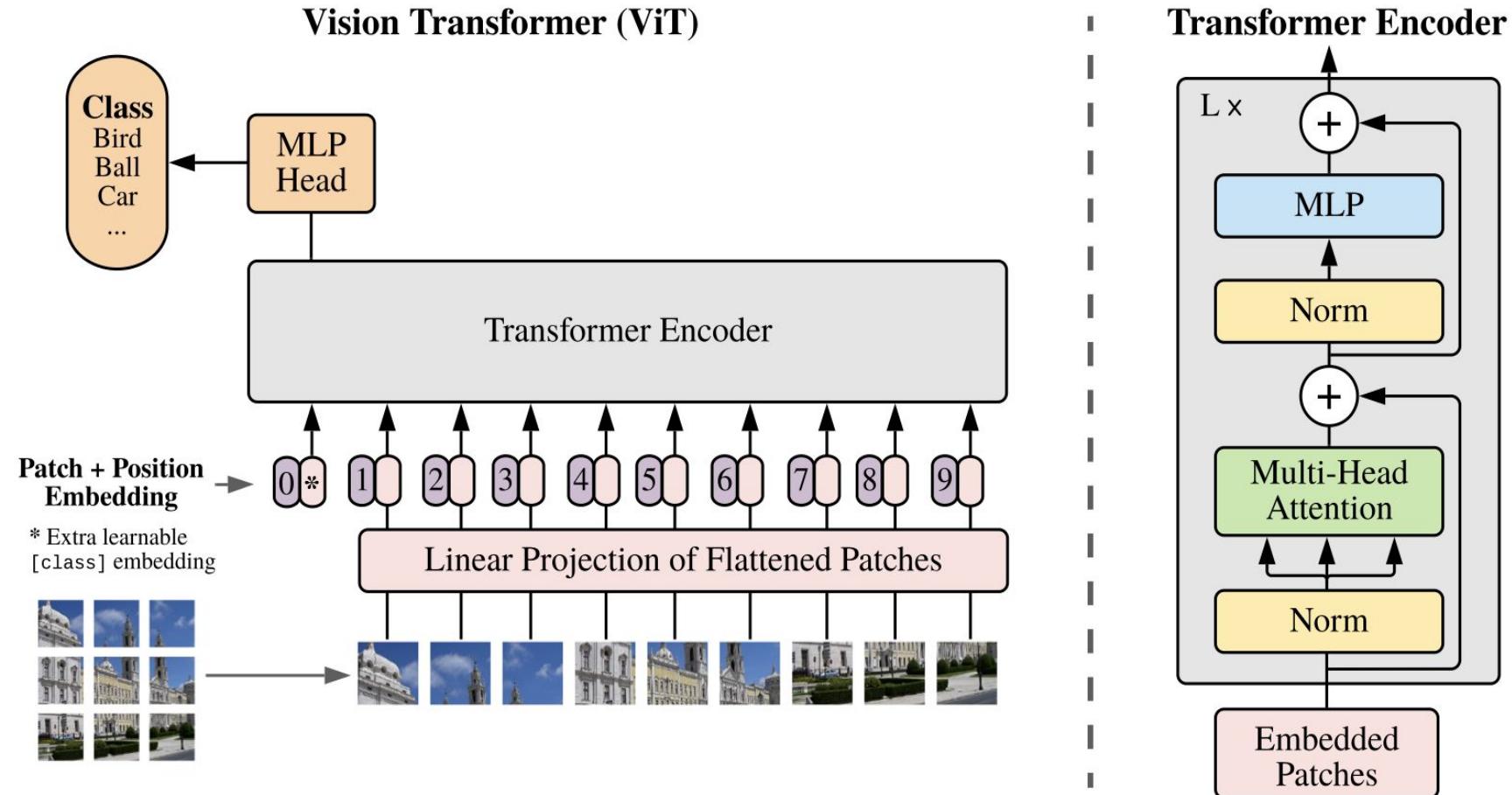


# Aplicaciones de ViT: Casos de Uso

Aplicación	Ejemplos de Uso
<b>Procesamiento de Video</b>	<ul style="list-style-type: none"> <li><b>Reconocimiento de actividades:</b> Identificar acciones en secuencias de video.</li> <li><b>Pronóstico de video:</b> Predecir frames futuros en un video.</li> </ul>
<b>Visión de Bajo Nivel</b>	<ul style="list-style-type: none"> <li><b>Súper-resolución de imágenes:</b> Mejorar la resolución de imágenes borrosas.</li> <li><b>Mejora de imágenes:</b> Aumentar la calidad visual.</li> <li><b>Colorización:</b> Asignar colores a imágenes en blanco y negro.</li> </ul>
<b>Análisis 3D</b>	<ul style="list-style-type: none"> <li><b>Clasificación y segmentación de nubes de puntos:</b> Clasificación y segmentación de datos 3D.</li> </ul>



# Arquitectura Base ViT



Acceso a un GPT personalizado para la cátedra:  
<https://chatgpt.com/g/g-6817f513823c8191b40515f49e3bda20-vit-ayudante>

# Dimensiones

Entrada Imagen con dimensiones  $x \in \mathbb{R}^{H \times W \times C}$ , donde H, W y C representan la altura, el ancho y el número de canales

División en Patches:

- Tamaño del patch:  $P \times P$
- Número de patches:  $N = HW/P^2 \quad N \in \mathbb{Z}$
- Embedding de cada patch:  $D = CP^2$

**ViT-B/16:** D=768    **ViT-L/16:** D=1024    **ViT-H/14:** D=1280

Positional Encoding:

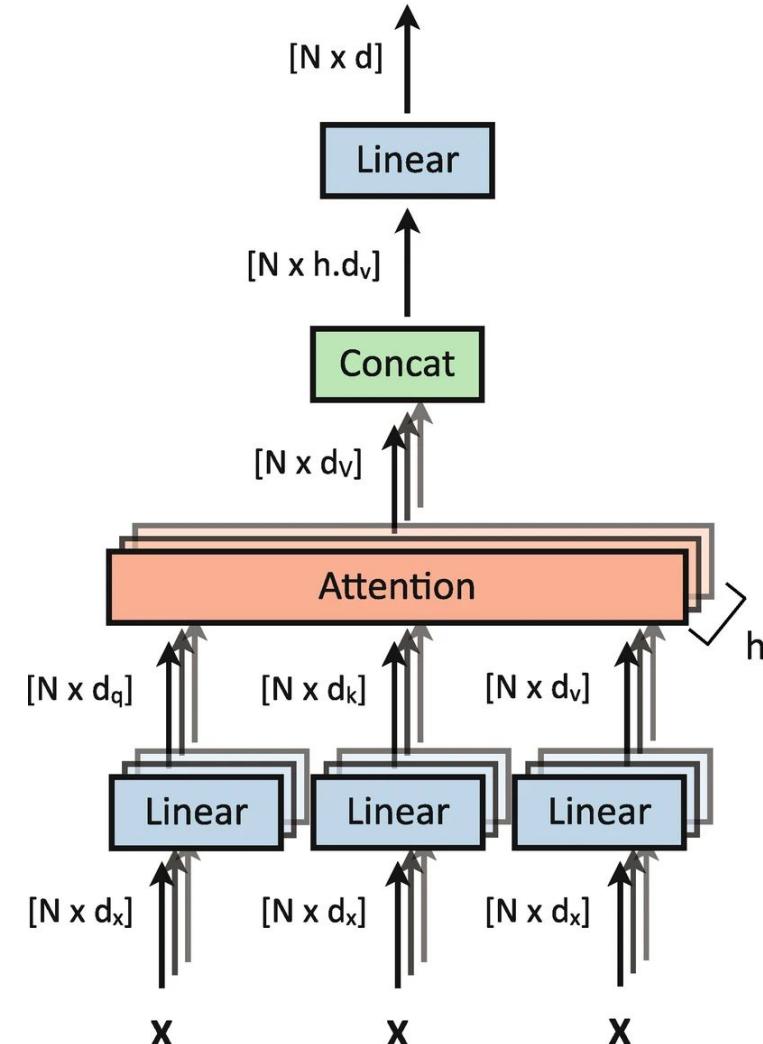
- Es añadido a cada patch embedding y no cambia la dimensión.

**Token [CLS]:**

- Representación global de la imagen
- Agregado como primer token:  $(N+1)D$

Dimensión del Modelo

- Sin [CLS] y Positional Encoding:  $N \times D$
- Con [CLS] agregado:  $(N+1) \times D$



Recomendable ver el ejemplo de  
[Transformers-for-NLP-and-Computer-Vision-3rd-Edition](#)

# Caso particular: clasificación

Método	Implementación
<b>Global Average Pooling (GAP)</b>	Simplicidad, eficiencia computacional
<b>Multihead Attention Pooling (MAP)</b>	Utiliza un mecanismo de atención multi-cabeza para ponderar y combinar las representaciones de los tokens de imagen, generando una representación global más informada.
<b>Classification Token [cls]</b>	A lo largo de las capas, el token [CLS] acumula información de todos los demás tokens mediante el mecanismo de atención.

# Caso particular: clasificación

Aspecto	Token [CLS]	GAP	MAP
Pérdida de Información Local	Alta: El token único puede no capturar detalles finos	Alta: El promedio diluye características distintivas	Media: La atención puede ser inconsistente
Complejidad Computacional	Baja: Un solo token a procesar	Baja: Solo un promedio sobre la dimensión	Alta: Cálculo de atención multi-cabeza
Sensibilidad a la Distribución	Media: Depende del entrenamiento del token	Alta: No captura distribuciones heterogéneas	Baja: Captura relaciones ponderadas
Escalabilidad	Alta: Aumenta linealmente con el número de parches	Alta: Mantiene eficiencia incluso en imágenes grandes	Baja: Costoso en imágenes de alta resolución
Interpretabilidad	Baja: Difícil saber qué representa el valor final	Baja: No muestra qué tokens son más relevantes	Media: Se pueden visualizar los pesos
Riesgo de Sobreajuste	Medio: Depende del contexto y calidad de datos	Bajo: Generaliza bien al ser un promedio	Alto: Puede memorizar patrones específicos
Impacto en Datos Escasos	Alto: Puede no aprender bien con pocos ejemplos	Bajo: Generaliza mejor al evitar dependencias puntuales	Alto: Tiende a memorizar en conjuntos pequeños
Dependencia de la Configuración	Baja: Generalmente funciona sin ajustes finos	Baja: Promedio fijo, sin hiperparámetros complejos	Alta: Cabezas y dimensiones influyen mucho

# The Annotated Transformer

[The annotated Transformer](#) es una de las primeras guías de implementación y explicativas acerca del Transformer.

**Recomendable leer!**

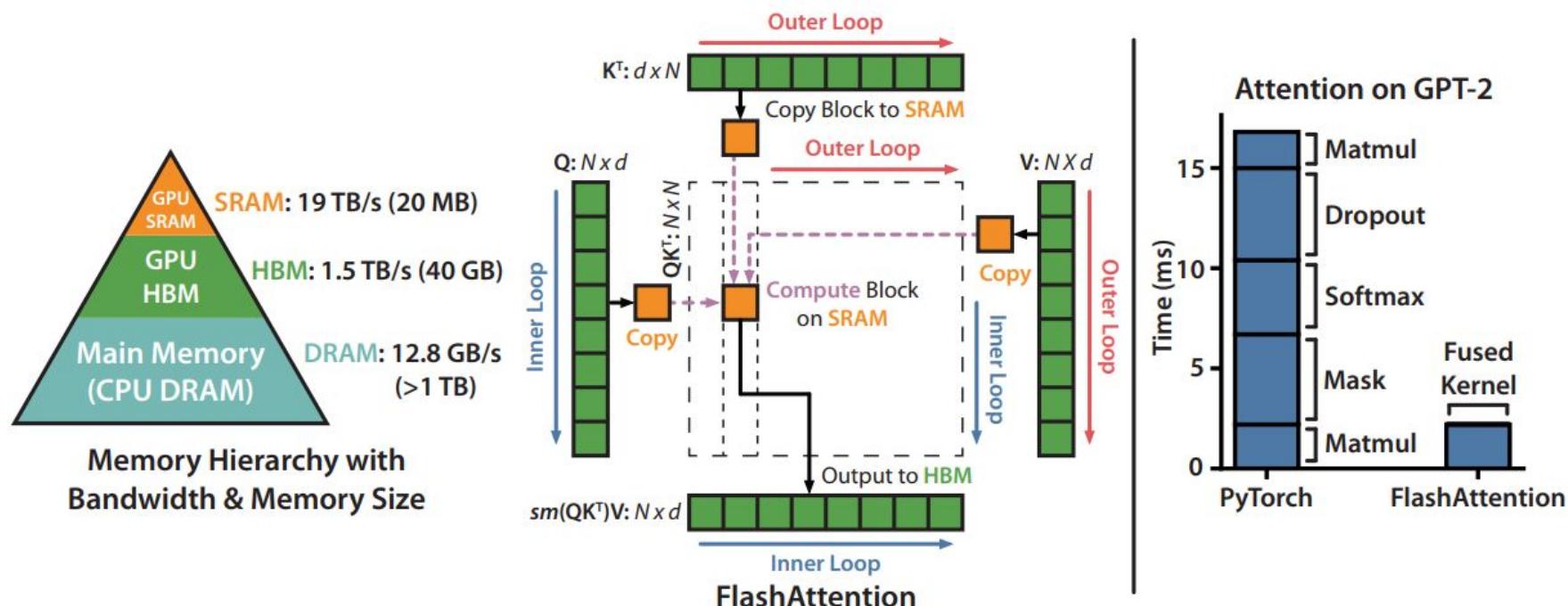
# Algunas variantes de Attention

- [Flash Attention](#) (estándar)
- [Window Multi-head Self Attention \(Swin Transformer\)](#)
- [Cascaded Group Attention](#)
- [Linear Self-Attention](#)

# FlashAttention

Hasta el momento se han presentado diversas propuestas de mejora de eficiencia para ViT. Sin embargo el estándar actual para modelos Transformer (NLP, ViT, etc) es FlashAttention.

- FlashAttention se presentó en el paper "[FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness](#)" en NeurIPS 2022.
- Optimiza la eficiencia de la atención en Transformers.
- Usa un enfoque IO-aware para reducir lecturas y escrituras en GPU.
- Aumenta la velocidad en modelos como GPT-2 y mejora el rendimiento en tareas de secuencias largas



# **Arquitecturas ViT**

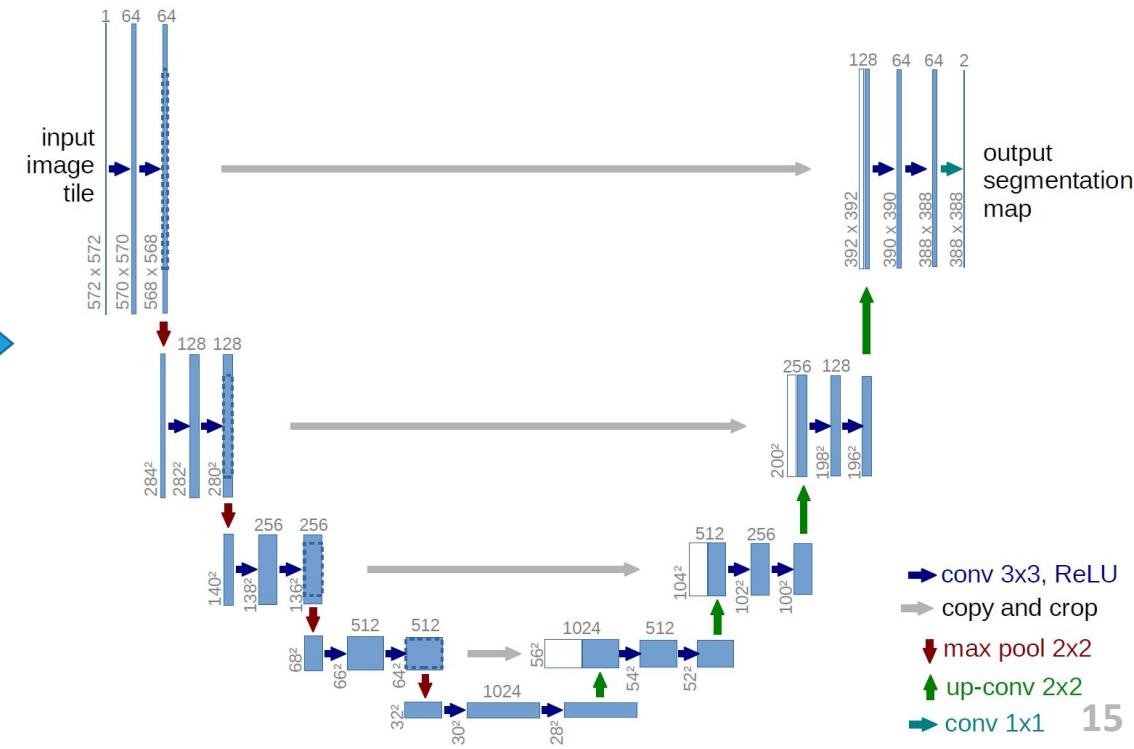
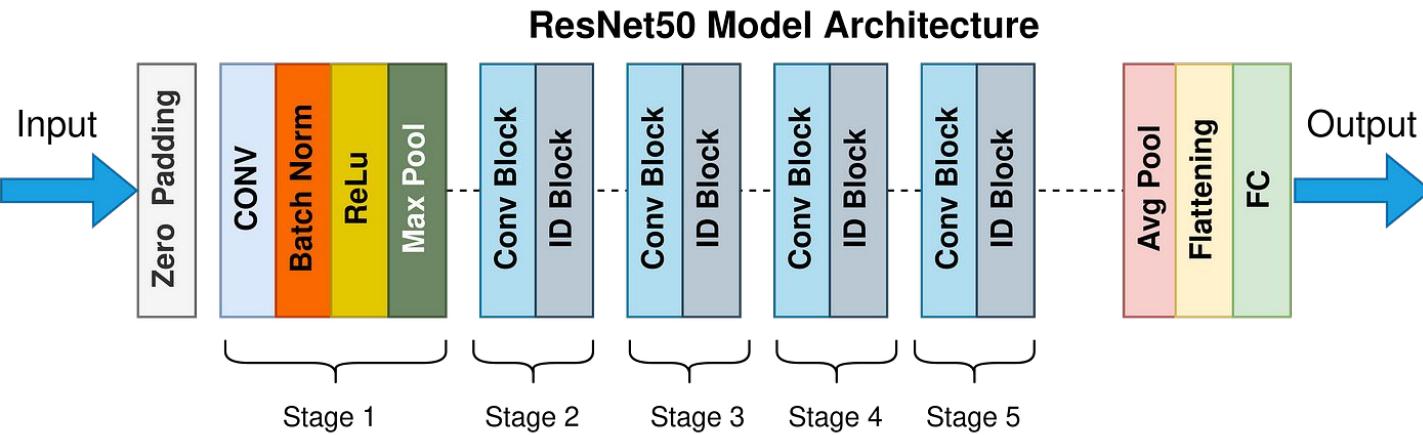
# Algunas arquitecturas ViT

- **Swin Transformer**
- Convolutional Vision Transformer (CvT)
- MobileViT
- Pyramid ViT

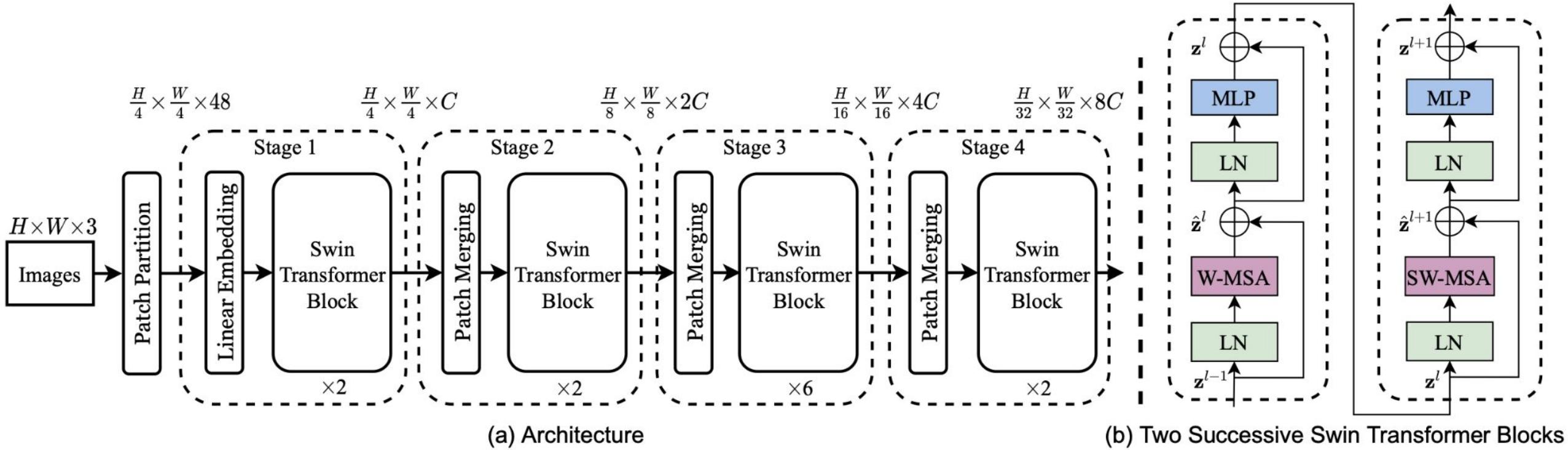
# Transformers Jerárquicos

Muchas arquitecturas ViT están diseñadas **jerárquicamente** similar a CNNs como ResNet.

La razón principal es debido a la extracción de características profunda que tienen las CNNs la cual se trata de replicar con Transformers.

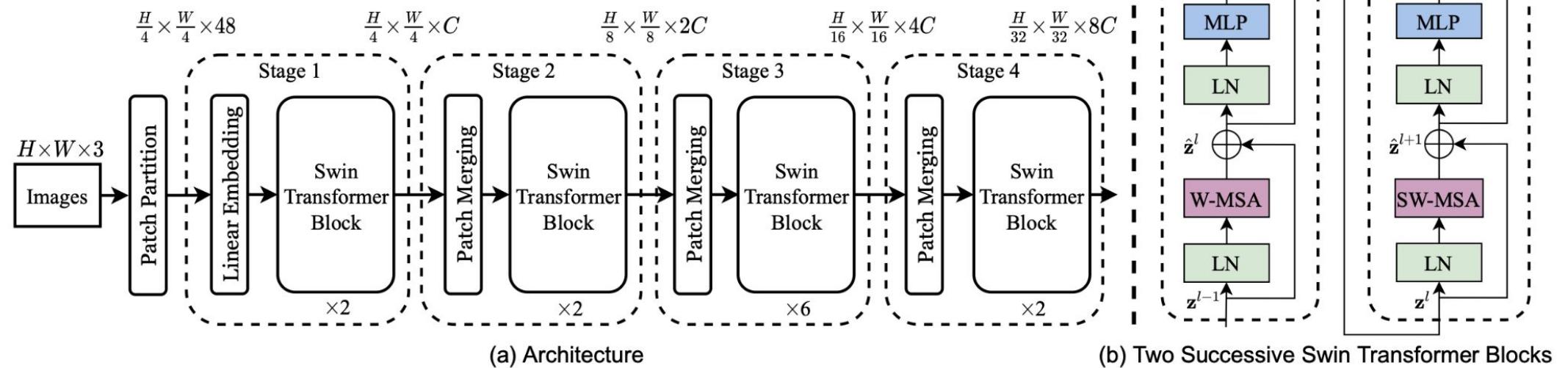


# Swin Transformer



- El Swin Transformer se presentó en ICCV en 2021. por Ze Liu et.al., en "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows" [Link-paper](#), [Link-huggingface](#)
- Mecanismo de atención **jerárquica basado en ventanas deslizantes**.
- W-MSA divide la imagen en ventanas no superpuestas para aplicar self-attention local.
- SW-MSA usa ventanas deslizantes para capturar información a mayor escala.
- Más eficiente en memoria y mejor en la captura de detalles locales y globales que ViT clásico.

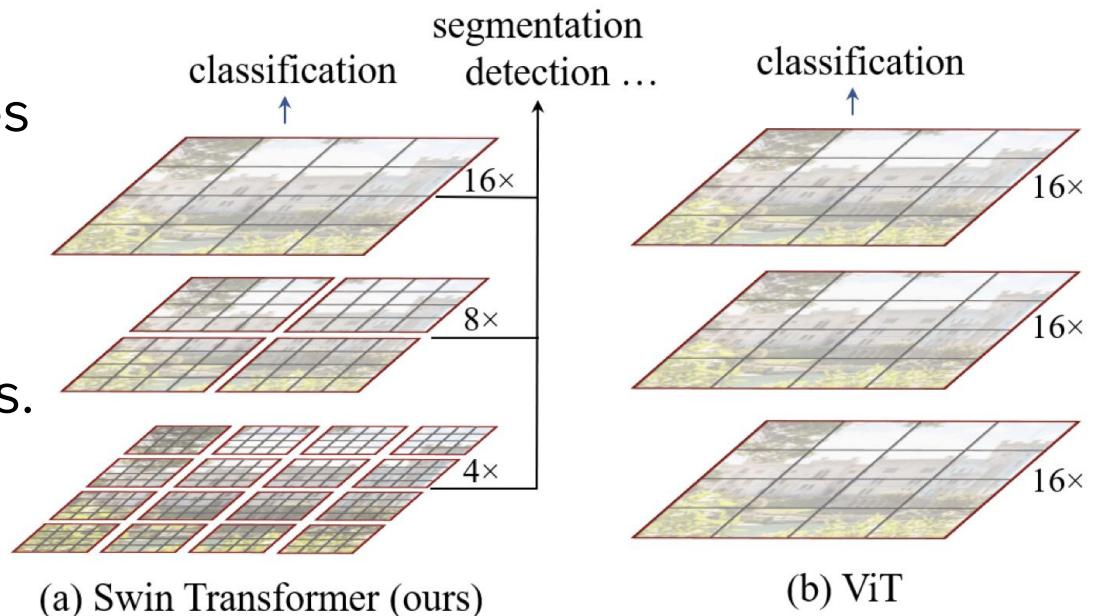
# Swin Transformer



Se realizan subdivisiones dentro de cada parche, aplicando **attention** y permitiendo aprender features en el **orden jerárquico**.

La jerarquía hace posible que funcione como backbone de propósito general para distintas tareas.

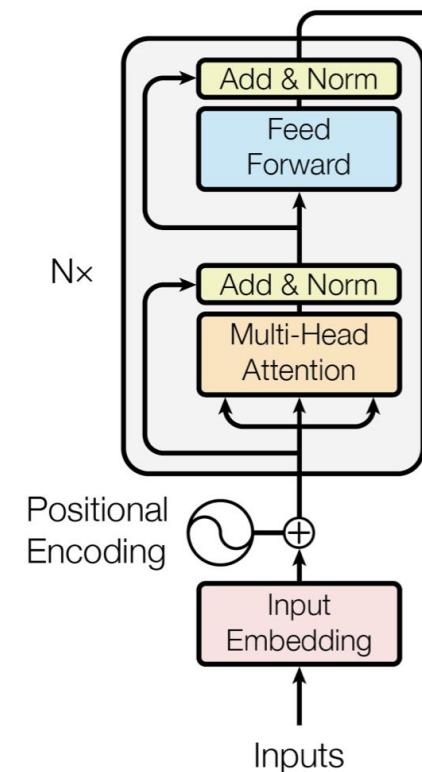
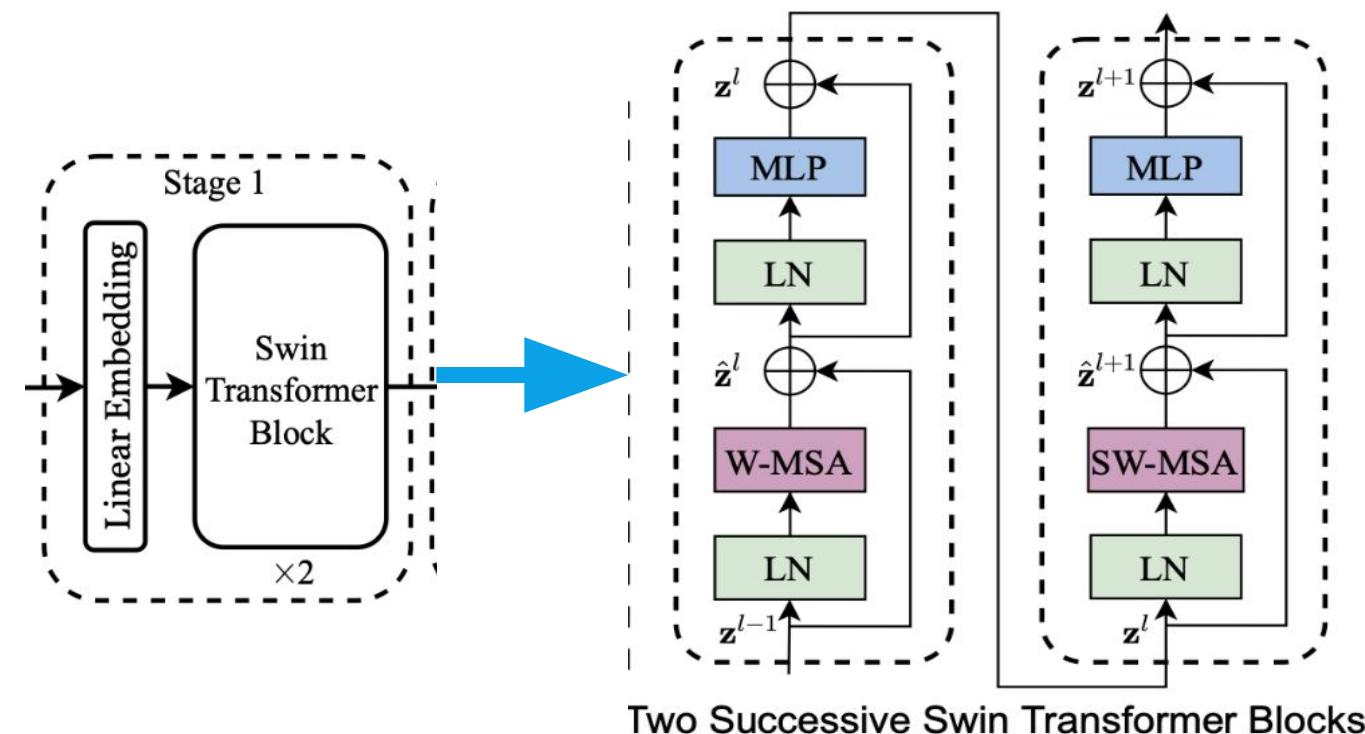
[Swin Transformer Explained \(Video\)](#)



# Swin Transformer

Cada **Stage** contiene pares de **stacks** de bloques Swin-T conectados, la razón es debido al mecanismo de atención utilizado. El stack es lo que permite la extracción de features jerárquicos.

El primer bloque aplica attention sobre ventanas individuales, el segundo bloque aplica atención para encontrar relaciones entre ventanas.

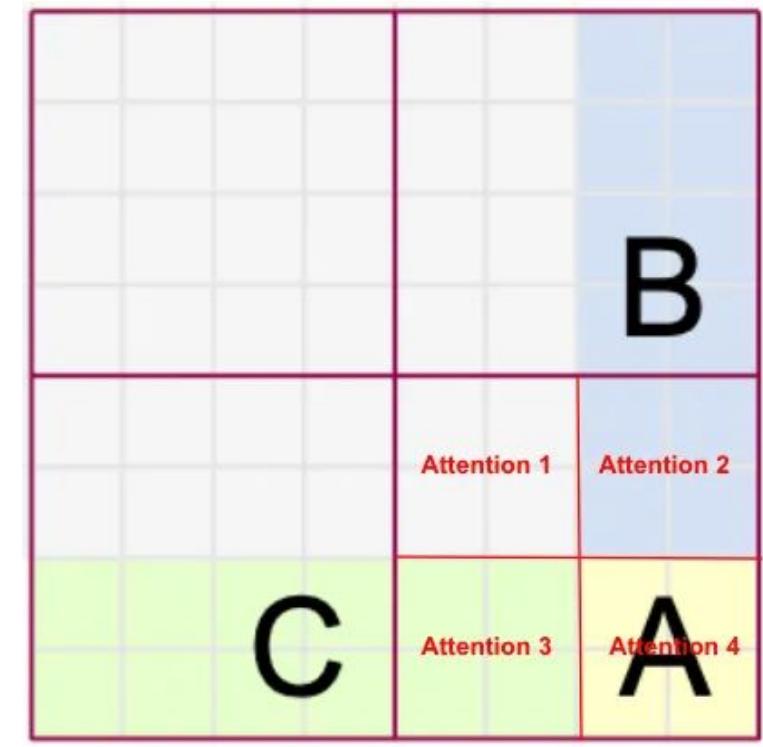
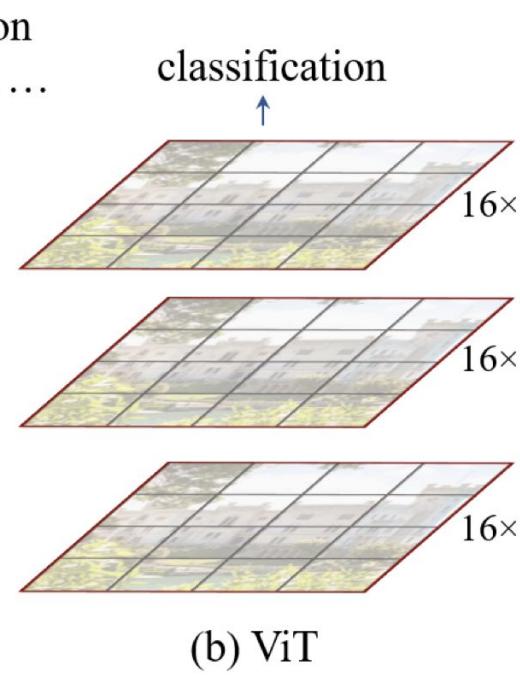
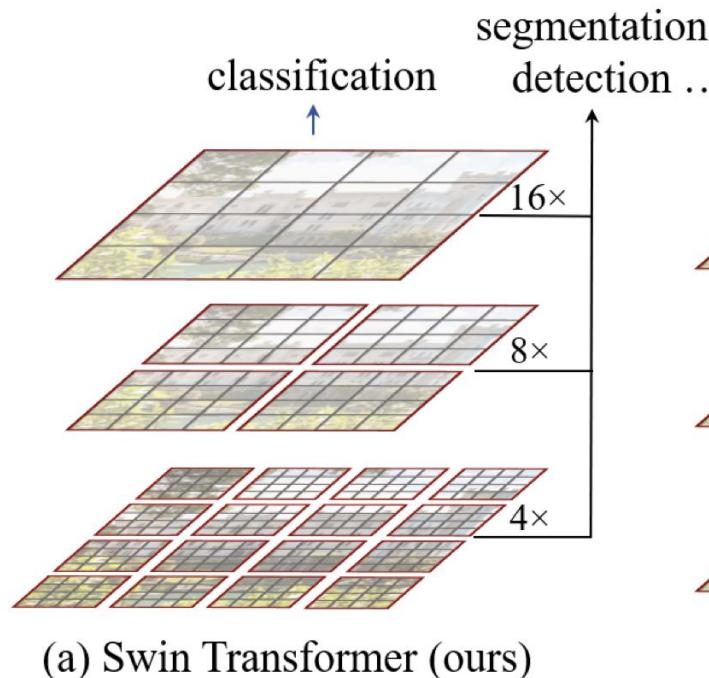


# Window Multi-head Self-Attention

Suponga que existen regiones dentro de la imagen llamadas **ventanas locales** que contienen  $M \times M$  patches:

$$\Omega(W\text{-MSA}) = 4hwC^2 + 2M^2hwC$$

En este caso **W-MSA es lineal** cuando  $M$  es **constante**, haciéndolo escalable. Sin embargo las ventanas están “desconectadas”, en otras palabras “atiende” de manera local.

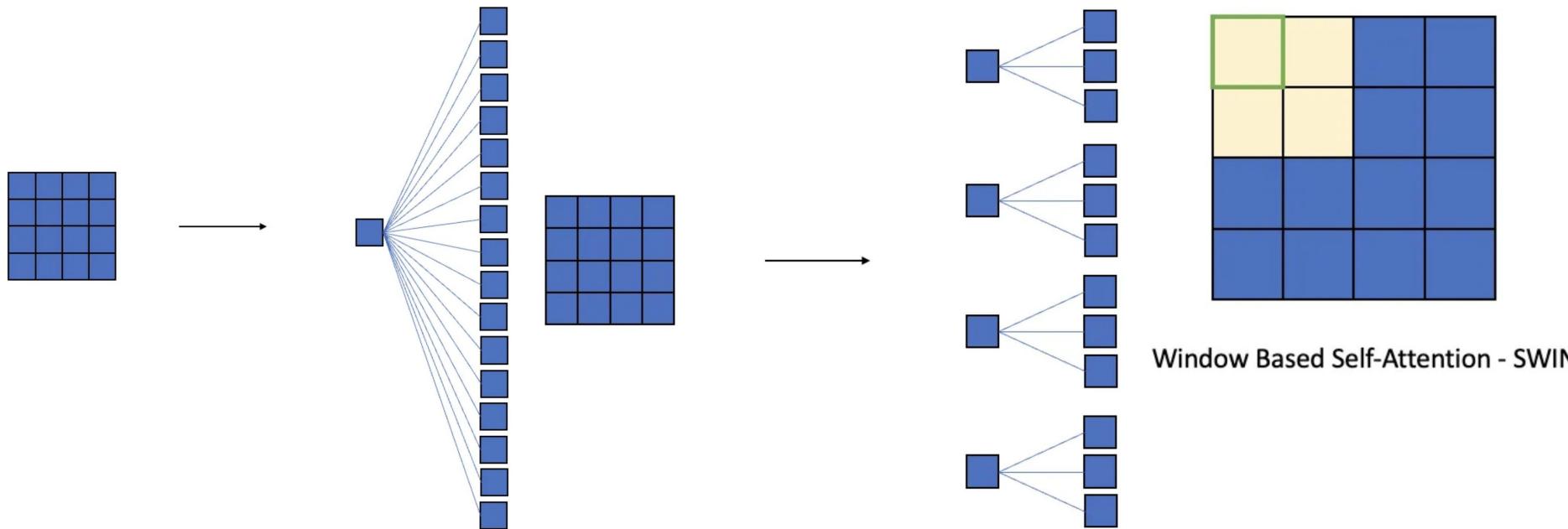


# MSA vs W-MSA

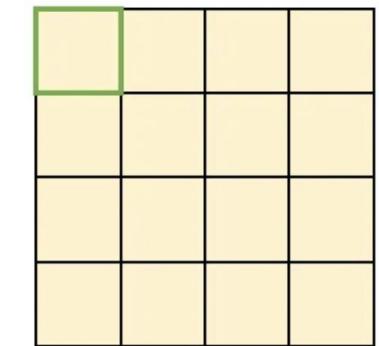
## Swin Transformer Explanation

MSA considera la relación de cada patch **respecto a todos los demás.**

W-MSA considera la relación entre **subconjuntos** de patches.



Window Based Self-Attention - SWIN

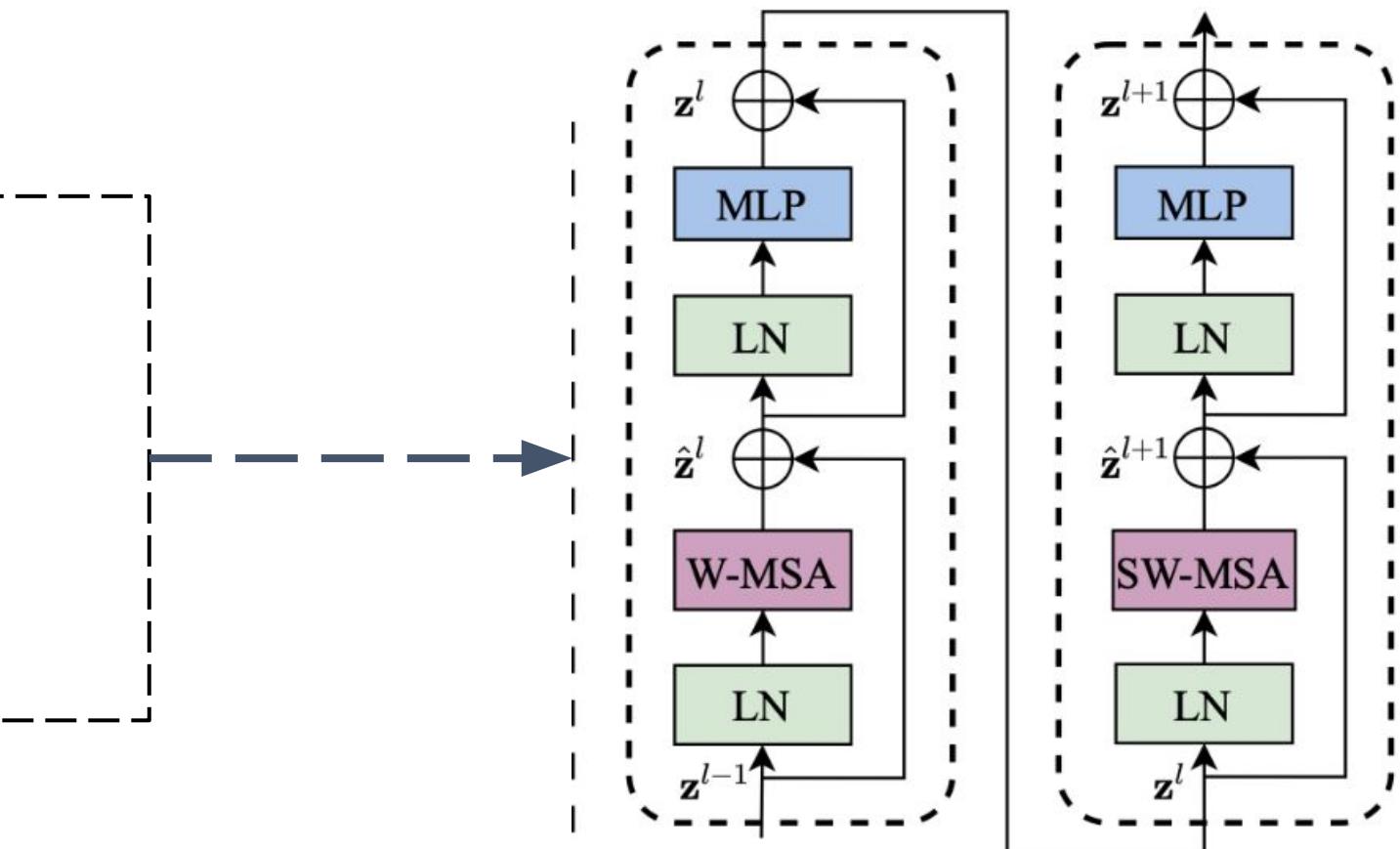


Global Self-Attention - ViT

# Shifted Window Multi-head Self-Attention

W-MSA corresponde a “atención local” sin embargo, para mantener el contexto entre ventanas y evitar perder el **contexto global** se realiza un desfase (shift) entre ventanas, esto sucede entre la conexión de las dos instancias del Swin-T block.

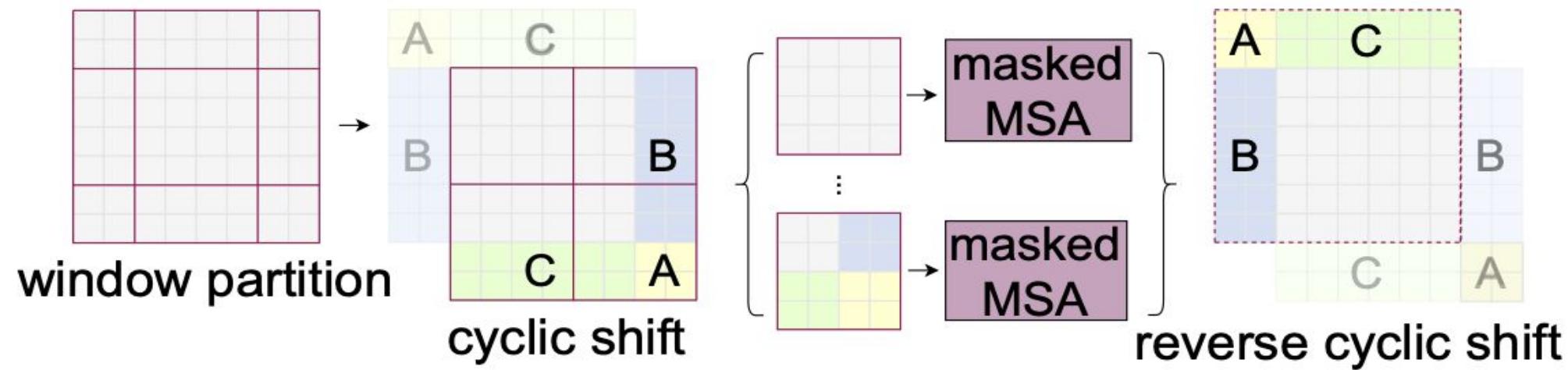
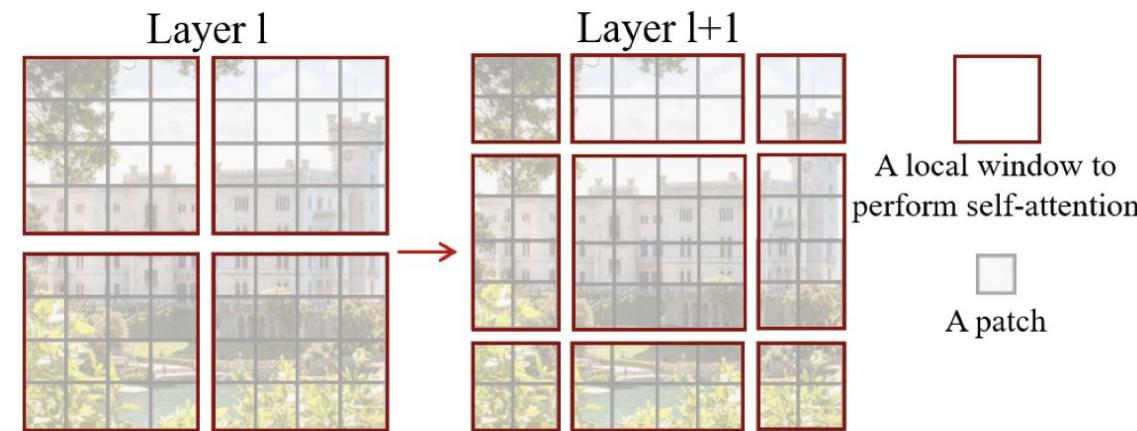
- $\hat{z}^l = (\text{W-MSA})\text{LN}(z^{l-1}) + z^{l-1}$
- $z^l = \text{MLP}(\text{LN}(\hat{z}^l) + \hat{z}^l)$
- $\hat{z}^{l+1} = \text{SW-MSA}(\text{LN}(z^l)) + z^l$
- $z^{l+1} = \text{MLP}(\text{LN}(\hat{z}^{l+1})) + \hat{z}^{l+1}$



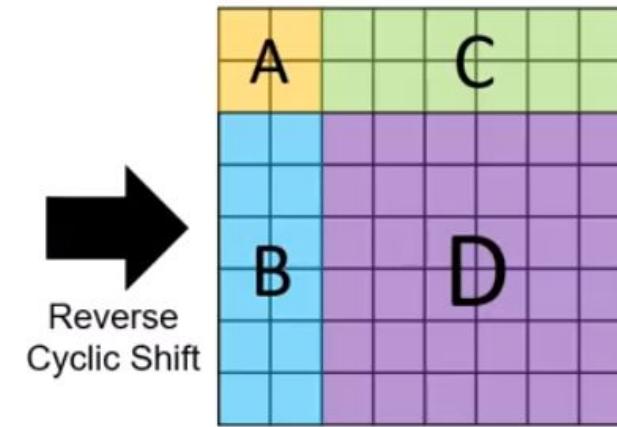
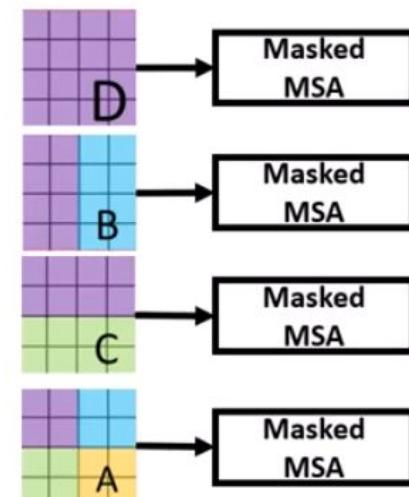
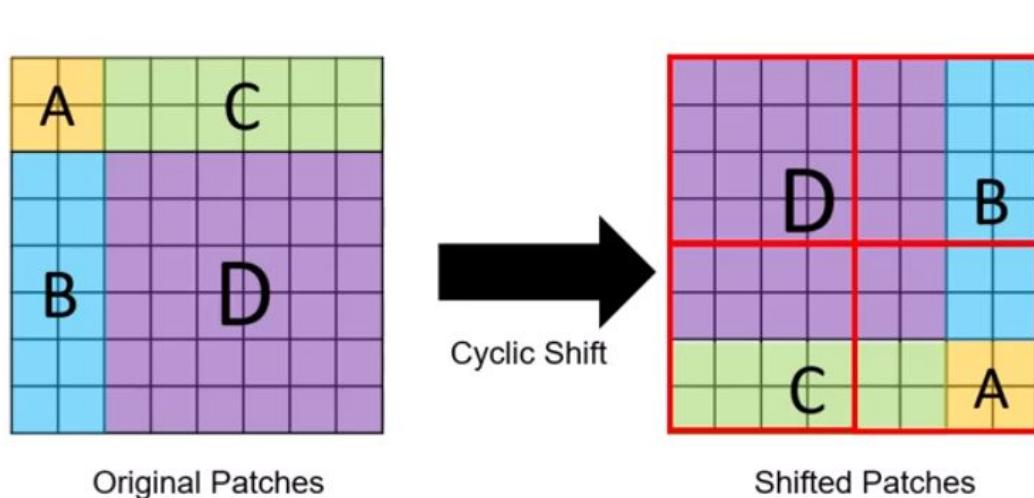
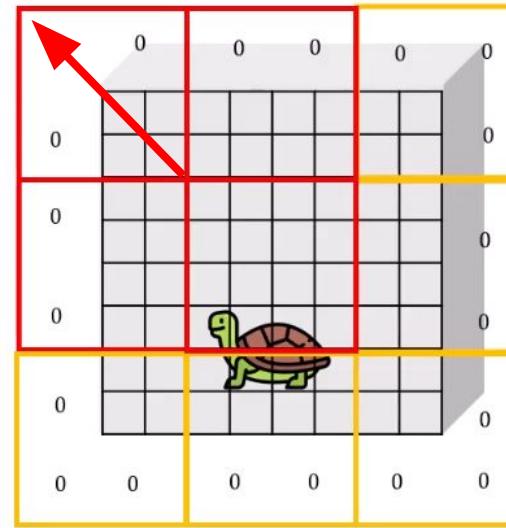
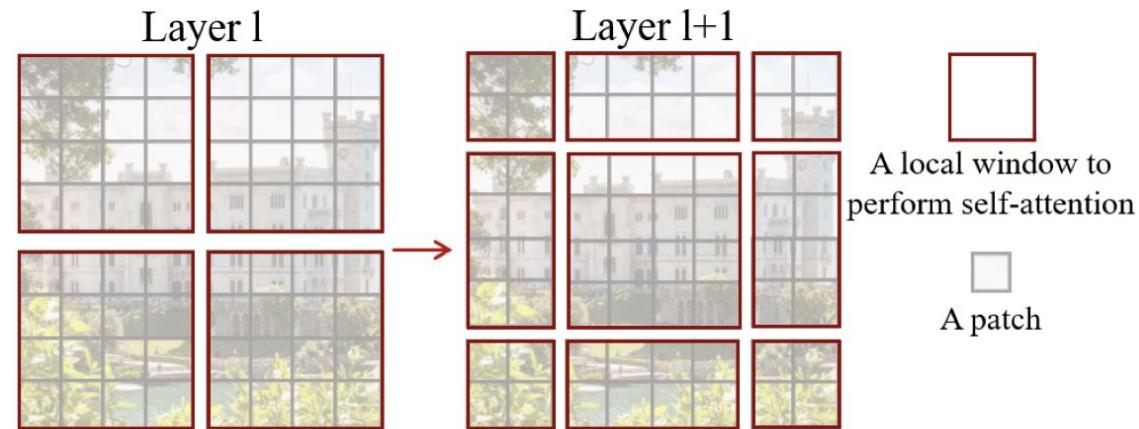
# Shifted Window Multi-head Self-Attention

Las ventanas se dividen en dimensiones de **M/2**, esto introduce el problema de crear ventanas con dimensiones diferentes y sean **ignorados**. Para evitar lo anterior se realiza “**cyclic shift**”, que implica desplazar las ventanas para completar las dimensiones faltantes.

El cyclic shift puede generar áreas las cuales no existían en el feature map anterior. W-MSA, aplica attention sobre estos y se crea una **mezcla parches no relacionados**. La corrección es mediante una máscara para calcular attention solamente en las **sub-ventanas originales**.



# Shifted Window Multi-head Self-Attention



# Swin-Transformer Performance

Entrenado con [ImageNet-22k](#) (clasificación de imágenes), Swin-T logra obtener **métricas superiores** a ViT, con una velocidad de inferencia considerablemente mayor.

[Swin Transformer Github](#)

method	(b) ImageNet-22K pre-trained models				
	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
R-101x3 [38]	384 <sup>2</sup>	388M	204.6G	-	84.4
R-152x4 [38]	480 <sup>2</sup>	937M	840.5G	-	85.4
ViT-B/16 [20]	384 <sup>2</sup>	86M	55.4G	85.9	84.0
ViT-L/16 [20]	384 <sup>2</sup>	307M	190.7G	27.3	85.2
Swin-B	224 <sup>2</sup>	88M	15.4G	278.1	85.2
Swin-B	384 <sup>2</sup>	88M	47.0G	84.7	86.4
Swin-L	384 <sup>2</sup>	197M	103.9G	42.1	87.3

method	MSA in a stage (ms)				Arch. (FPS)		
	S1	S2	S3	S4	T	S	B
sliding window (naive)	122.5	38.3	12.1	7.6	183	109	77
sliding window (kernel)	7.6	4.7	2.7	1.8	488	283	187
Performer [14]	4.8	2.8	1.8	1.5	638	370	241
window (w/o shifting)	2.8	1.7	1.2	0.9	770	444	280
shifted window (padding)	3.3	2.3	1.9	2.2	670	371	236
shifted window (cyclic)	3.0	1.9	1.3	1.0	755	437	278

Table 5. Real speed of different self-attention computation methods and implementations on a V100 GPU.

# Convolutional Vision Transformer (CvT)

CvT fue presentado en el artículo titulado "CvT: Introducing Convolutions to Vision Transformers" por Haiping Wu et.al., ICCV en 2021. [Link-paper](#), [Link-huggingface](#)

ViT tiene **graves problemas** cuando se entrena con **datasets pequeños**. Una propuesta poderosa es el híbrido CvT, el cual goza de propiedades de ViT y CNNs.

CvT promete incrementar el **rendimiento y robustez** de ViT mientras se conserva una **alta eficiencia computacional**. Introduce convolución en dos partes de ViT:

- Reemplaza la proyección lineal por proyección convolucional.
- Utiliza una estructura jerárquica en múltiples etapas similar a CNNs.

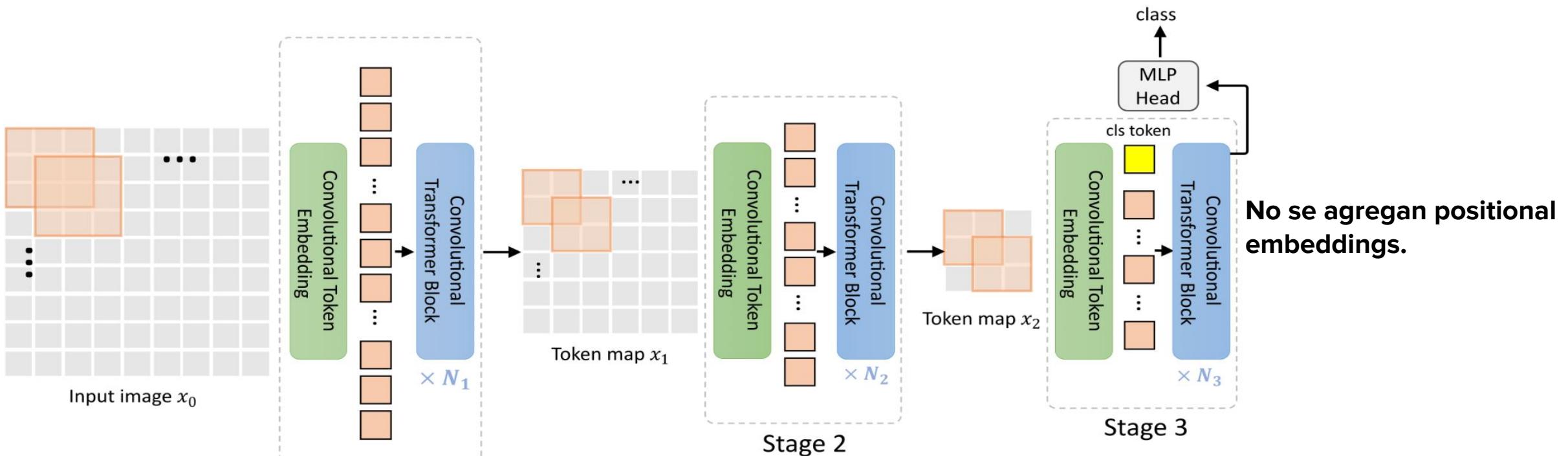
Architecture	Positional Embedding	Token Embedding	Projection	Hierarchical Transformer
ViT	Yes	Non-overlapping	Linear	No
CvT	No	Overlapping (convolution)	Convolution	Yes

# Convolutional Vision Transformer (CvT)

La imagen pasa por una capa de **convolutional token embedding**, lo cual genera un **downsample** de las secuencias, representando patrones visuales complejos, similar a las CNNs.

La introducción de los convolutional projections y embeddings agrega relaciones espaciales locales, lo cual permite a **eliminar** el positional embedding de ViT sin afectar el rendimiento.

Similar al Swin-T, está dividido en jerarquías donde progresivamente se reduce la cantidad de tokens.



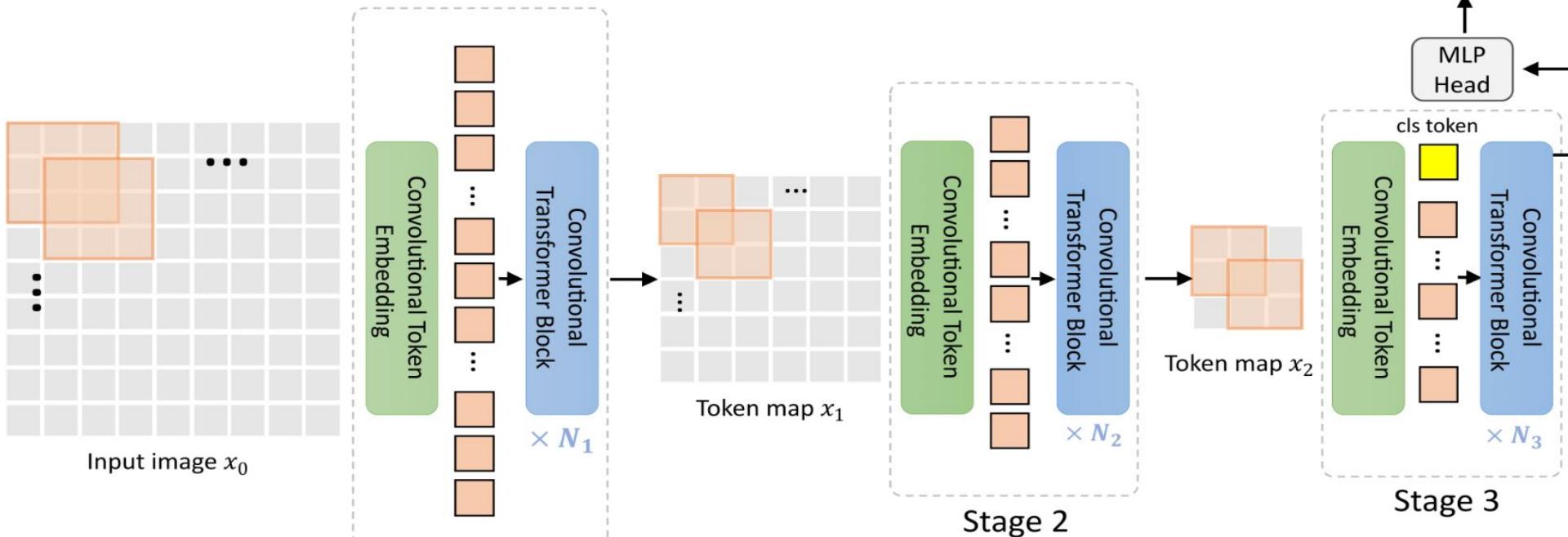
# Convolutional Token Embedding

Utilizando la salida 2D de un mapa de tokens de un estado anterior  $x_{i-1} \in \mathbb{R}^{Hi-1 \times Wi-1 \times Ci-1}$  como entrada al estado  $i$ . Se aprende una función  $f(\square)$  que mapea  $x_{i-1}$  en nuevos tokens  $f(x_{i-1})$  con tamaño de canal  $C_i$  donde  $f(\square)$  es el operador de convolución 2D con:

- kernel Size  $s \times s$ ,
- stride  $s-o$
- padding  $p$ ,

entonces el mapa de tokens se define como:  $f(x_{i-1}) \in \mathbb{R}^{Hi-1 \times Wi-1 \times Ci-1}$

**Se trata de modelar contexto espacial local sobre un enfoque jerárquico.**



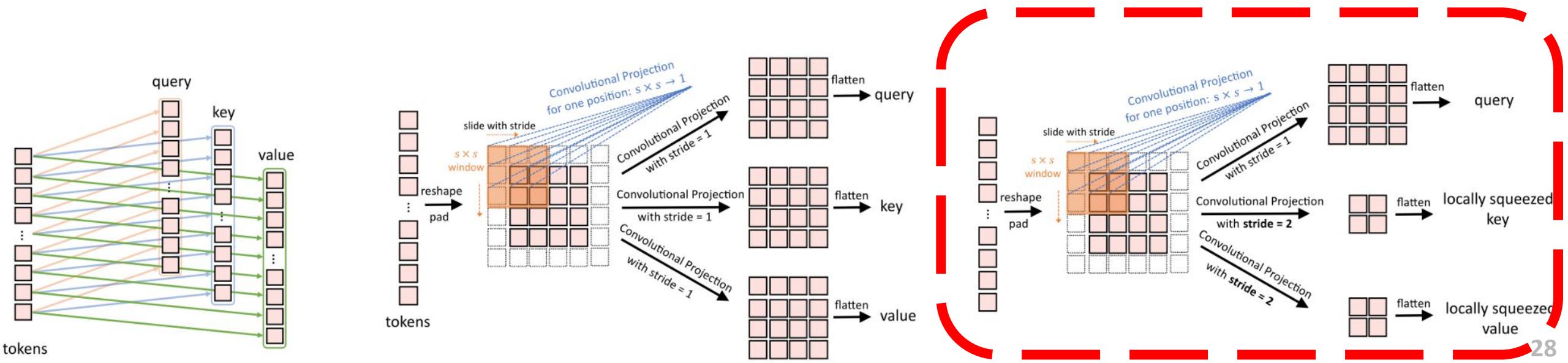
$$H_i = \left\lfloor \frac{H_{i-1} + 2p - s}{s - o} + 1 \right\rfloor$$

$$W_i = \left\lfloor \frac{W_{i-1} + 2p - s}{s - o} + 1 \right\rfloor \quad (1)$$

# Convolutional Projection for Attention

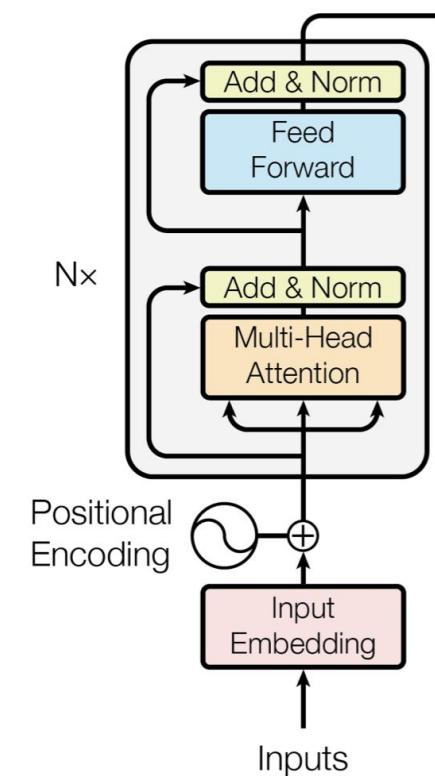
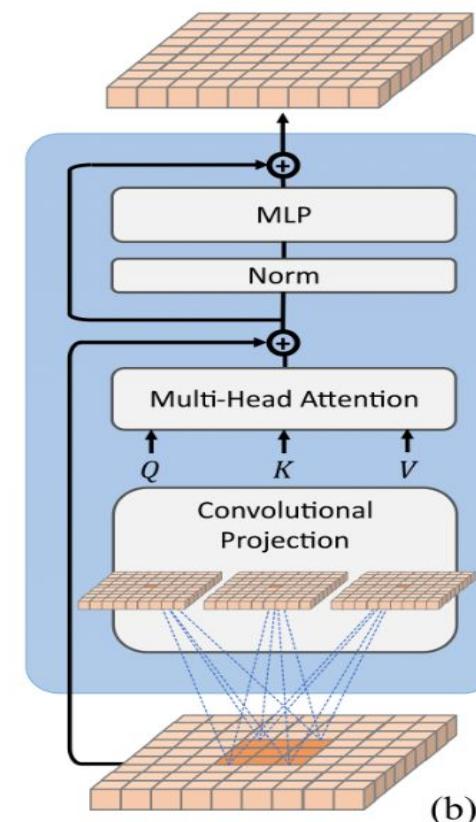
Dentro del **Conv. Transformer Block**, se realiza **Conv. Projection**, el propósito de esta capa es obtener **contexto espacial local** y proveer eficiencia al realizar un **undersampling/squeeze** de las matrices  $k$  y  $v$  de Attention, reduciendo alrededor de **x4** el costo computacional respecto a ViT.

$$x_i^{q/k/v} = \text{Flatten}(\text{Conv2d}(\text{Reshape2D}(x_i), s))$$



# Convolutional Projection for Attention

En este caso, el Attention es el mismo que hemos visto desde el principio, solo que con una proyección convolucional en lugar de lineal.



# CvT Perfomance

Method Type	Network	#Param. (M)	image size	FLOPs (G)	ImageNet top-1 (%)	Real top-1 (%)	V2 top-1 (%)
<i>Convolution Networks</i> <sub>22k</sub>	BiT-M <sub>↑480</sub> [18]	928	480 <sup>2</sup>	837	85.4	-	-
<i>Transformers</i> <sub>22k</sub>	ViT-B/16 <sub>↑384</sub> [11]	86	384 <sup>2</sup>	55.5	84.0	88.4	-
	ViT-L/16 <sub>↑384</sub> [11]	307	384 <sup>2</sup>	191.1	85.2	88.4	-
	ViT-H/16 <sub>↑384</sub> [11]	632	384 <sup>2</sup>	-	85.1	88.7	-
<i>Convolutional Transformers</i> <sub>22k</sub>	<b>Ours:</b> CvT-13 <sub>↑384</sub>	20	384 <sup>2</sup>	16	83.3	88.7	72.9
	<b>Ours:</b> CvT-21 <sub>↑384</sub>	32	384 <sup>2</sup>	25	84.9	89.8	75.6
	<b>Ours:</b> CvT-W24 <sub>↑384</sub>	277	384 <sup>2</sup>	193.2	<b>87.7</b>	<b>90.6</b>	<b>78.8</b>

Method Type	Network	#Param. (M)	image size	FLOPs (G)	ImageNet top-1 (%)	Real top-1 (%)	V2 top-1 (%)
<i>Convolutional Networks</i>	ResNet-50 [15]	25	224 <sup>2</sup>	4.1	76.2	82.5	63.3
	ResNet-101 [15]	45	224 <sup>2</sup>	7.9	77.4	83.7	65.7
	ResNet-152 [15]	60	224 <sup>2</sup>	11	78.3	84.1	67.0
<i>Transformers</i>	ViT-B/16 [11]	86	384 <sup>2</sup>	55.5	77.9	83.6	-
	ViT-L/16 [11]	307	384 <sup>2</sup>	191.1	76.5	82.2	-
	DeiT-S [30][arxiv 2020]	22	224 <sup>2</sup>	4.6	79.8	85.7	68.5
	DeiT-B [30][arxiv 2020]	86	224 <sup>2</sup>	17.6	81.8	86.7	71.5
	PVT-Small [34][arxiv 2021]	25	224 <sup>2</sup>	3.8	79.8	-	-
	PVT-Medium [34][arxiv 2021]	44	224 <sup>2</sup>	6.7	81.2	-	-
	PVT-Large [34][arxiv 2021]	61	224 <sup>2</sup>	9.8	81.7	-	-
	T2T-ViT <sub>t</sub> -14 [41][arxiv 2021]	22	224 <sup>2</sup>	6.1	80.7	-	-
	T2T-ViT <sub>t</sub> -19 [41][arxiv 2021]	39	224 <sup>2</sup>	9.8	81.4	-	-
<i>Convolutional Transformers</i>	T2T-ViT <sub>t</sub> -24 [41][arxiv 2021]	64	224 <sup>2</sup>	15.0	82.2	-	-
	TNT-S [14][arxiv 2021]	24	224 <sup>2</sup>	5.2	81.3	-	-
	TNT-B [14][arxiv 2021]	66	224 <sup>2</sup>	14.1	82.8	-	-
	<b>Ours:</b> CvT-13	20	224 <sup>2</sup>	4.5	81.6	86.7	70.4
	<b>Ours:</b> CvT-21	32	224 <sup>2</sup>	7.1	82.5	87.2	71.3
	<b>Ours:</b> CvT-13 <sub>↑384</sub>	20	384 <sup>2</sup>	16.3	83.0	87.9	71.9
	<b>Ours:</b> CvT-21 <sub>↑384</sub>	32	384 <sup>2</sup>	24.9	<b>83.3</b>	<b>87.7</b>	<b>71.9</b>
<b>Ours:</b> CvT-13-NAS		18	224 <sup>2</sup>	4.1	82.2	87.5	71.3

Entrenado con **Imagenet-22k** CvT supera considerablemente a ViT.

**Imagenet-1K** obtiene métricas mejores a ViT, esto significa que CvT es una propuesta apta para datasets pequeños.

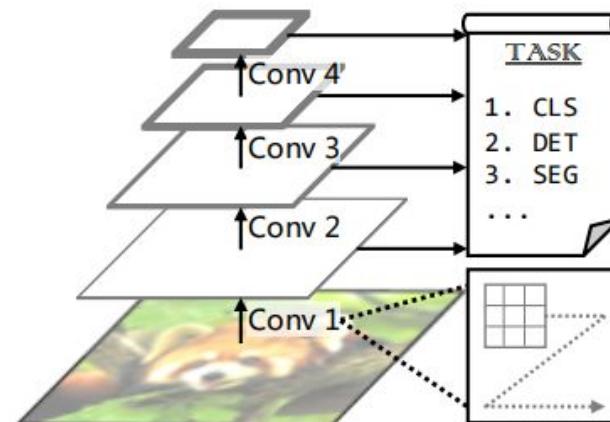
Implementación oficial

# PyramidViT (PvT)

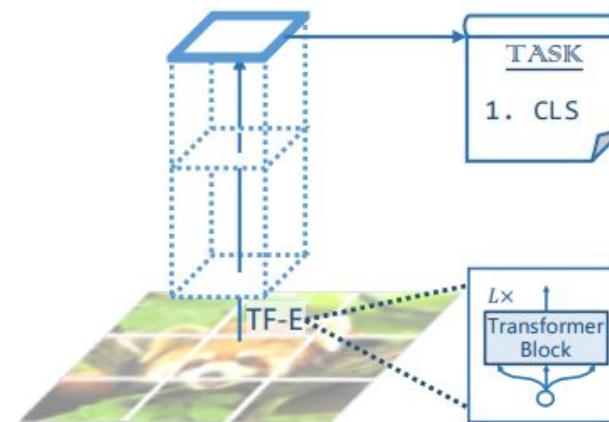
- Pyramid Vision Transformer (PVT) fue propuesta en “Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions” por Wenhui Wang et.al. en 2021 como una alternativa a las CNNs sin usar convoluciones, diseñada para tareas de predicción densa como detección de objetos y segmentación. [Link-paper](#), [Link-huggingface](#)
- A diferencia de ViT, PVT genera salidas de alta resolución con menores costos computacionales y de memoria.
- Combina ventajas de CNNs y Transformers, convirtiéndose en un backbone unificado para diversas tareas de visión.
- Utiliza una pirámide progresiva y atención con reducción espacial para mejorar la resolución bajo recursos limitados.

# PyramidViT (PvT)

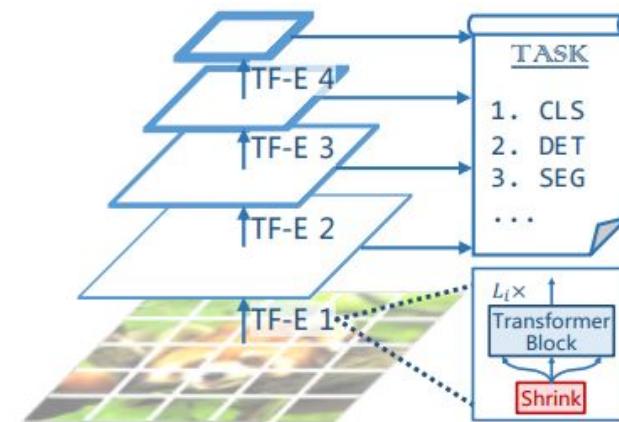
PvT incorpora la estructura piramidal comúnmente encontrada en CNNs, con el propósito de ser backbone de múltiples tareas de visión artificial. Es muy similar a Swin-T y CvT.



(a) CNNs: VGG [54], ResNet [22], *etc.*



(b) Vision Transformer [13]



(c) Pyramid Vision Transformer (ours)

# PyramidViT (PvT)

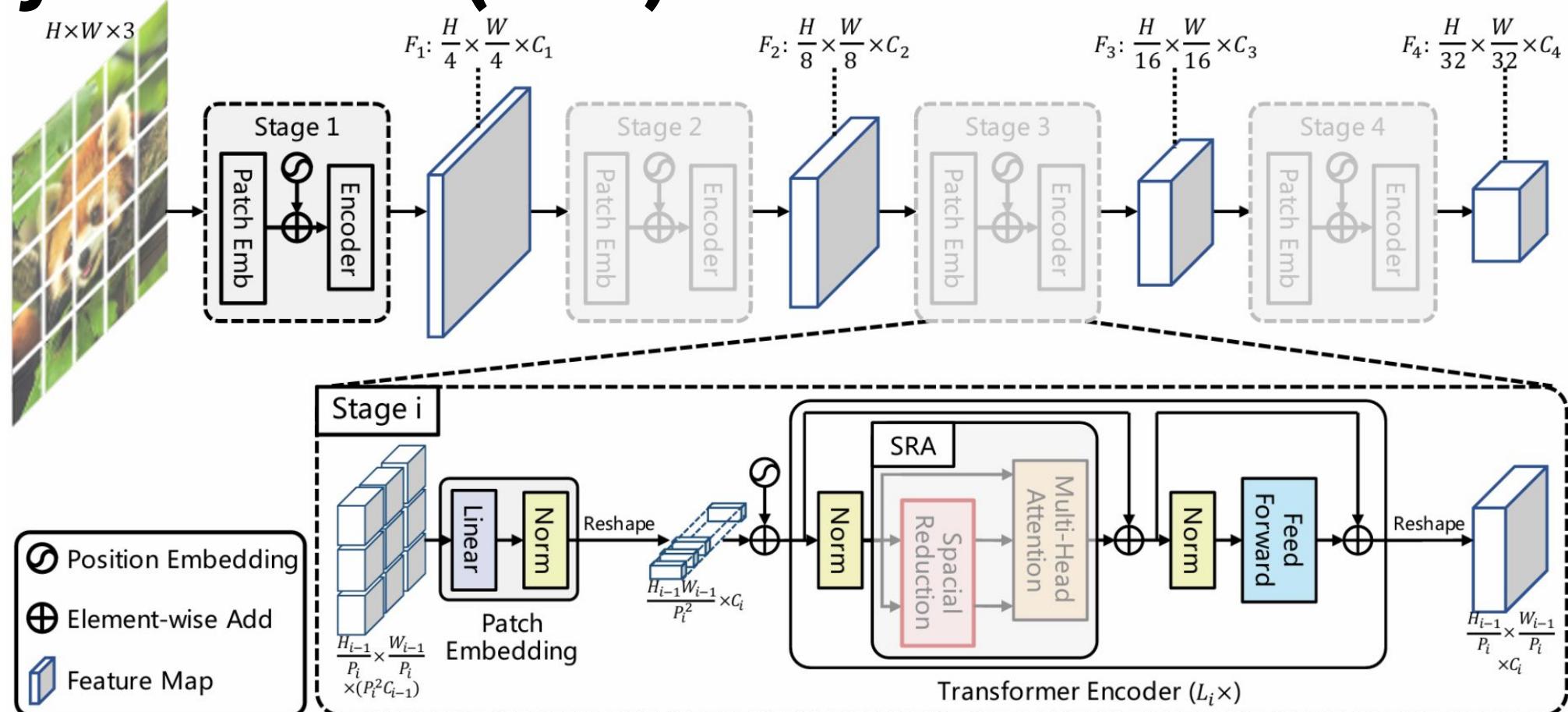
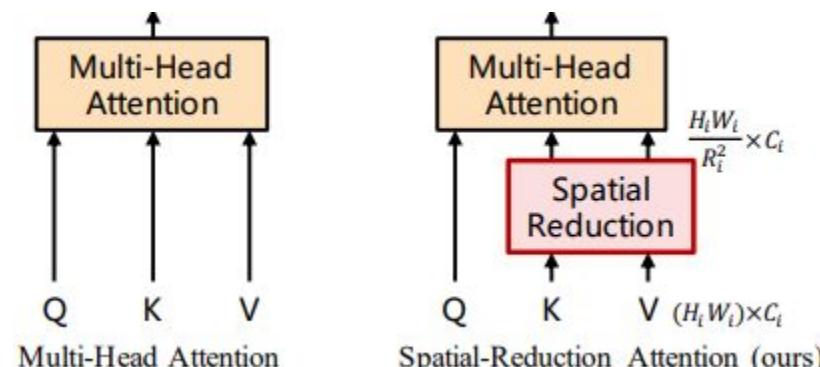


Figure 3: **Overall architecture of Pyramid Vision Transformer (PvT).** The entire model is divided into four stages, each of which is comprised of a patch embedding layer and a  $L_i$ -layer Transformer encoder. Following a pyramid structure, the output resolution of the four stages progressively shrinks from high (4-stride) to low (32-stride).

# Spatial-Reduction Attention (SRA)

SRA, busca reducir las matrices K y V (similar a CvT), la reducción permite manipular secuencias más grandes.

**Norm = LayerNorm**



$$\text{SRA}(Q, K, V) = \text{Concat}(\text{head}_0, \dots, \text{head}_{N_i})W^O,$$

$$\text{head}_j = \text{Attention}(QW_j^Q, \text{SR}(K)W_j^K, \text{SR}(V)W_j^V),$$

$$\text{SR}(\mathbf{x}) = \text{Norm}(\text{Reshape}(\mathbf{x}, R_i)W^S).$$

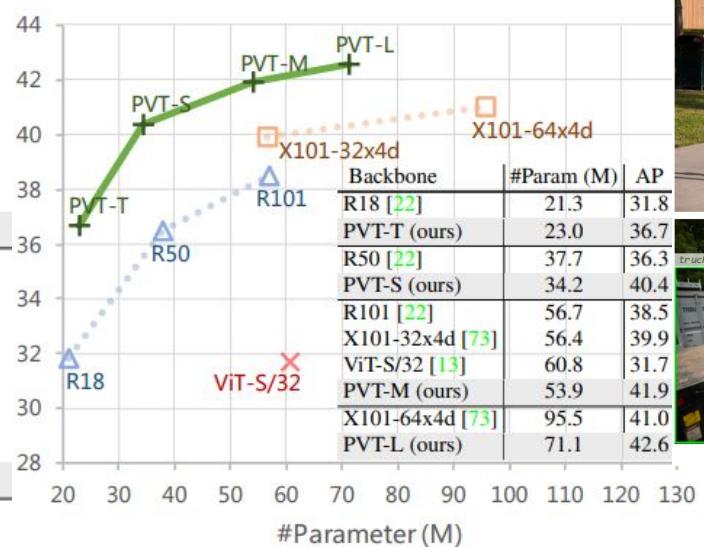
$$\text{Reshape}(\mathbf{x}, R_i) \rightarrow \frac{H_i W_i}{R_i^2} \times \bar{(R_i^2 C_i)}$$

$$\text{Attention}(\mathbf{q}, \mathbf{k}, \mathbf{v}) = \text{Softmax}\left(\frac{\mathbf{q} \mathbf{k}^\top}{\sqrt{d_{\text{head}}}}\right)\mathbf{v}.$$

# PvT Performance

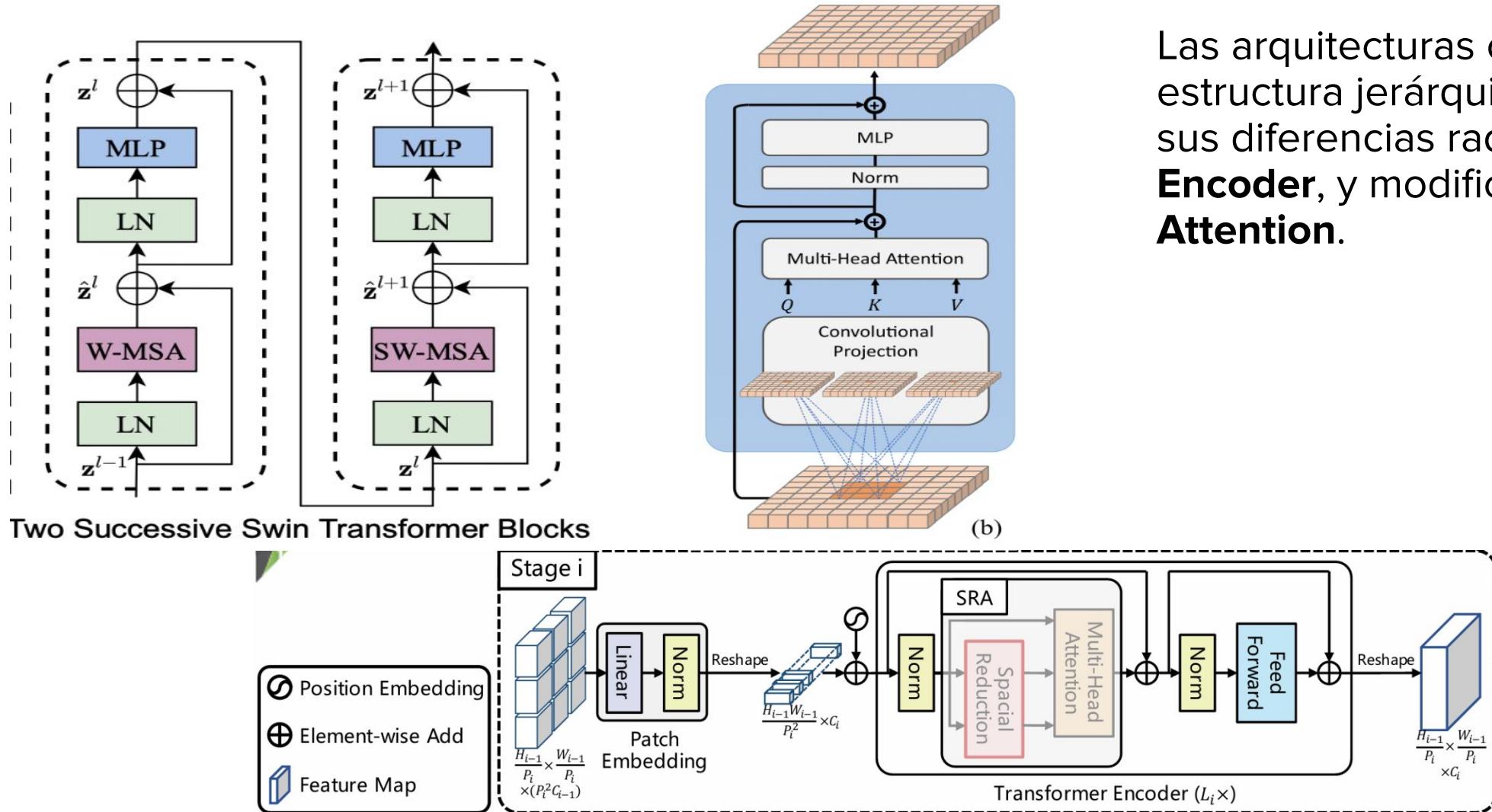
En RetinaNet, PvT, demuestra muy buenos resultados en versus ResNet50.

Method	#Param (M)	GFLOPs	Top-1 Err (%)
ResNet18* [22]	11.7	1.8	30.2
ResNet18 [22]	11.7	1.8	31.5
DeiT-Tiny/16 [63]	5.7	1.3	27.8
PVT-Tiny (ours)	13.2	1.9	24.9
ResNet50* [22]	25.6	4.1	23.9
ResNet50 [22]	25.6	4.1	21.5
ResNeXt50-32x4d* [73]	25.0	4.3	22.4
ResNeXt50-32x4d [73]	25.0	4.3	20.5
T2T-ViT <sub>t</sub> -14 [75]	22.0	6.1	19.3
TNT-S [19]	23.8	5.2	18.7
DeiT-Small/16 [63]	22.1	4.6	20.1
PVT-Small (ours)	24.5	3.8	20.2
ResNet101* [22]	44.7	7.9	22.6
ResNet101 [22]	44.7	7.9	20.2
ResNeXt101-32x4d* [73]	44.2	8.0	21.2
ResNeXt101-32x4d [73]	44.2	8.0	19.4
T2T-ViT <sub>t</sub> -19 [75]	39.0	9.8	18.6
ViT-Small/16 [13]	48.8	9.9	19.2
PVT-Medium (ours)	44.2	6.7	18.8
ResNeXt101-64x4d* [73]	83.5	15.6	20.4
ResNeXt101-64x4d [73]	83.5	15.6	18.5
ViT-Base/16 [13]	86.6	17.6	18.2
T2T-ViT <sub>t</sub> -24 [75]	64.0	15.0	17.8
TNT-B [19]	66.0	14.1	17.2
DeiT-Base/16 [63]	86.6	17.6	18.2
PVT-Large (ours)	61.4	9.8	18.3



Method	Scale	GFLOPs	Time (ms)	RetinaNet 1x		
				AP	AP <sub>50</sub>	AP <sub>75</sub>
ResNet50 [22]	800	239.3	55.9	36.3	55.3	38.6
PVT-Small (ours)	640	157.2	51.7	38.7	59.3	40.8

# PvT vs Swin-T vs CvT



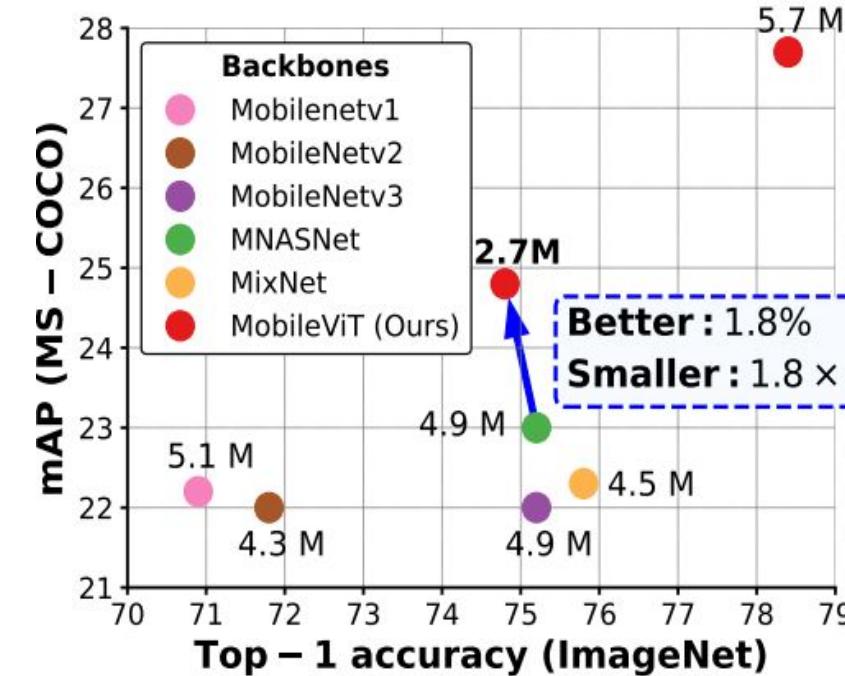
Las arquitecturas comparten la estructura jerárquica sin embargo, sus diferencias radican en el **Encoder**, y modificaciones de **Attention**.

# MobileViT

- Propuesto en “ MOBILEVIT: Light-Weight, general-purpose, and mobile-friendly vision transformer” por Sachin Mehta y Mohammad Rastegari en 2022, combina lo mejor de CNNs y ViTs **para tareas de visión en dispositivos móviles.** [Link-paper](#), [Link-huggingface](#)
- Sustituye el procesamiento local de las convoluciones con procesamiento global usando transformers.
- Supera a redes basadas en CNN y ViT en precisión con un modelo ligero y eficiente, obteniendo un 78.4% de precisión en ImageNet-1k.

# MobileViT

MobileViT enfatiza el uso de ViTs ligeros, de baja latencia, y precisos para aplicaciones en dispositivos con **recursos restringidos**. Para alcanzar este objetivo, la idea es aprender representaciones globales con transformers como convolución.



Model	# Params. ↓	FLOPs ↓	Time ↓	Top-1 ↑
MobileNetv2	3.5 M	0.3 G	0.92 ms	73.3
MobileViT-XS (Ours; 8,4,2)	<b>2.3 M</b>	0.7 G	5.93 ms	73.8
MobileViT-XS (Ours; 2,2,2)	<b>2.3 M</b>	0.7 G	7.28 ms	<b>74.8</b>

(a) ImageNet-1k classification

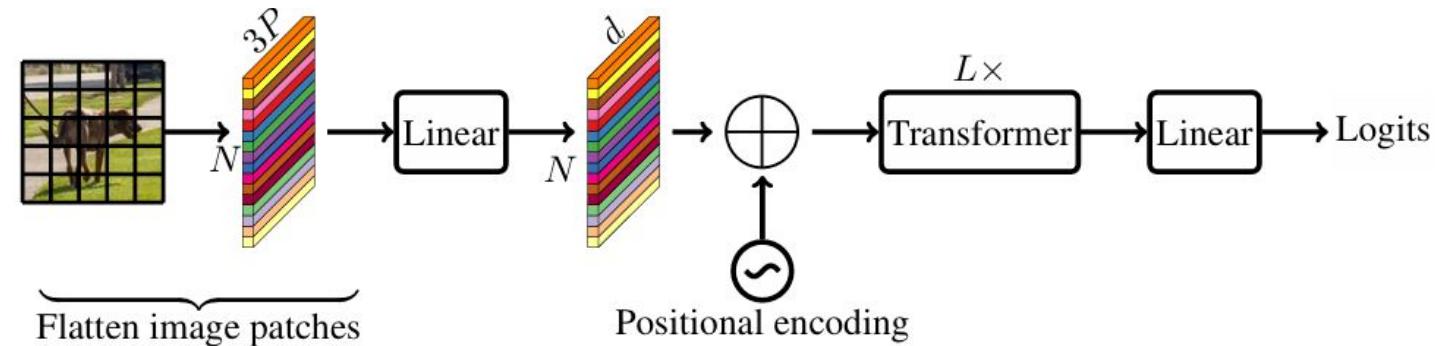
Backbone	# Params. ↓	FLOPs ↓	Time ↓	mAP ↑
MobileNetv2	4.3 M	0.8 G	2.3 ms	22.1
MobileViT-XS (Ours; 8,4,2)	<b>2.7 M</b>	1.6 G	10.7 ms	23.1
MobileViT-XS (Ours; 2,2,2)	<b>2.7 M</b>	1.6 G	12.6 ms	<b>24.8</b>

(b) Object detection w/ SSDLite.

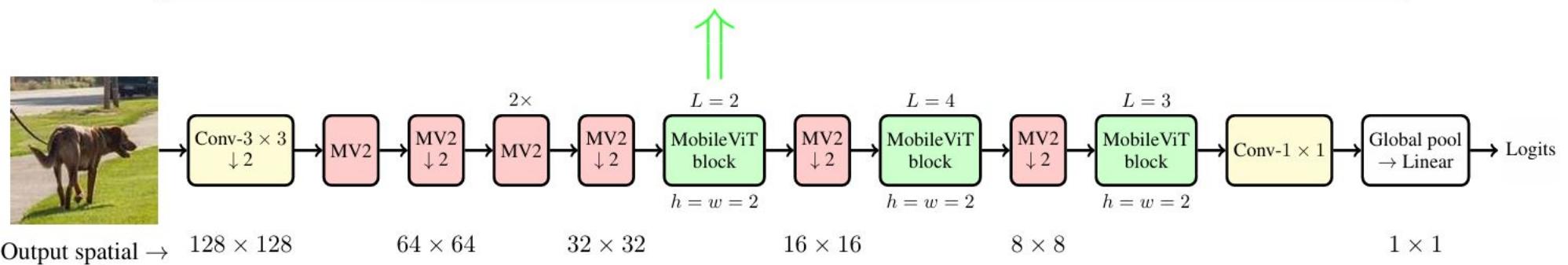
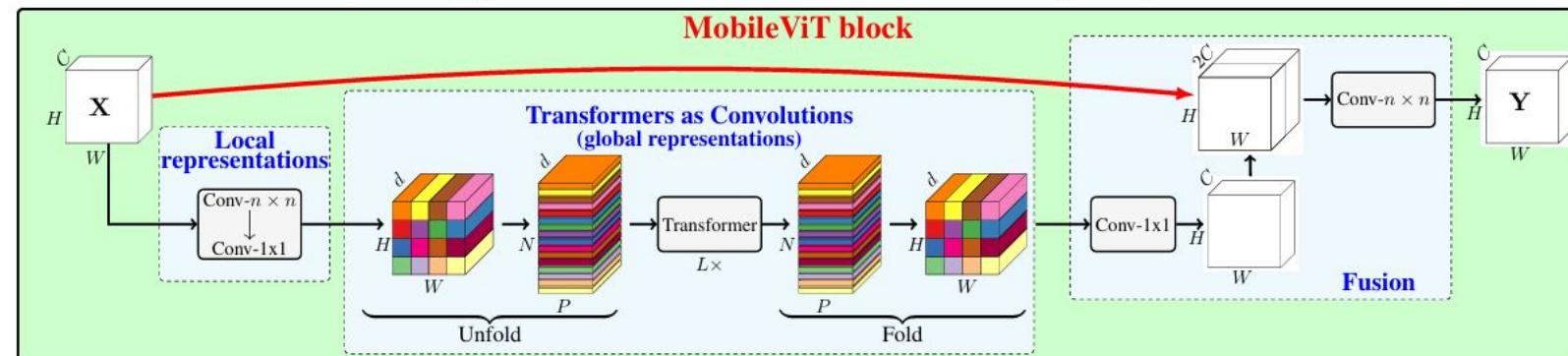
Backbone	# Params. ↓	FLOPs ↓	Time ↓	mIOU ↑
MobileNetv2	4.3 M	5.8 G	6.5 ms	75.7
MobileViT-XS (Ours)	<b>2.9 M</b>	<b>5.7 G</b>	25.1 ms	75.4
MobileViT-XS (Ours)	<b>2.9 M</b>	<b>5.7 G</b>	32.3 ms	<b>77.1</b>

(c) Semantic segmentation w/ DeepLabv3.

# MobileViT



(a) Standard visual transformer (ViT)



# MobileViT

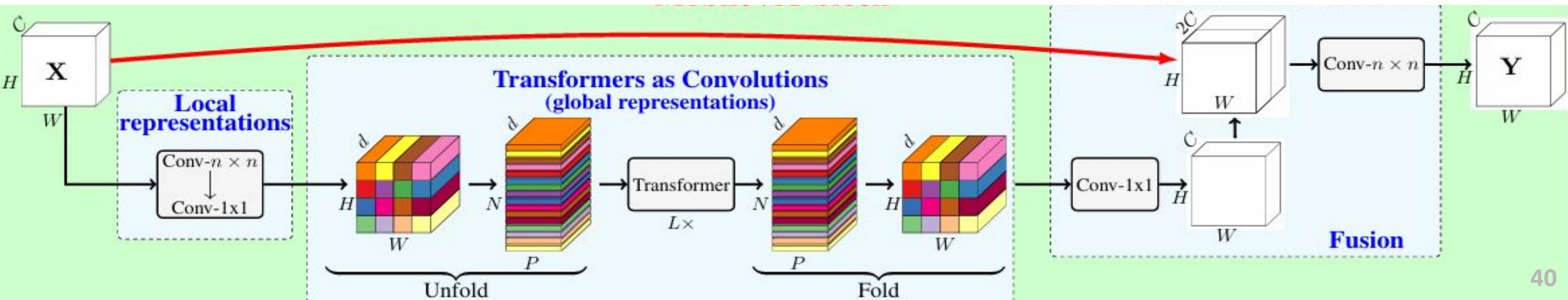
Dada la entrada  $x \in \mathbb{R}^{H \times W \times C}$  se realiza una conv.  $n \times n$  seguida de una conv.  $1 \times 1$  para producir  $x_L \in \mathbb{R}^{H \times W \times d}$ .

*La conv.  $n \times n$  encodifica información **espacial local**, mientras que la conv.  $1 \times 1$  **proyecta el tensor a un espacio d-dimensional** donde  $d > C$  (channels).*

Para lograr aprender representaciones **globales** se **desglosa**  $x_L$  en parches no traslapantes  $x_U \in \mathbb{R}^{P \times W \times d}$ , donde  $P = wh$  y  $N = HW/P$ .

Para cada  $p \in \{1, \dots, P\}$ , las relaciones entre parches son codificadas mediante un Transformer para obtener  $x_G \in \mathbb{R}^{P \times N \times d}$

$$\mathbf{X}_G(p) = \text{Transformer}(\mathbf{X}_U(p)), 1 \leq p \leq P$$



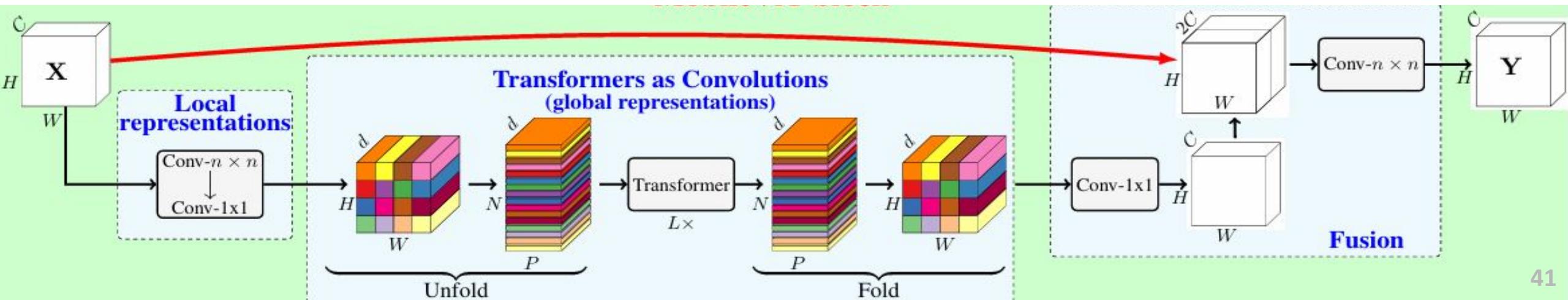
# MobileViT

MobileViT no pierde el orden de los parches, por lo tanto se pueden unir los parches a  $x_F \in \mathbf{R}^{H \times W \times d}$ ,  $x_F$  es proyectado en un espacio C-dimensional mediante conv. y combinando con la entrada  $x$  mediante concat. Posteriormente una conv.  $n \times n$  es utilizada para fusionar los features de concat.

$x_U$  contiene información local.

$x_G$  contiene información global.

$$\mathbf{X}_G(p) = \text{Transformer}(\mathbf{X}_U(p)), 1 \leq p \leq P$$

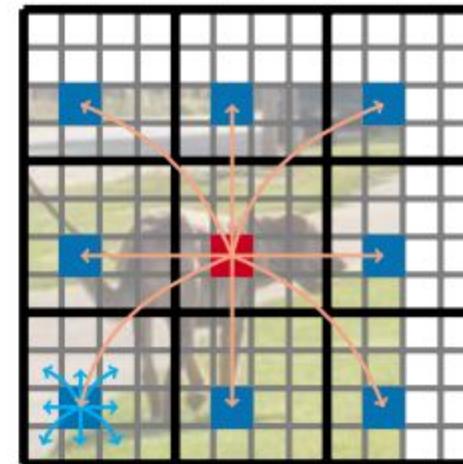
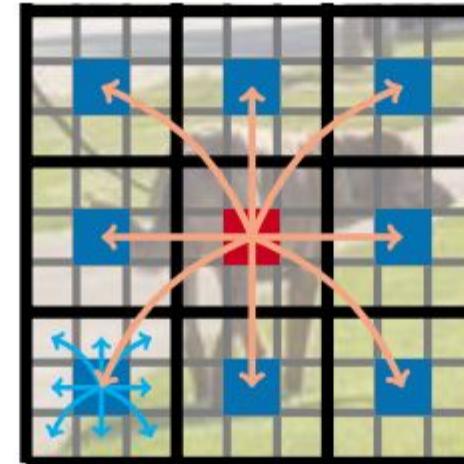
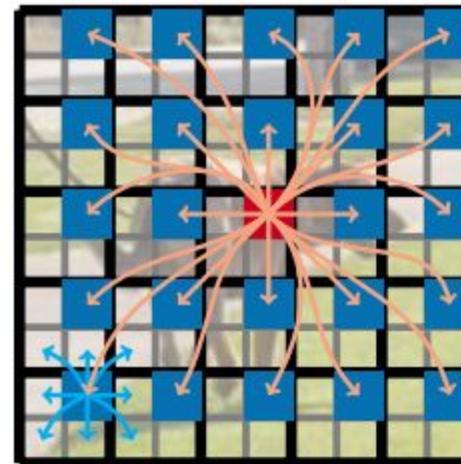


# MobileViT Attention

En MobileViT el M-HSA **sigue siendo  $O(N^2)$** .

Los pixeles en rojo atienden los pixeles en azul, donde los azules contienen información espacial gracias a la conv.

MobileViT no realiza cambios sobre el Transformer utilizado en ViT, por lo tanto existen potenciales mejoras, como modificar el Attention.



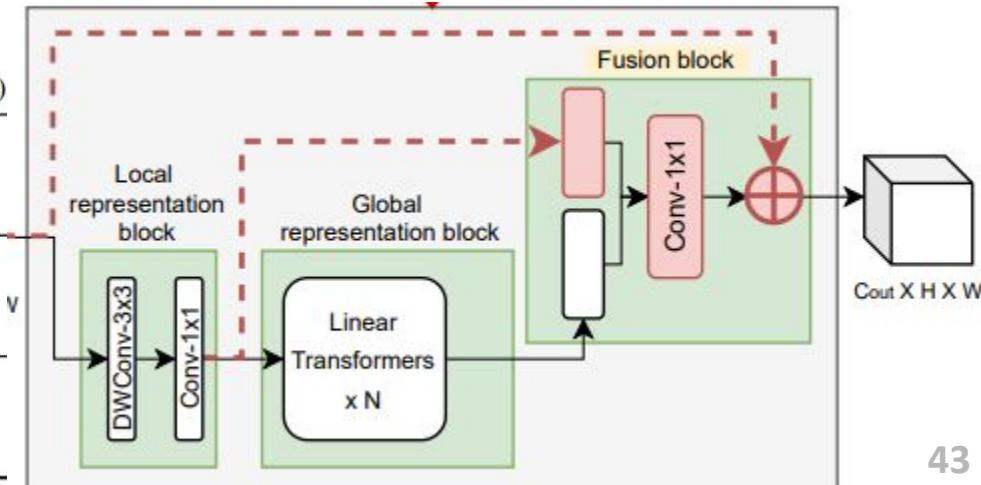
# MobileViT Variantes

MobileViTv2, modifica self-Attention por una implementación **O(N)** con el énfasis de reducir la latencia en un Iphone 12 se obtuvieron los resultados:

Attention unit	Latency ↓	Top-1 ↑
Self-attention in Transformer (Fig. 3a; [5])	9.9 ms	<b>78.4</b>
Self-attention in Linformer (Fig. 3b; [10])	10.2 ms	78.2
Separable self-attention (Ours; Fig. 3c)	<b>3.4</b> ms	78.1

MobileViTv3, modifica el último paso en v1 que es el Fusion Block

Model	Blocks (↓)	FLOPs (M)↓	# Params (M)↓	Top-1 (%)↑	Throughput (↑)	# Time (ms) (↓)
MobileViTv1-XXS	4	364	1.30	69.00	2124	7.24
MobileViTv3-XXS	4	289	1.25	70.98 (↑1.98%)	2146	7.12
<b>MobileViTv3-XXS</b>	<b>2</b>	256 (↓30%)	1.14	<b>70.23</b> (↑1.23%)	<b>2308</b>	<b>6.24</b>
MobileViTv1-XS	4	986	2.3	74.80	1097	7.32
MobileViTv3-XS	4	927	2.5	76.7 (↑1.9%)	1078	7.20
<b>MobileViTv3-XS</b>	<b>2</b>	853 (↓13.5%)	2.3	<b>76.1</b> (↑1.3%)	<b>1129</b>	<b>6.35</b>
MobileViTv1-S	4	2009	5.6	78.40	822	7.34
MobileViTv3-S	4	1841	5.8	79.3 (↑0.9%)	824	7.29
<b>MobileViTv3-S</b>	<b>2</b>	1651 (↓17.82%)	5.2	<b>79.06</b> (↑0.6%)	<b>876</b>	<b>6.60</b>



# TP-II

El trabajo práctico se encuentra en el [GitHub](#) de la materia CEIA-ViT:  
**Plazo de entrega antes de la clase 4.**

[CEIA-ViT/TrabajosPracticos/TP2 at main · FIUBA-Posgrado-Inteligencia-Artificial/CEIA-ViT · GitHub](#)

# Bibliografía

- Transformers and Visual Transformers, Part of the book series: [Neuromethods \(\(NM,volume 197\)\)](#) [Link](#)
- Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows, Ze Liu, Yutong Lin et.al, ICCV 2021 ([Link-paper](#), [Link-huggingface](#))
- CvT: Introducing Convolutions to Vision Transformers, Haiping Wu et.al., ICCV 2021. ([Link-paper](#), [Link-huggingface](#))
- MOBILEVIT: Light-Weight, general-purpose, and mobile-friendly vision transformer”, Sachin Mehta y Mohammad Rastegari. ([Link-paper](#), [Link-huggingface](#))
- Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions, Wenhui Wang et.al., 2021. ([Link-paper](#), [Link-huggingface](#))

# Preguntas?