

# Sesión 3.1

## *Object recognition*

*YOLO. Pose & Depth estimation*

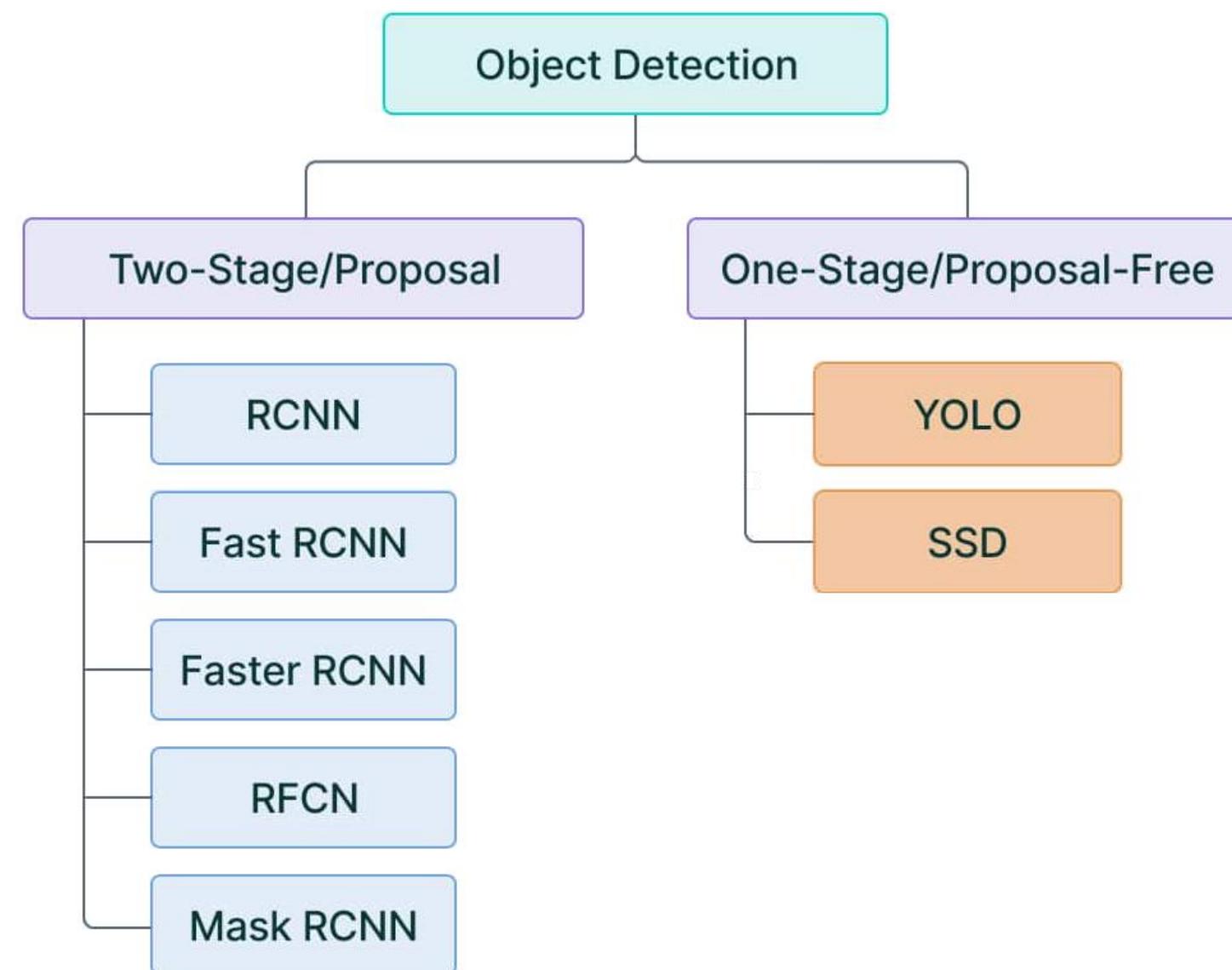
1.



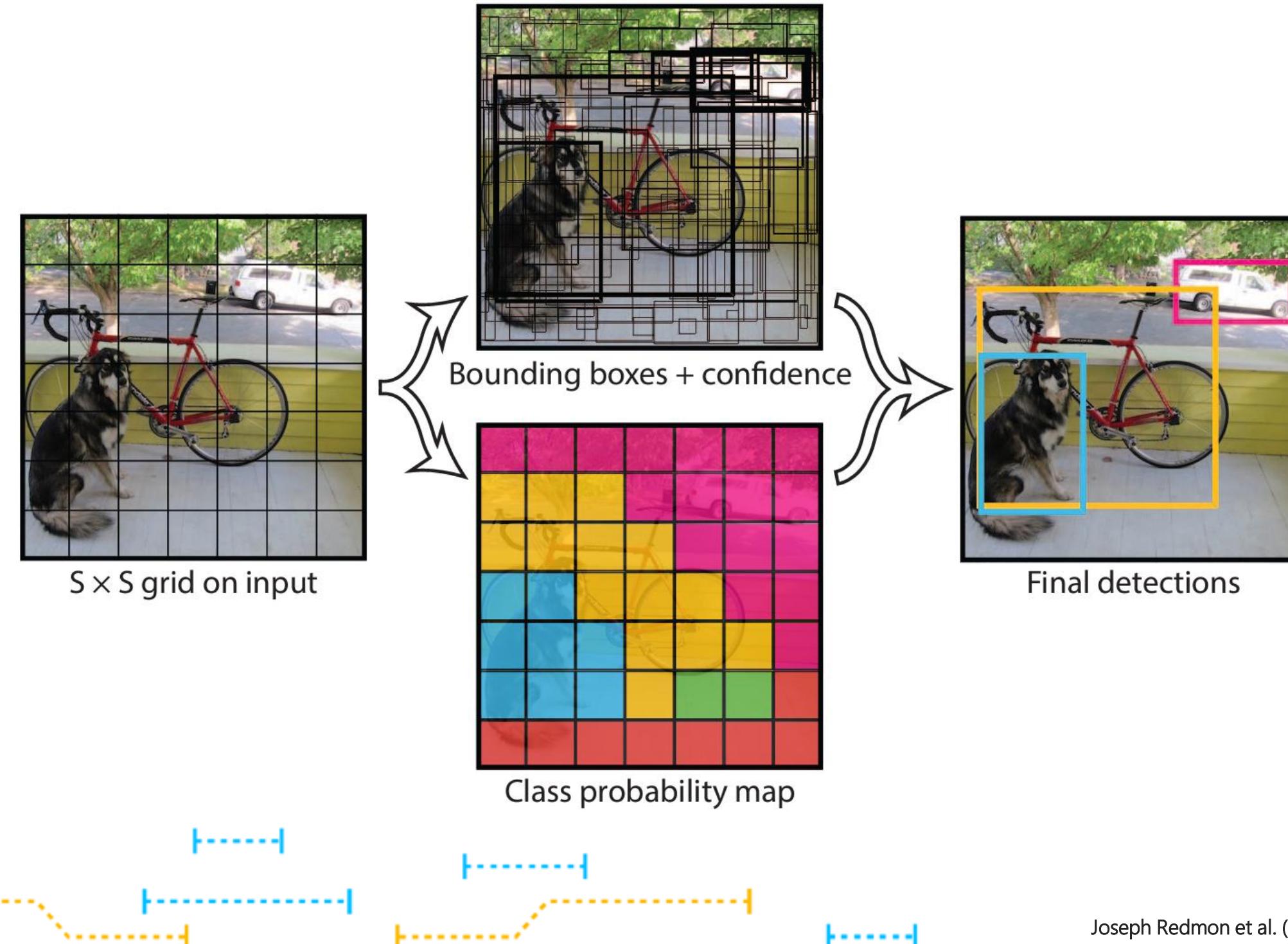
**TRANSFORMATEC**



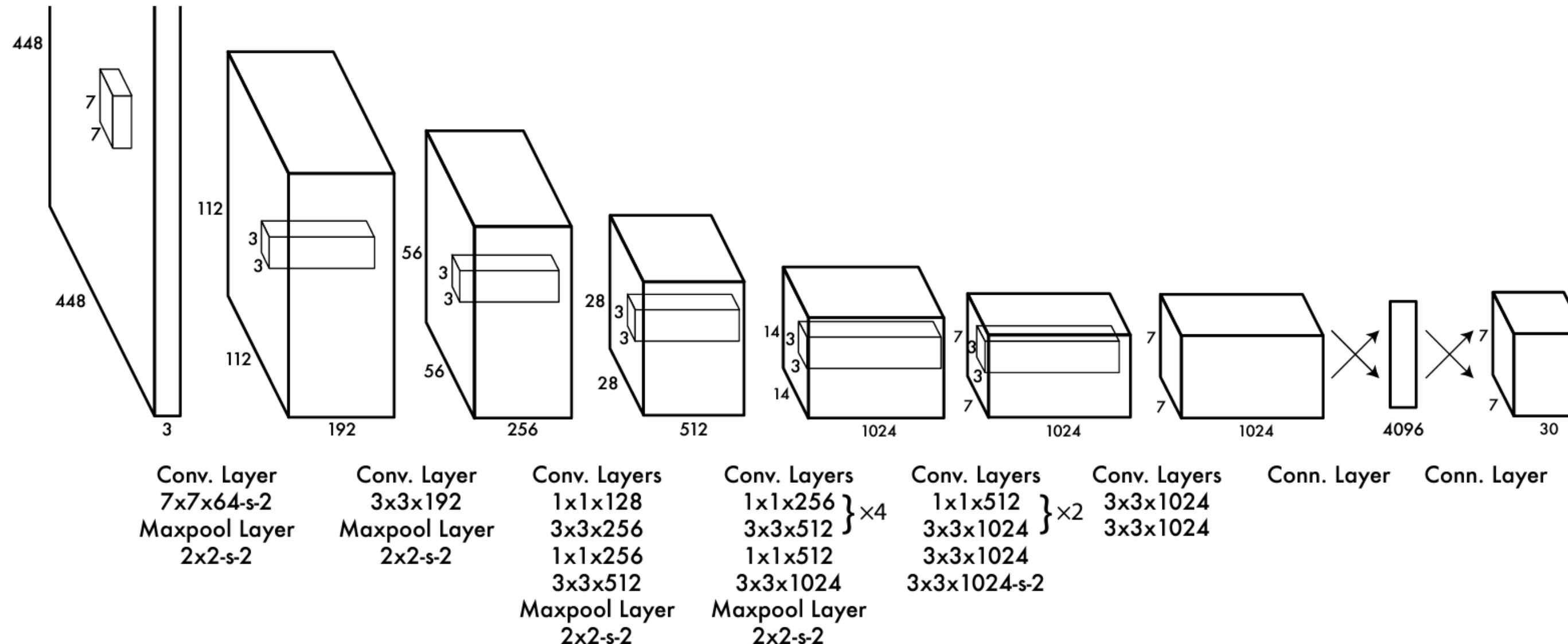
# YOLO: You Only Look Once

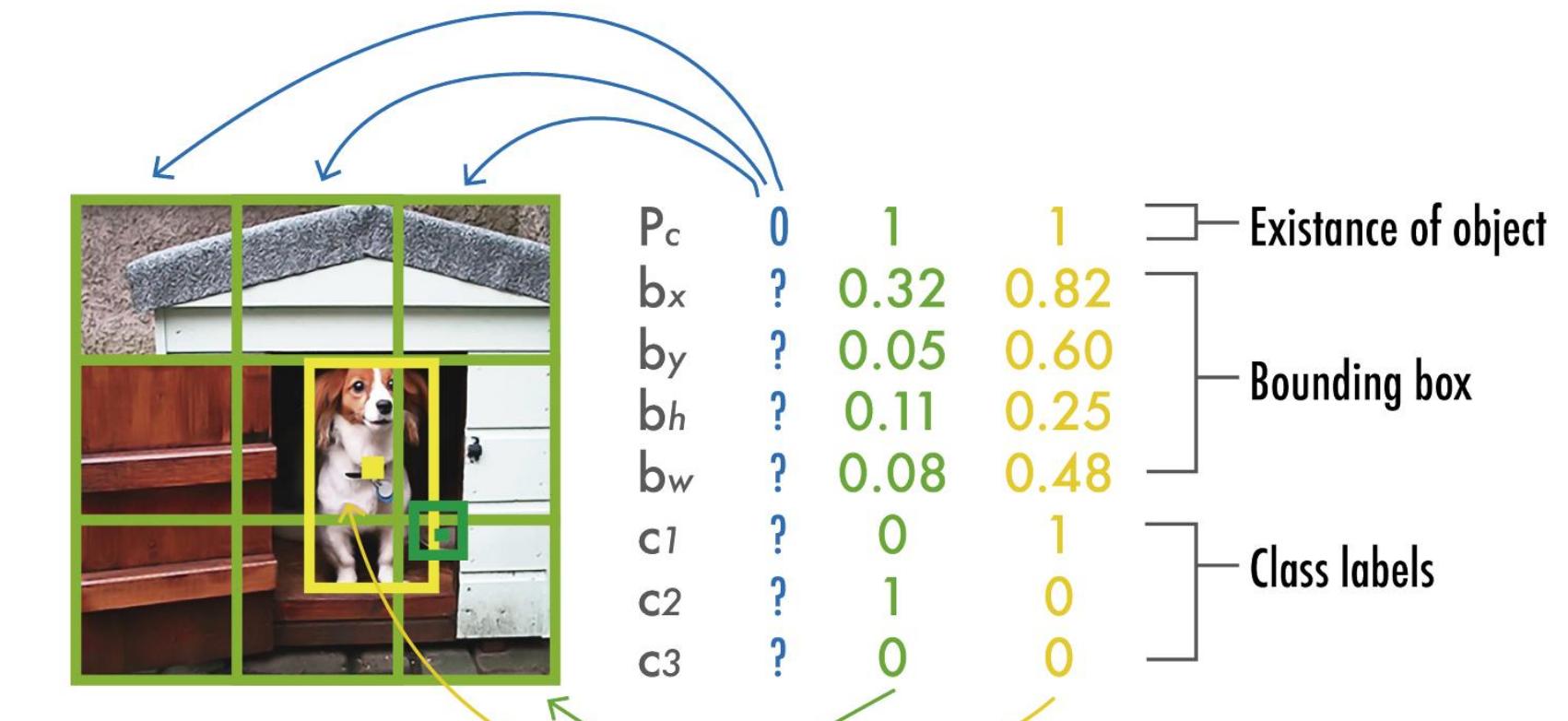
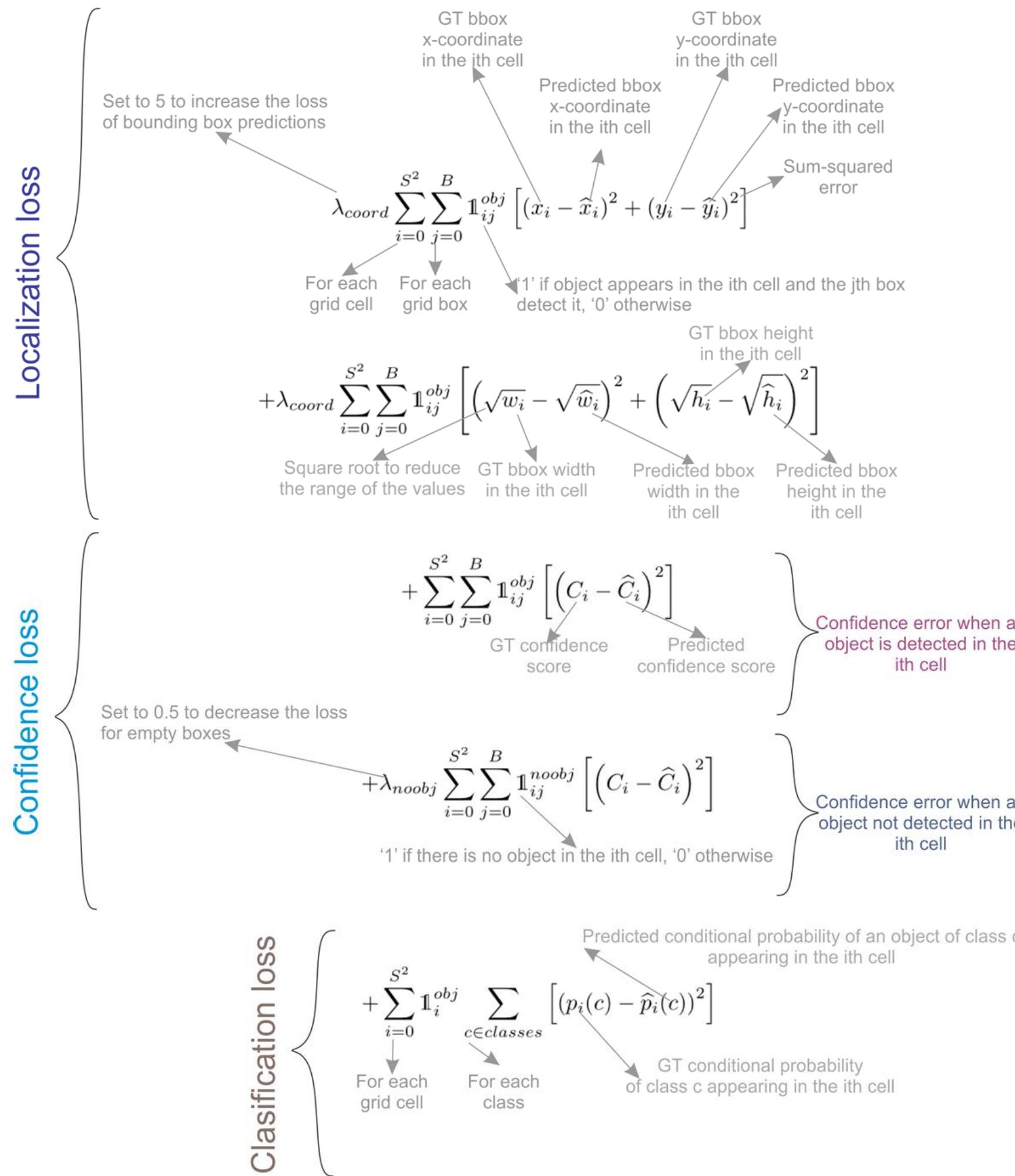


# YOLO: You Only Look Once

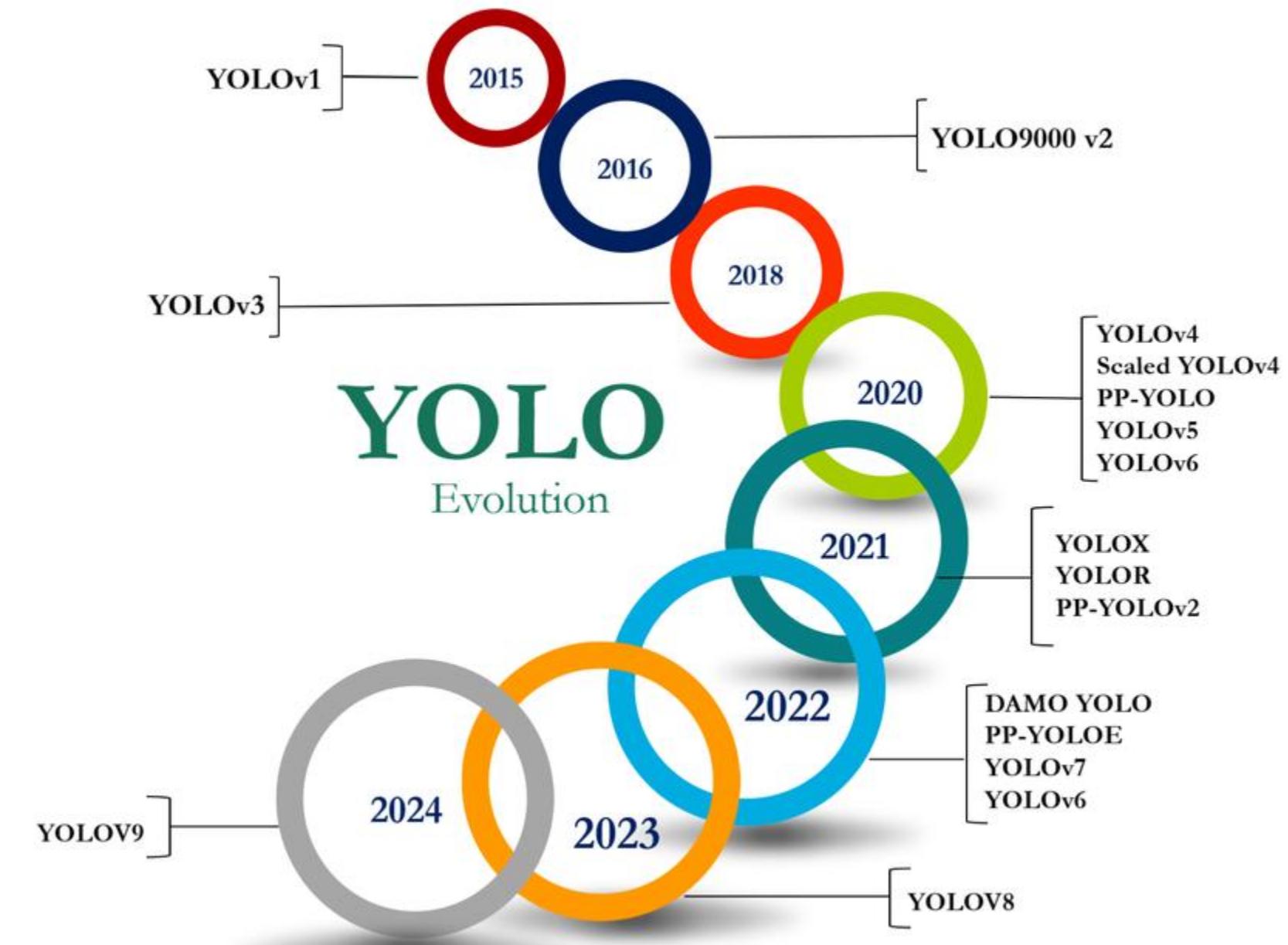


# YOLO: You Only Look Once

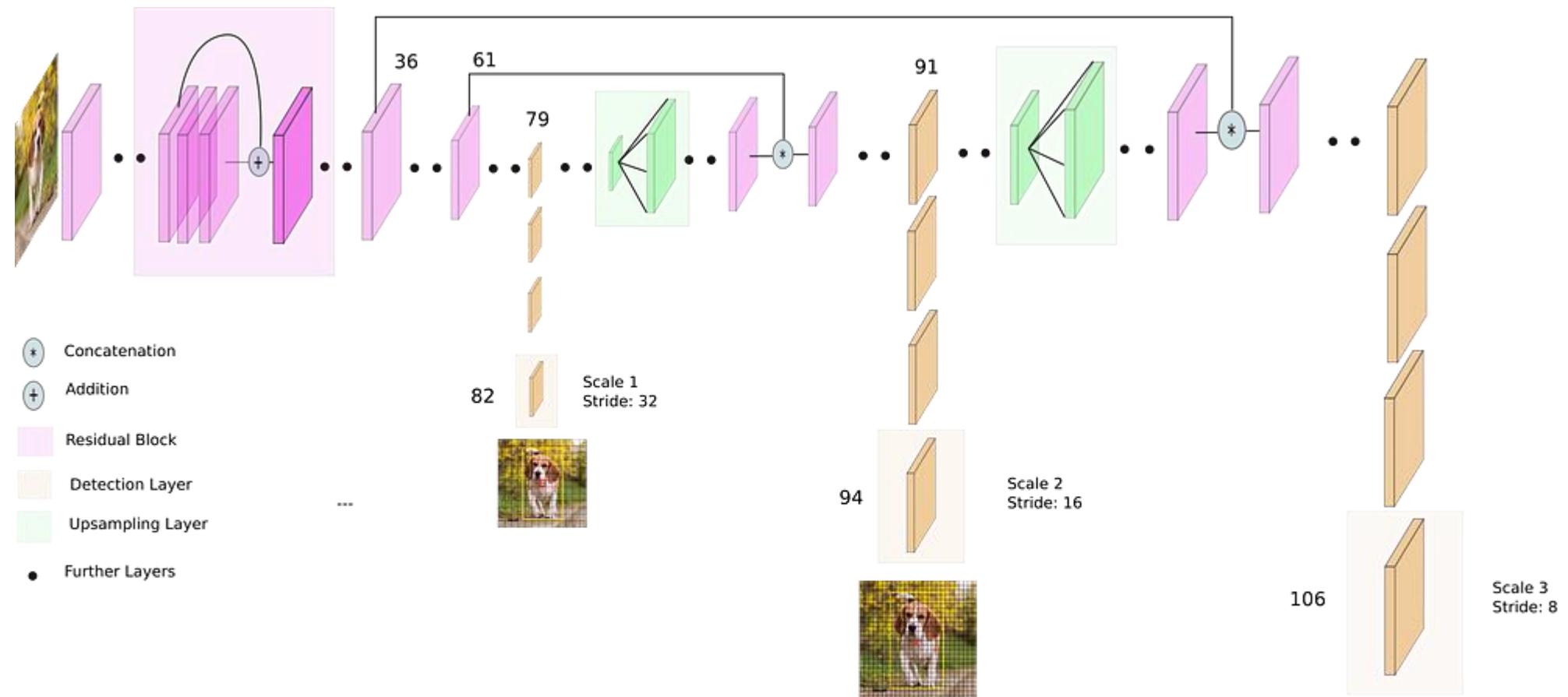




# YOLO: You Only Look Once



# YOLO V3

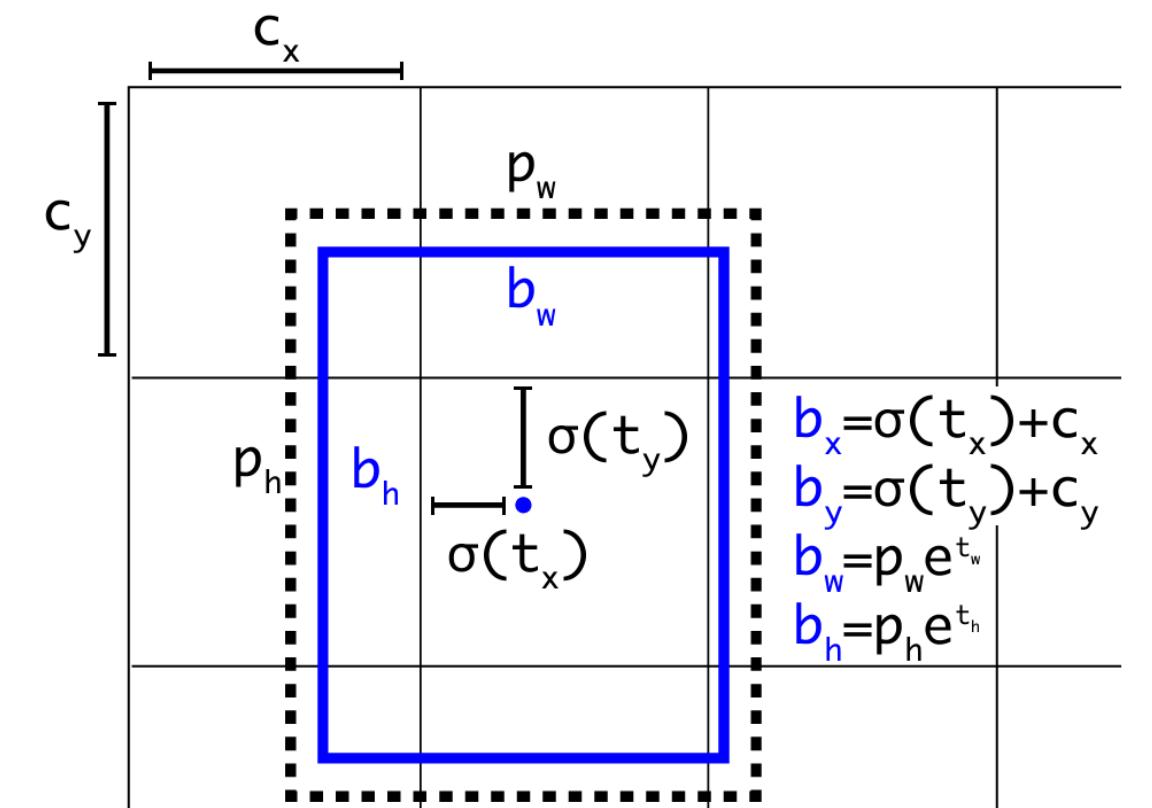
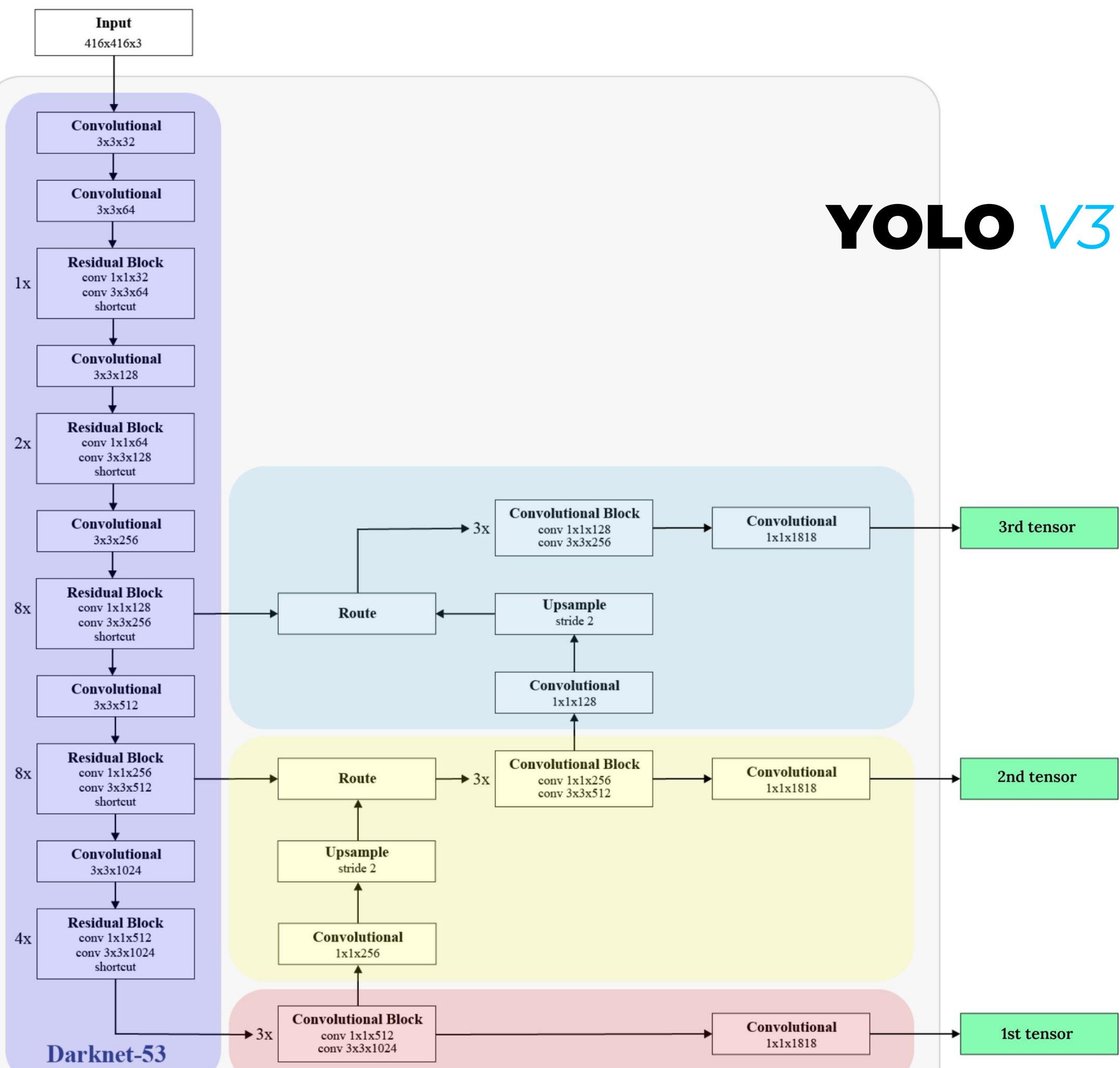


**TRANSFORMATEC**

Joseph Redmon and Ali Farhadi (2018) "YOLOv3: An Incremental Improvement".  
arXiv preprint arXiv:1804.02767

> Reinventa el mundo <

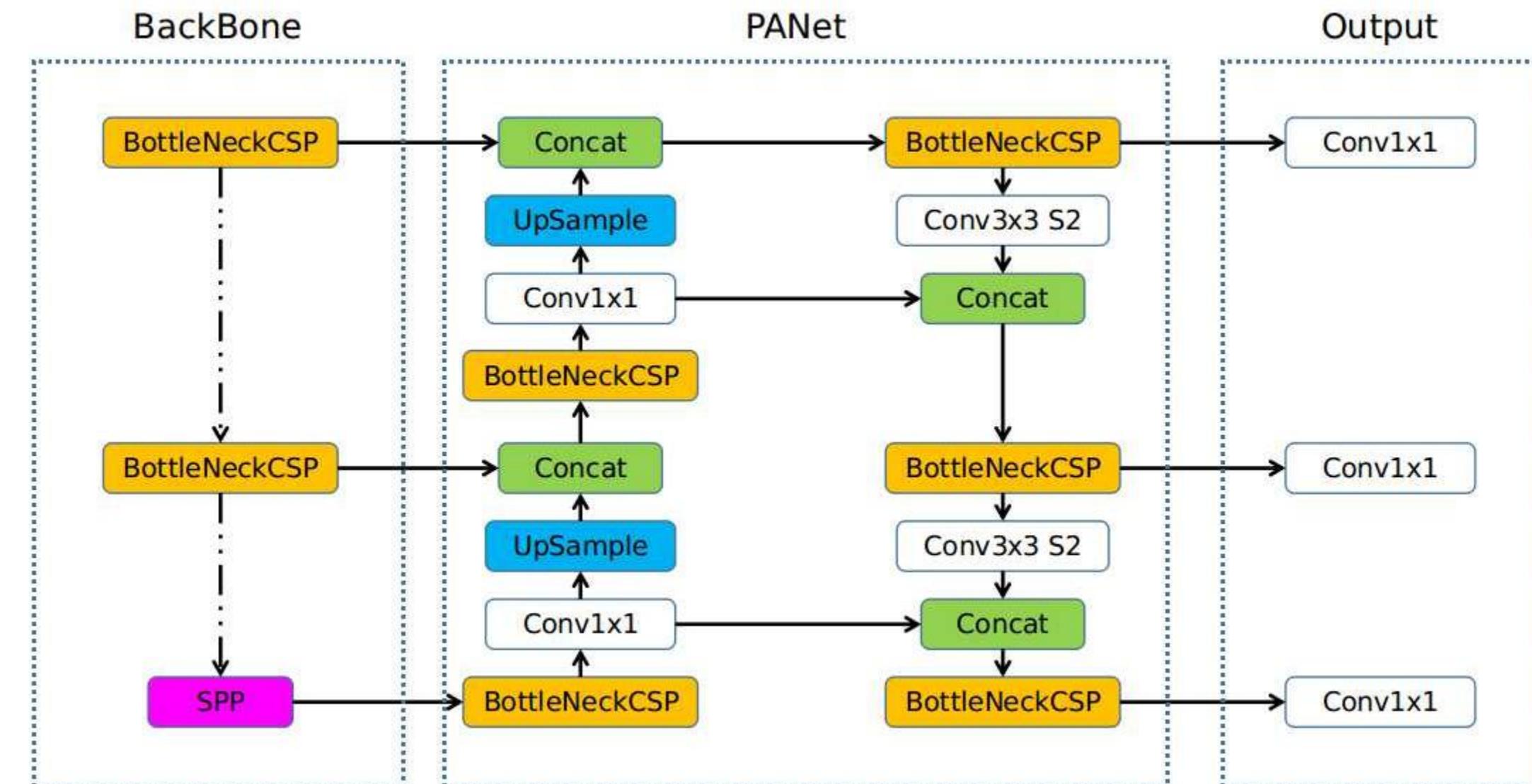
# YOLO V3



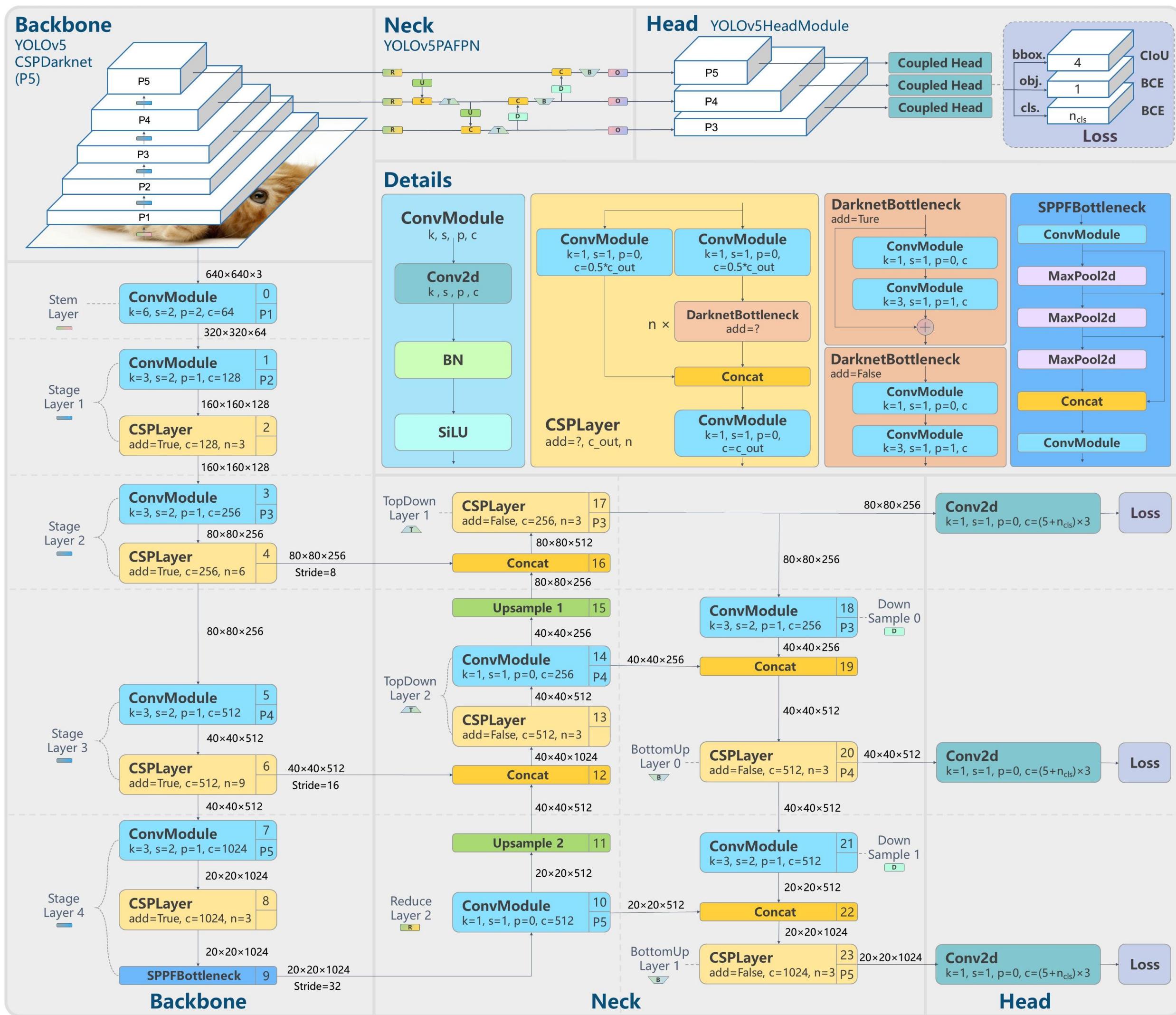
TRANSFORMATEC

Joseph Redmon and Ali Farhadi (2018) "YOLOv3: An Incremental Improvement".  
arXiv preprint arXiv:1804.02767

# YOLO V5



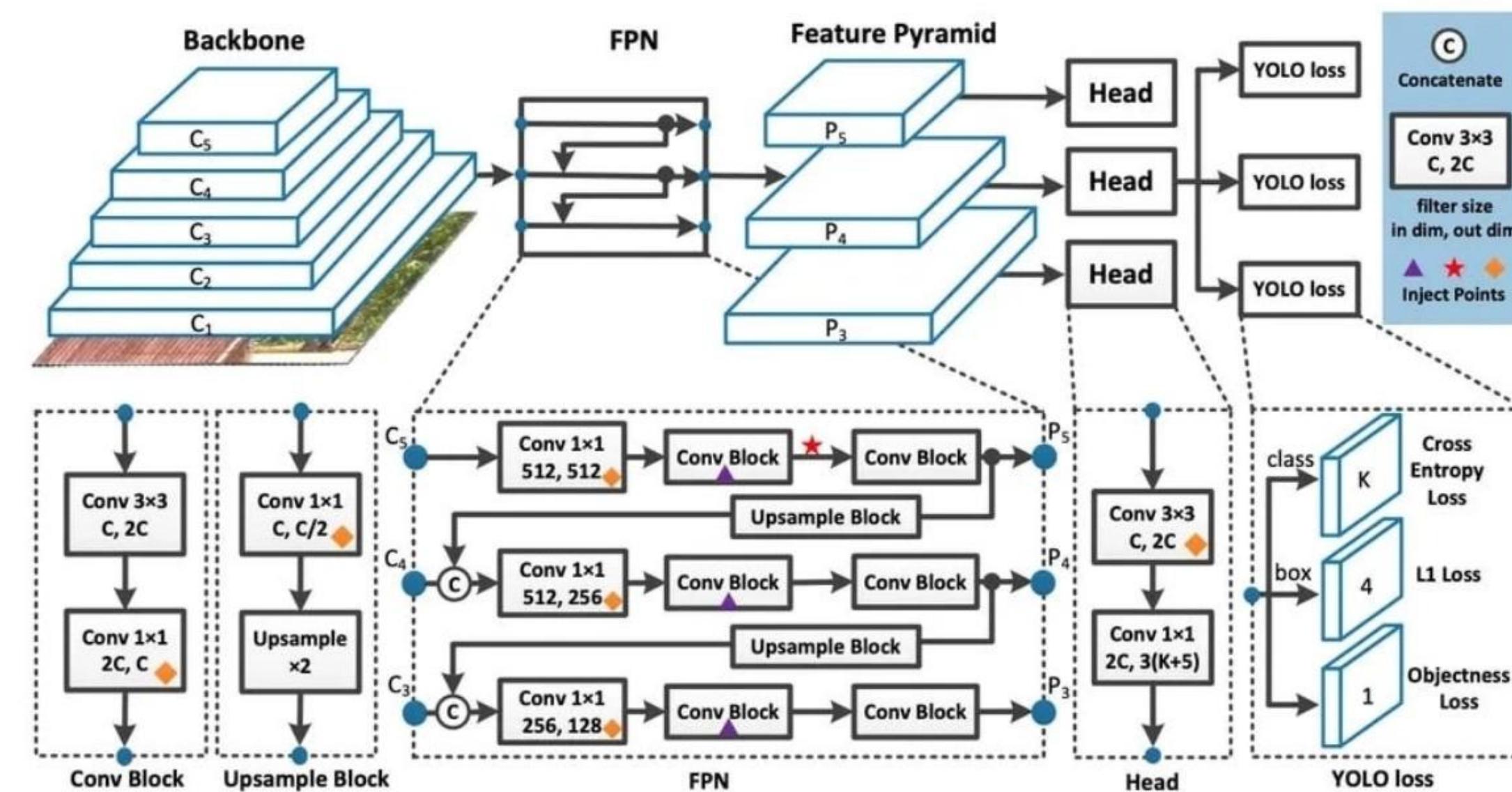
> Reinventa el mundo <

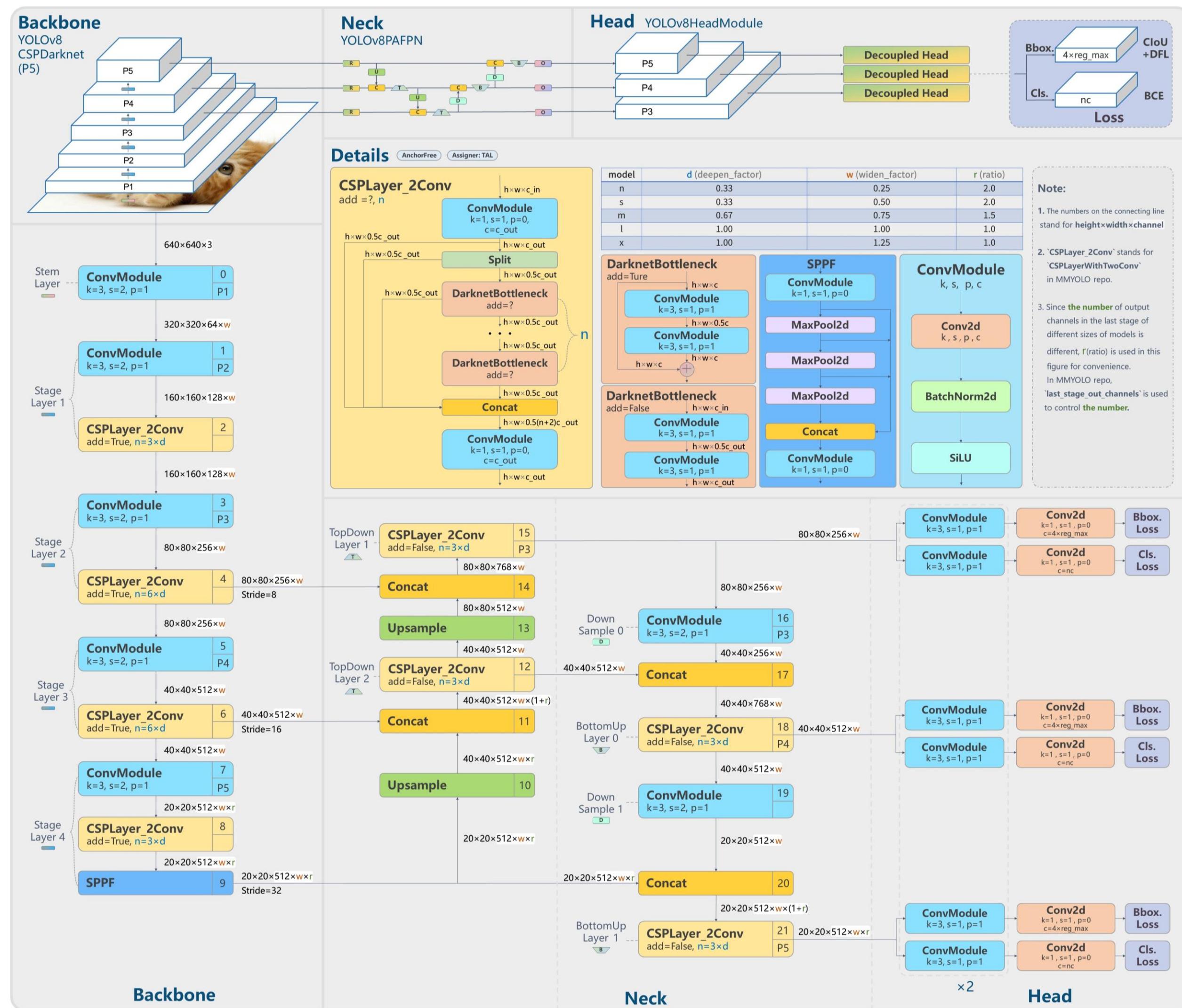


**TRANSFORMATEC**

G. Jocher (2020) "YOLOv5 by Ultralytics".  
<https://github.com/ultralytics/yolov5>

# YOLO V8

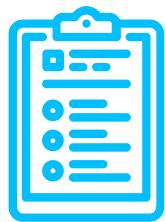




> Reinventa el mundo <

# TRANSFORMA TEC

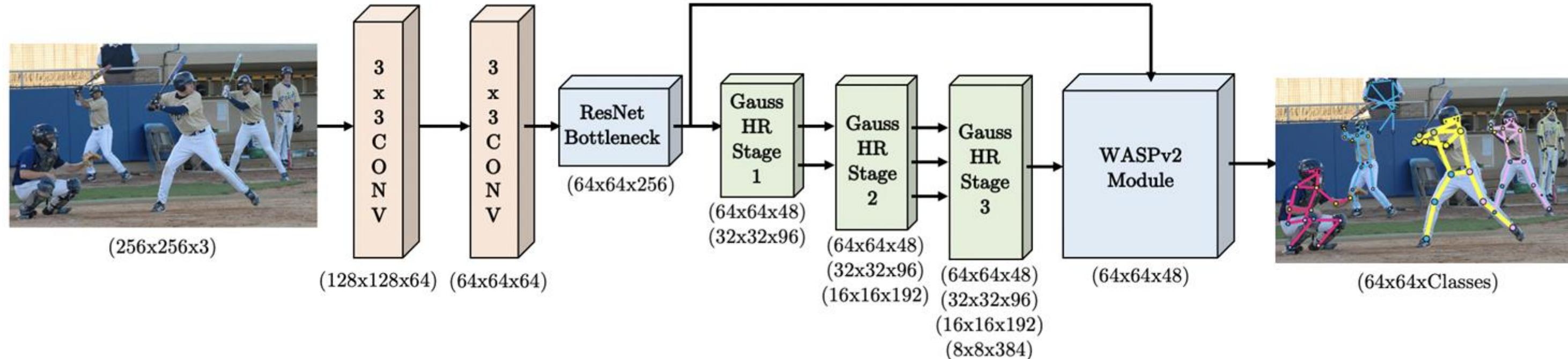
# 2.



## Pose *Estimation*



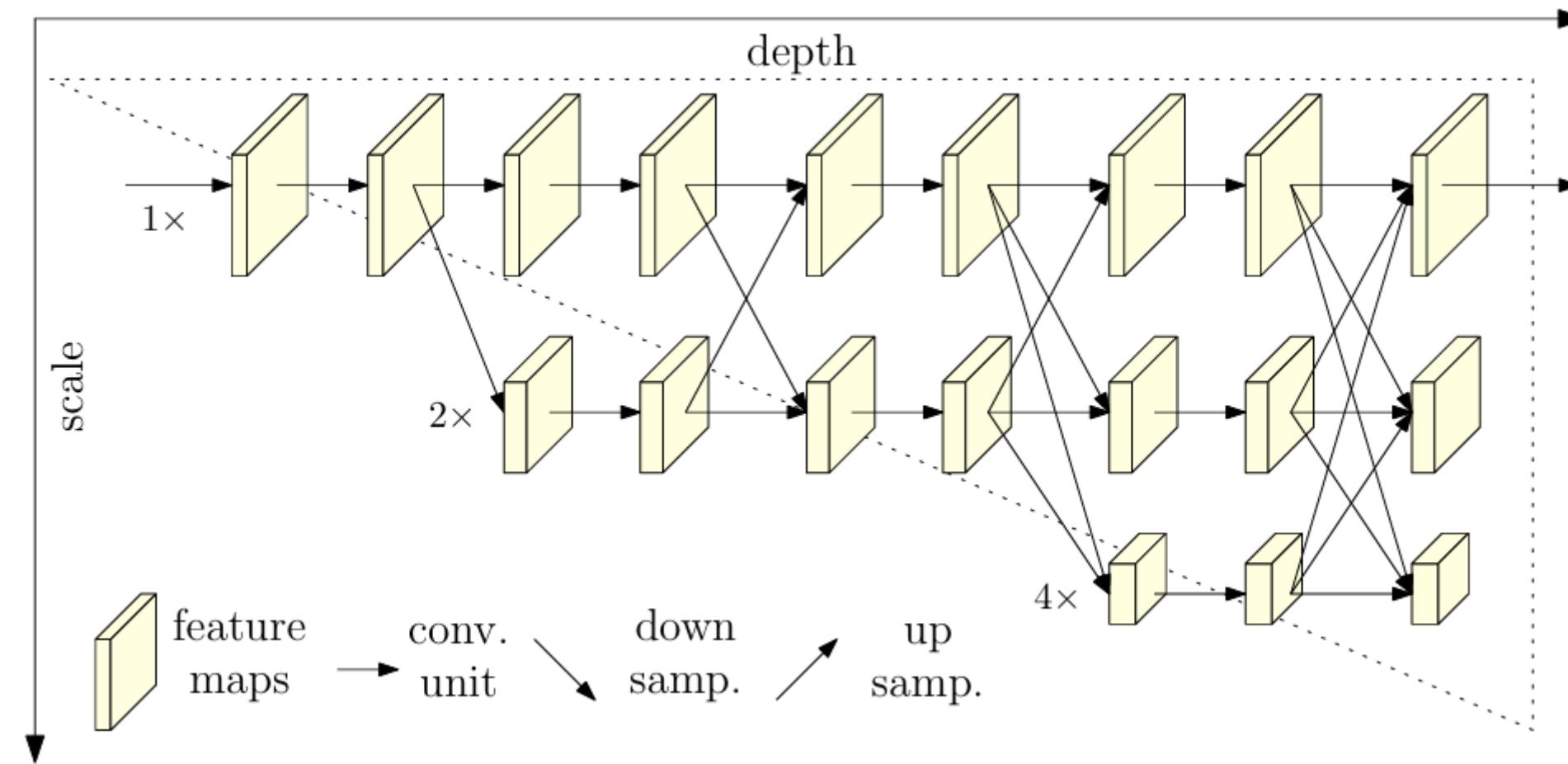
# OmniPose



**TRANSFORMATEC**

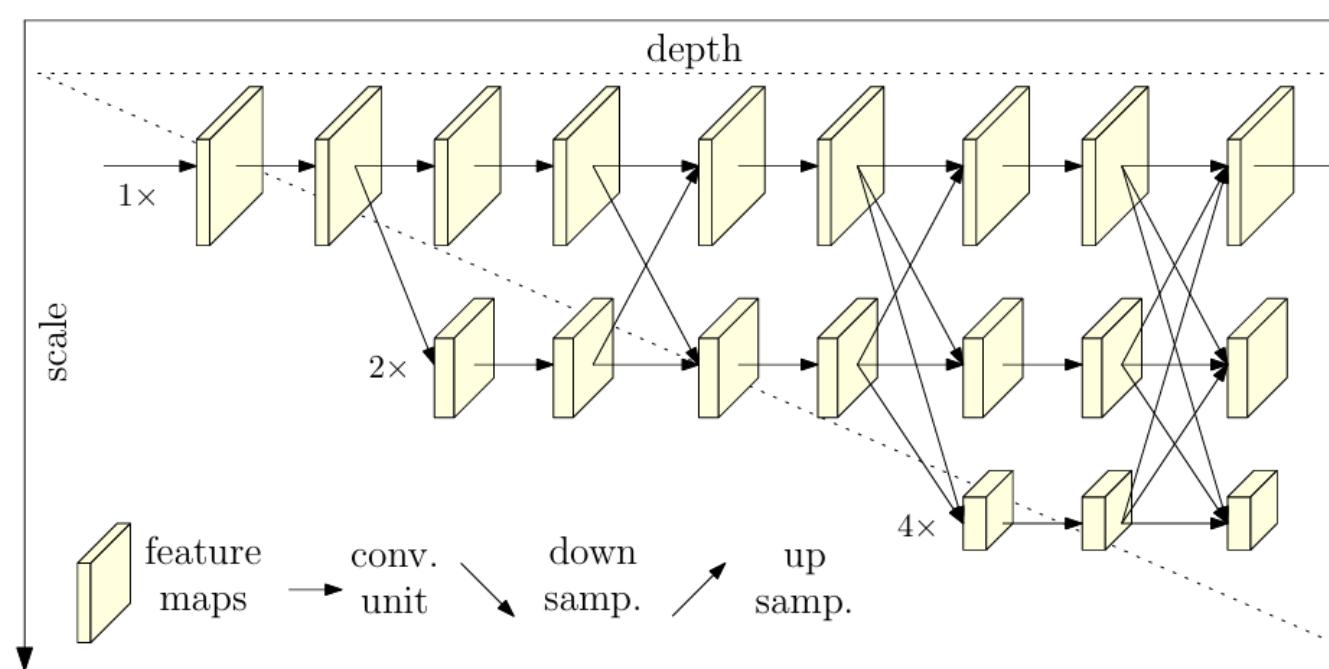
# OmniPose

**HR stage**

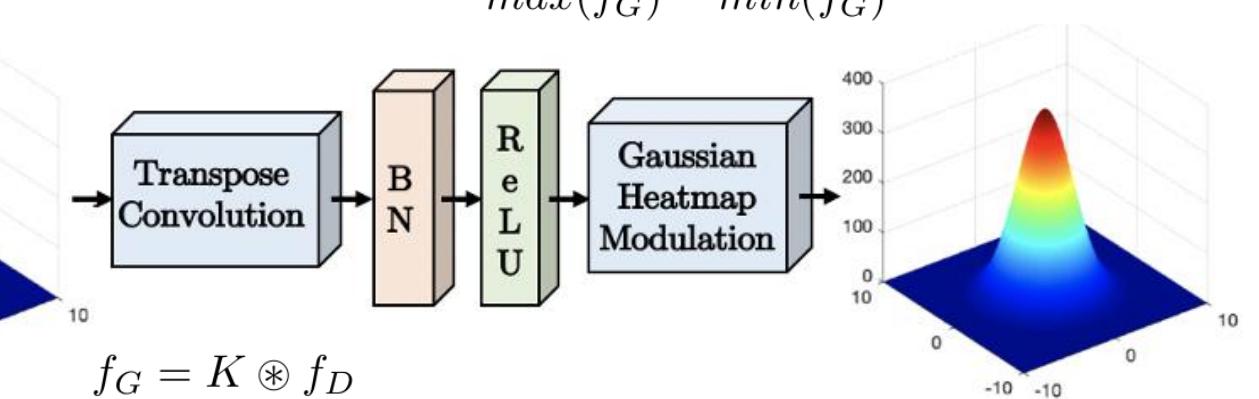


# OmniPose

## Gauss HR stage



$$f_{G_s} = \frac{f_G - \min(f_G)}{\max(f_G) - \min(f_G)} * \max(f_D)$$



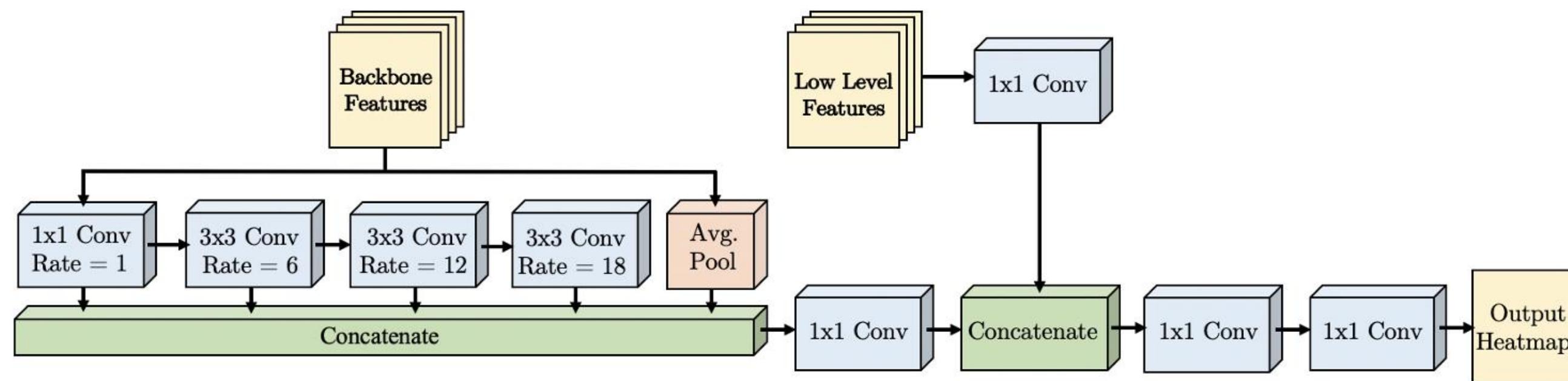
$$f_G = K \circledast f_D$$



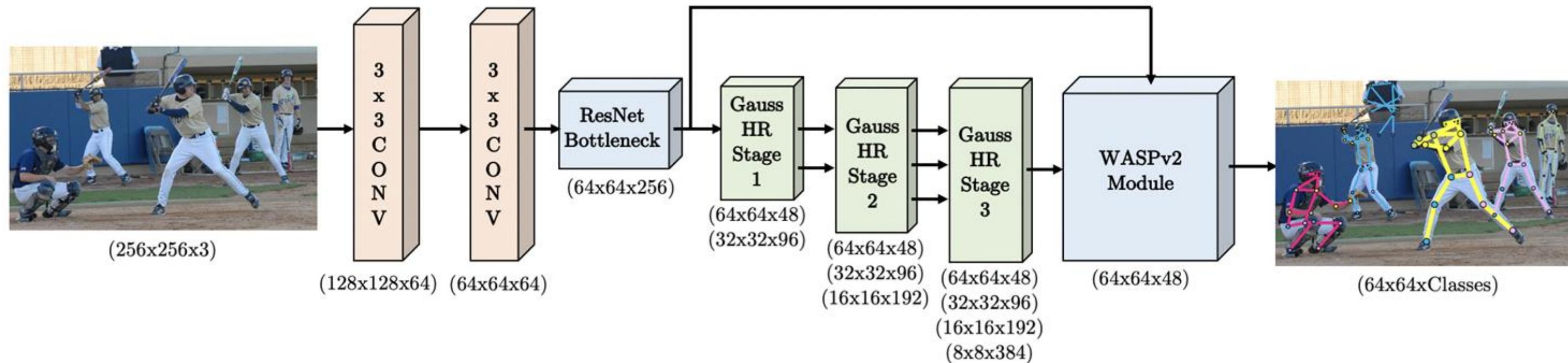
**TRANSFORMATEC**

# OmniPose

## WASPv2 module



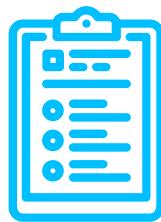
# OmniPose



**TRANSFORMATEC**

Bruno Artacho et al. (2021) "OmniPose: A Multi-Scale Framework for Multi-Person Pose Estimation".  
arXiv preprint arXiv:2103.10180.

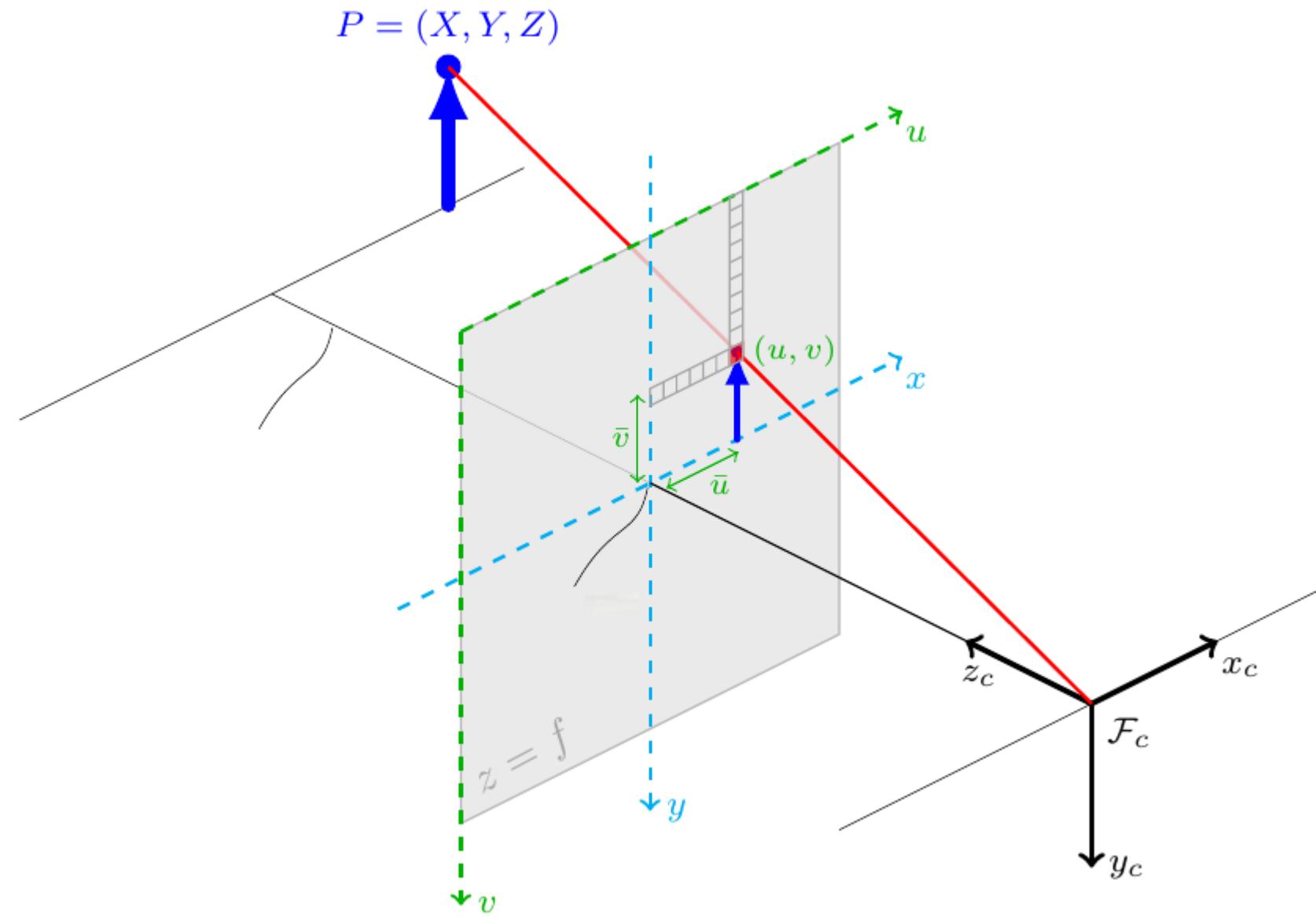
**3.**



## **Depth** *Estimation*



# Modelado de la Proyección en Cámaras



**Pinhole camera model**

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

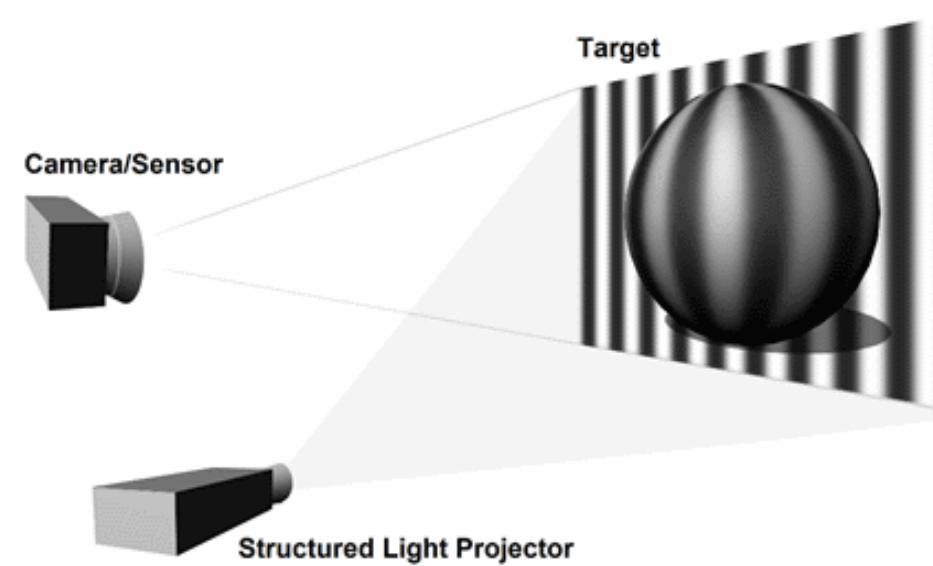
donde:

- $(X, Y, Z)$  son las coordenadas 3D del punto en el espacio.
- $(u, v)$  son las coordenadas proyectadas en la imagen.
- $K$  es la matriz intrínseca de la cámara, que contiene la distancia focal  $f$  y el centro óptico.
- $R$  y  $t$  representan la rotación y traslación de la cámara con respecto al mundo.
- $s$  es un factor de escala.

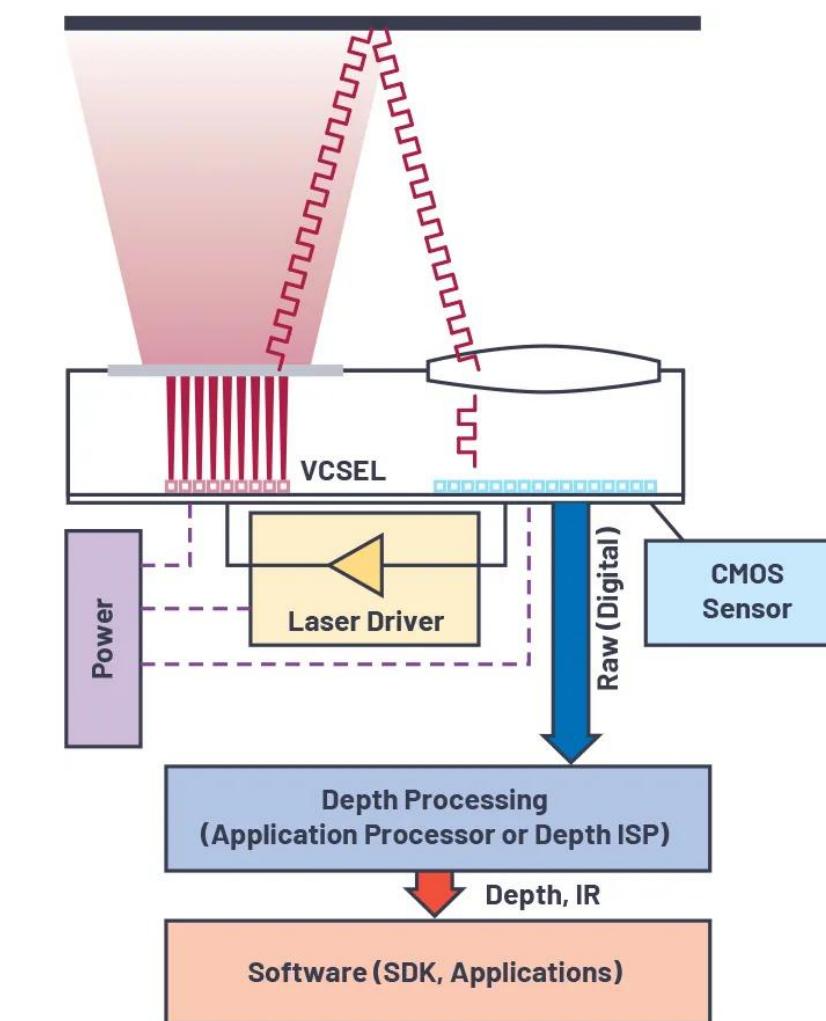


# Depth *images*

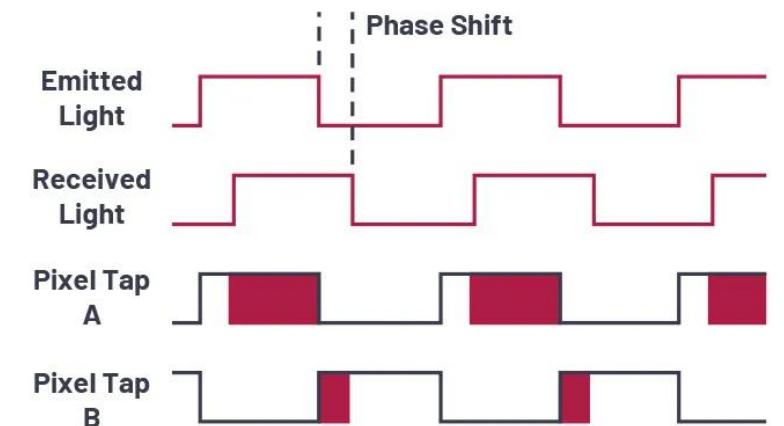
## Structured Light



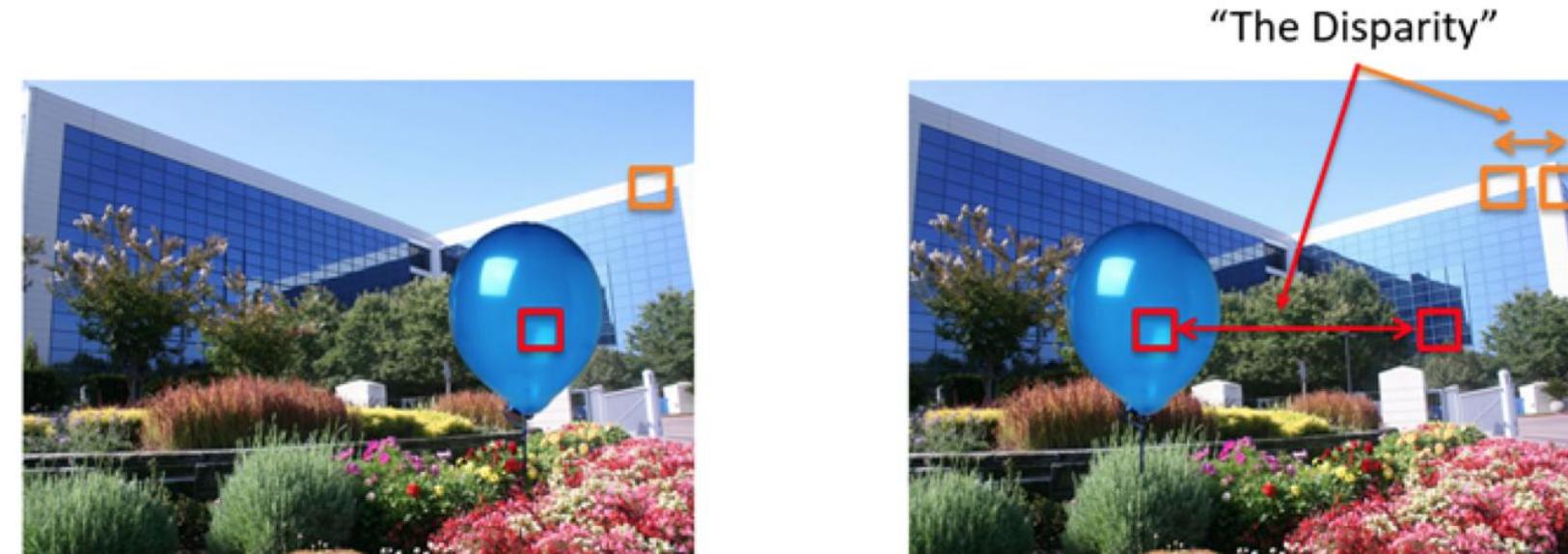
## Time-of-flight (ToF)



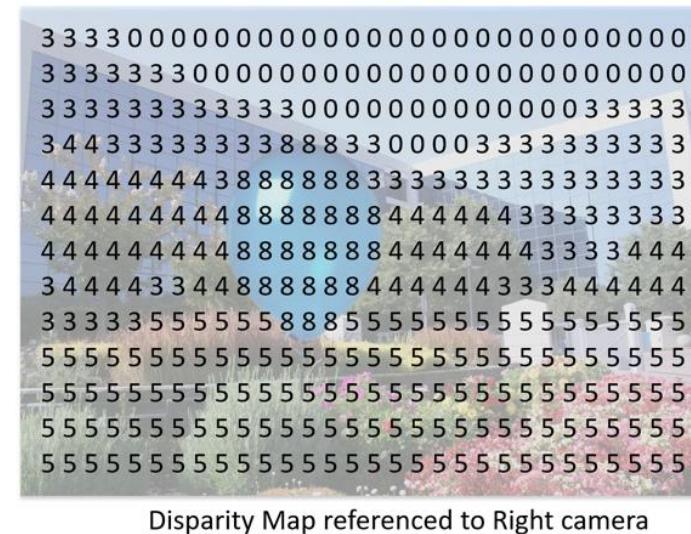
## Sensor infrarrojo



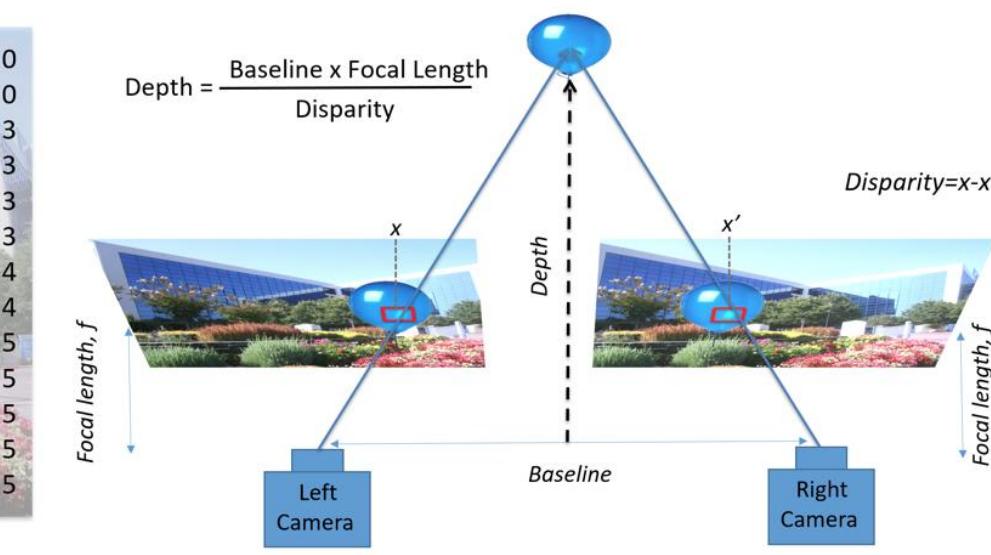
# Passive Stereo



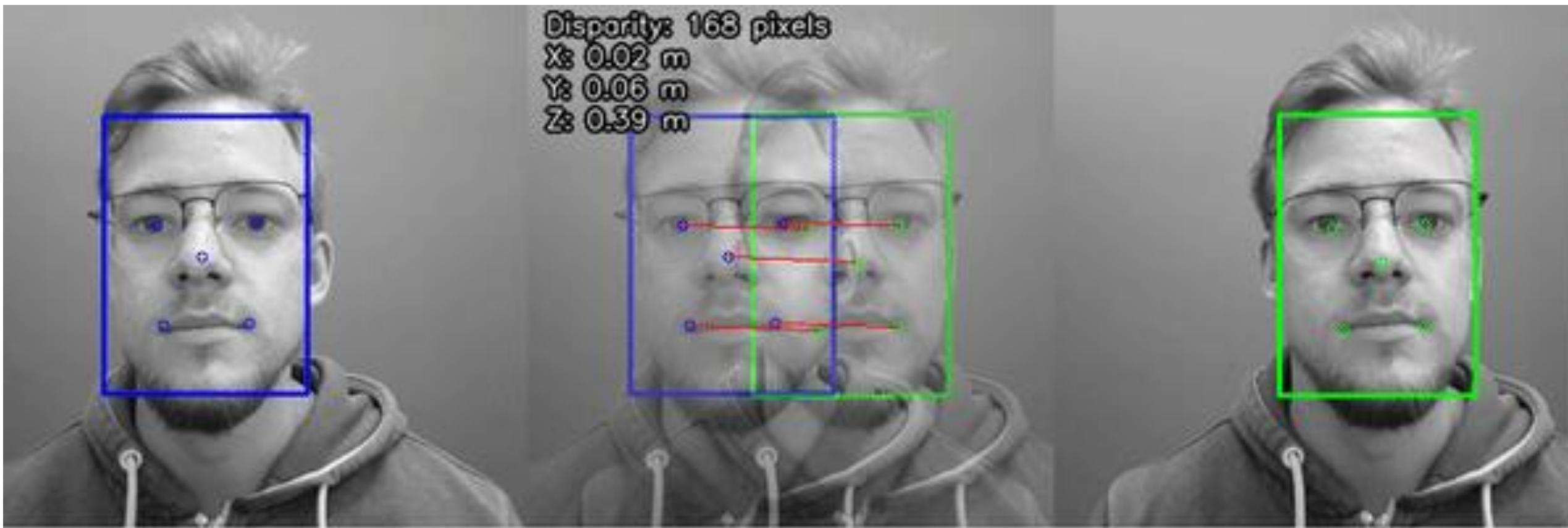
Left camera



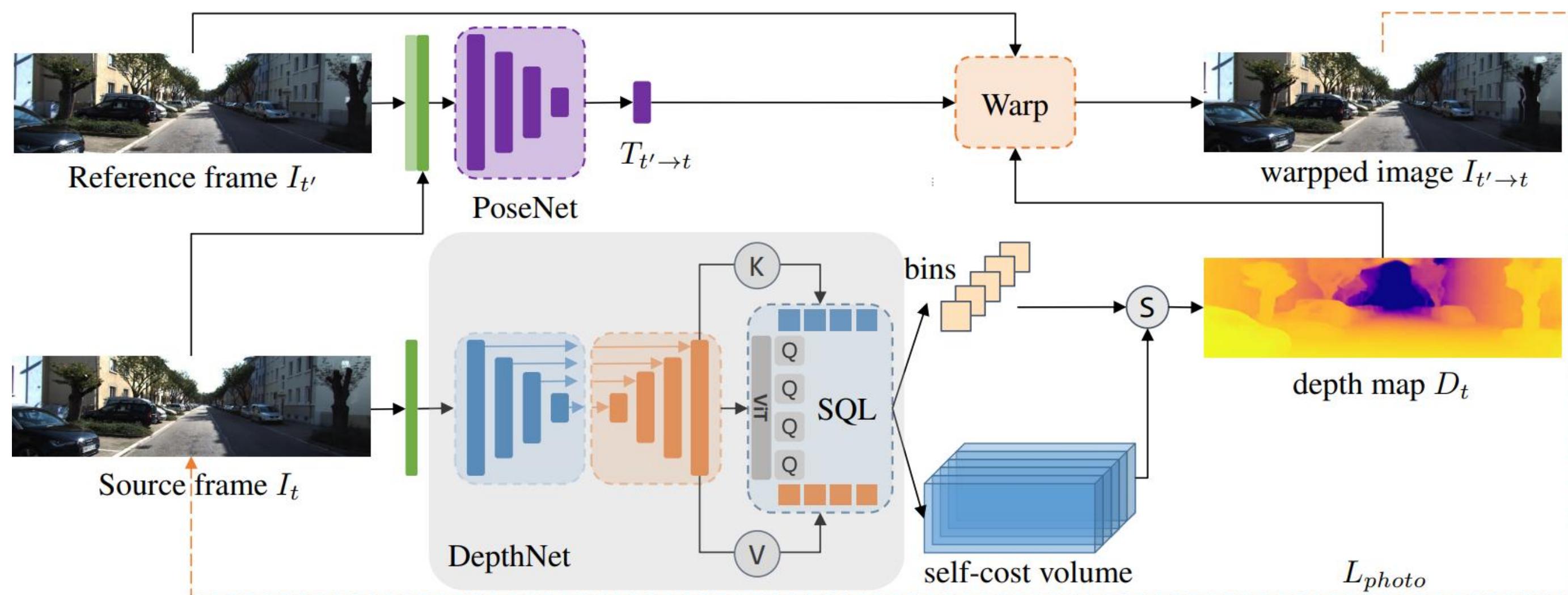
Right camera



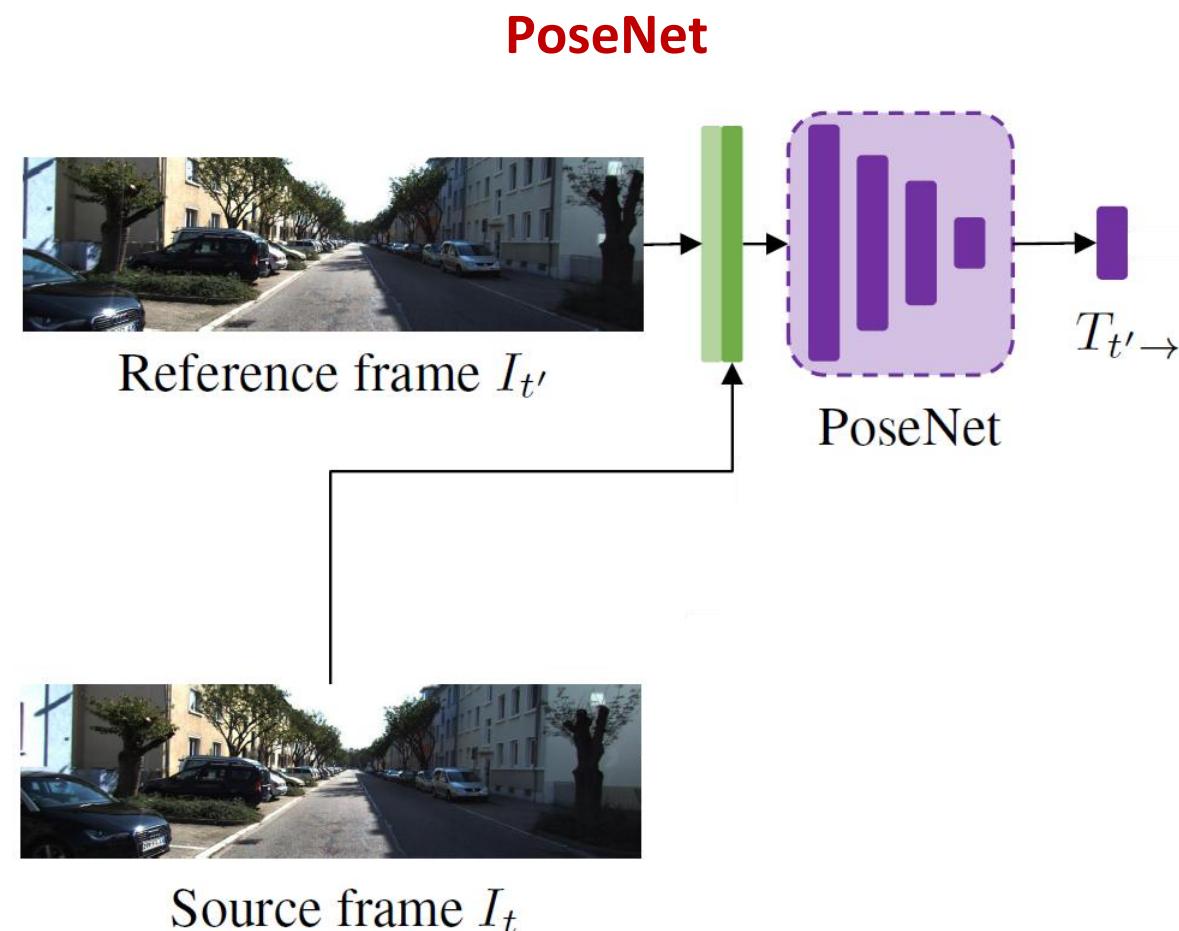
# Passive Stereo



# SQLdepth



# SQL<sup>depth</sup>



PoseNet sigue la arquitectura típica de una red de regresión convolucional y su objetivo es estimar la matriz de transformación

$$T_{t' \rightarrow t} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$$

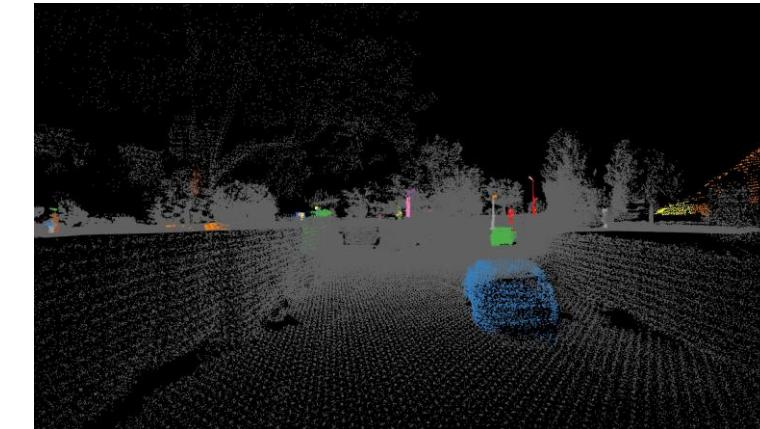


# SQL<sup>depth</sup>

Geometric warp



Source frame  $I_t$



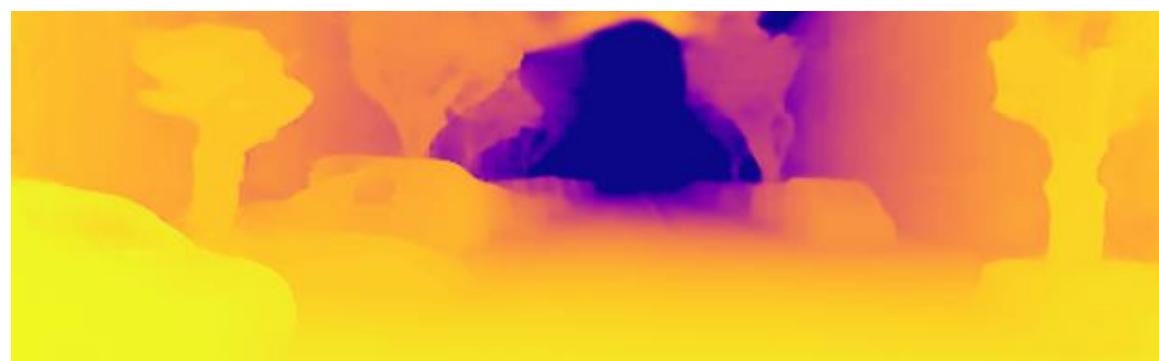
$$X_t = K^{-1} \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \cdot D_t(u, v)$$

$$X_{t'} = T_{t' \rightarrow t} \cdot X_t$$

$$\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = K \cdot X_{t'}$$



Source frame  $I_{t'}$

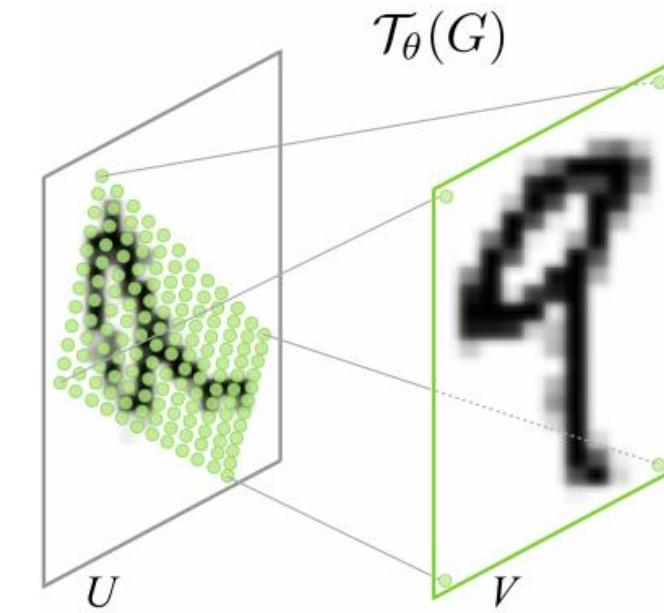
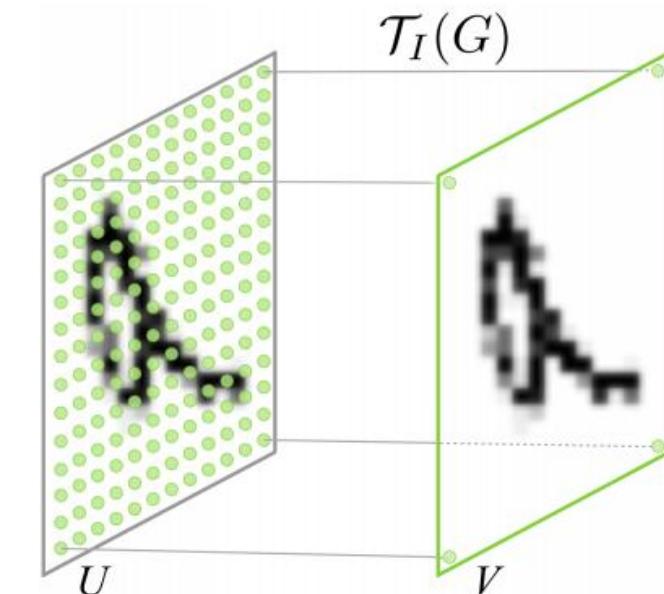
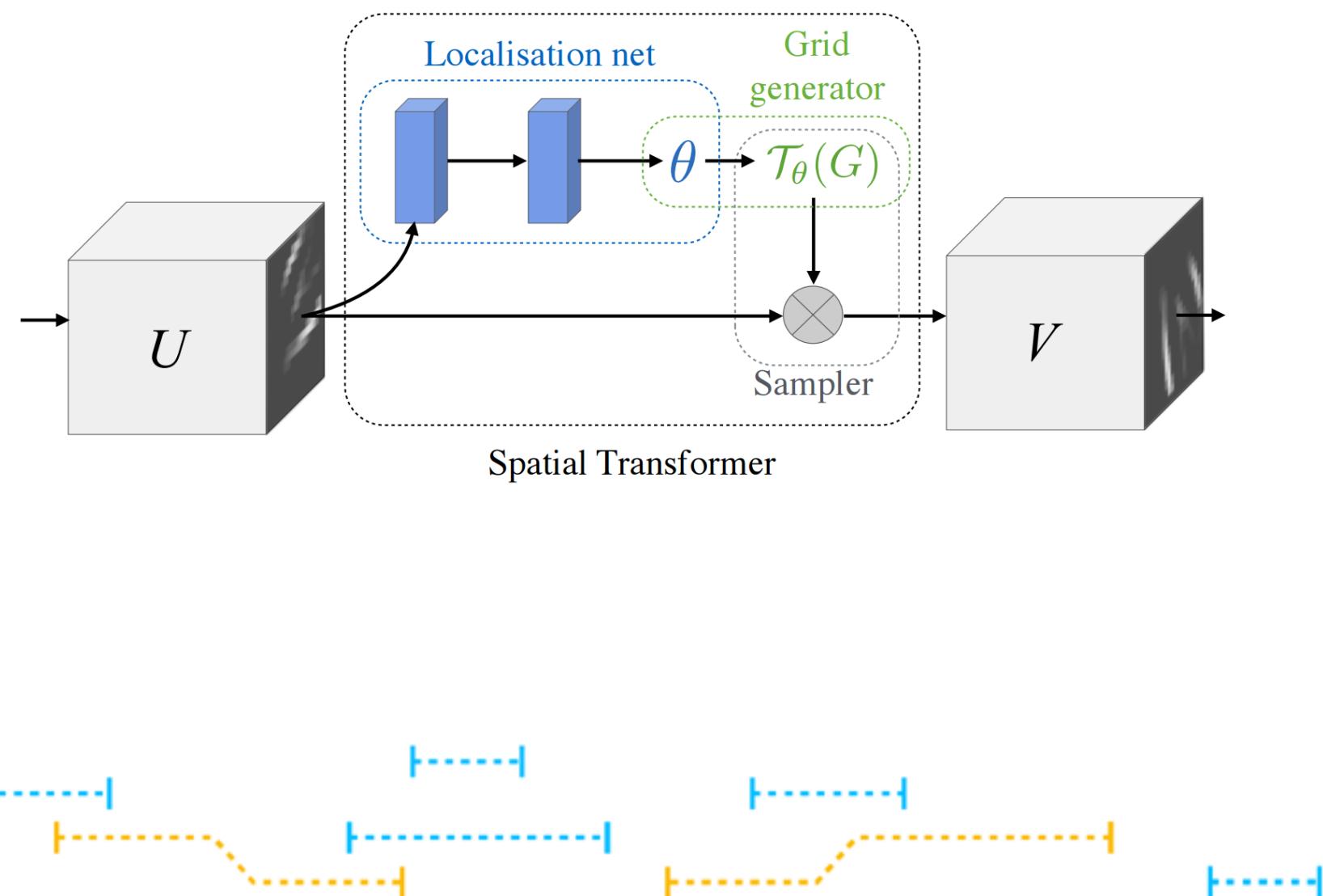


Depth image  $D_t$



# SQL<sup>depth</sup>

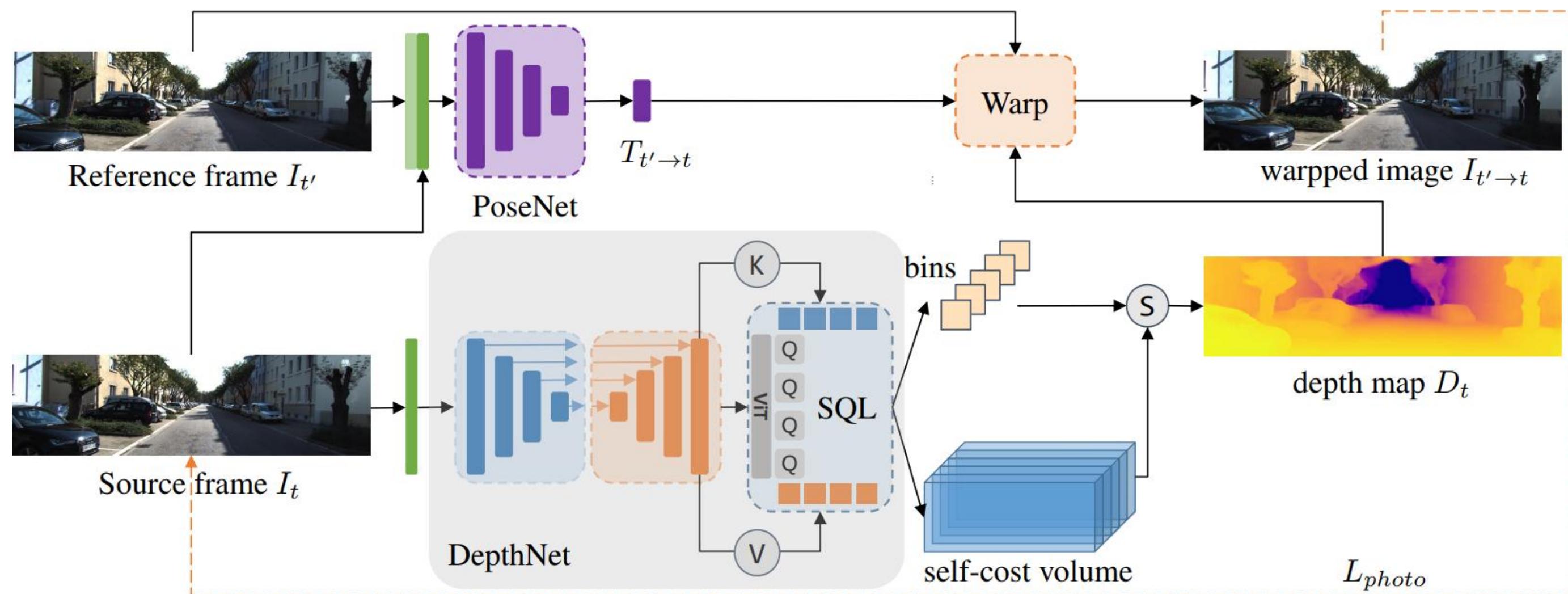
## Geometric warp



**TRANSFORMATEC**

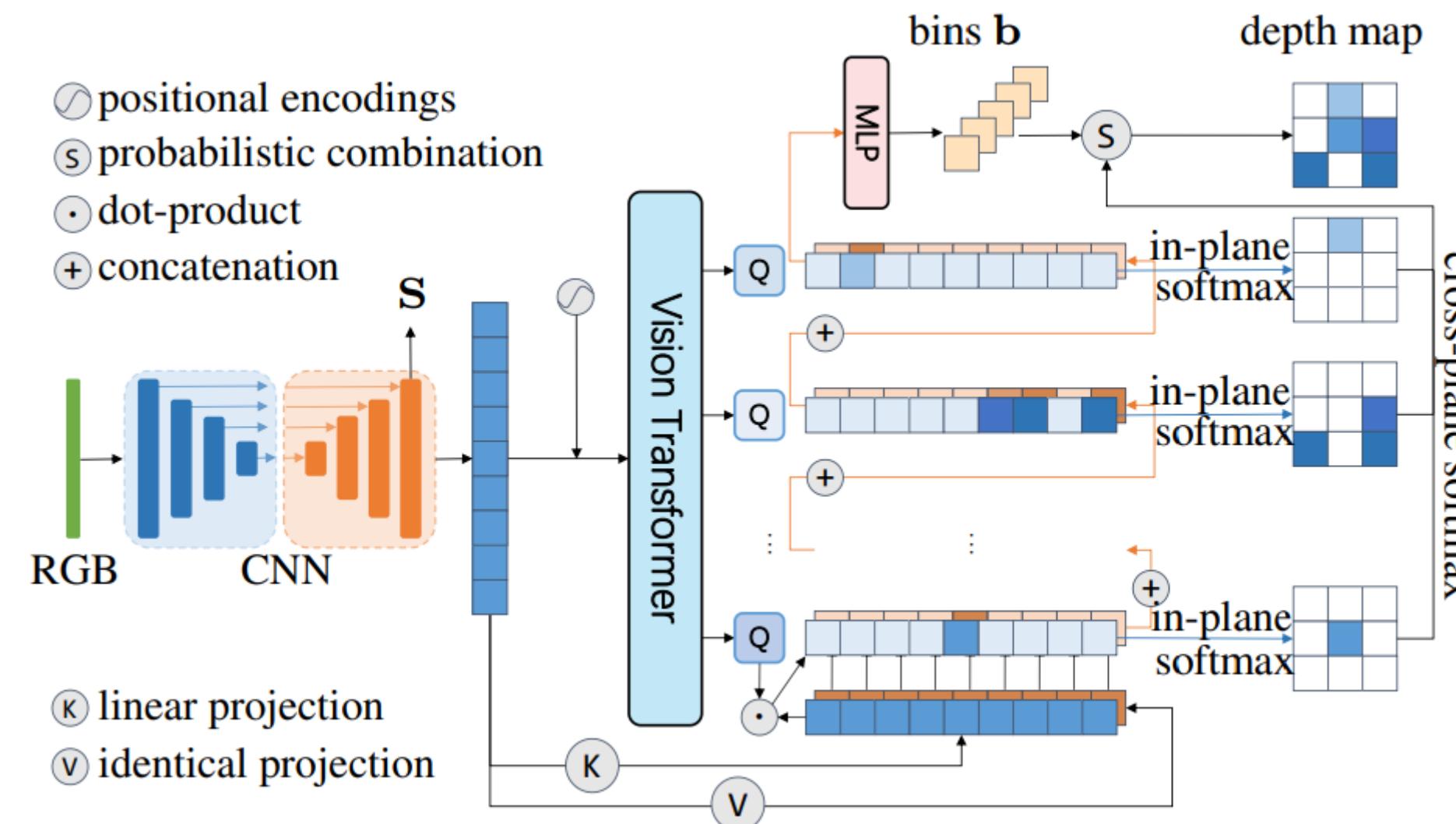
Max Jaderberg et al. (2015) "Spatial Transformer Networks".  
Advances in neural information processing systems, 28.

# SQLdepth



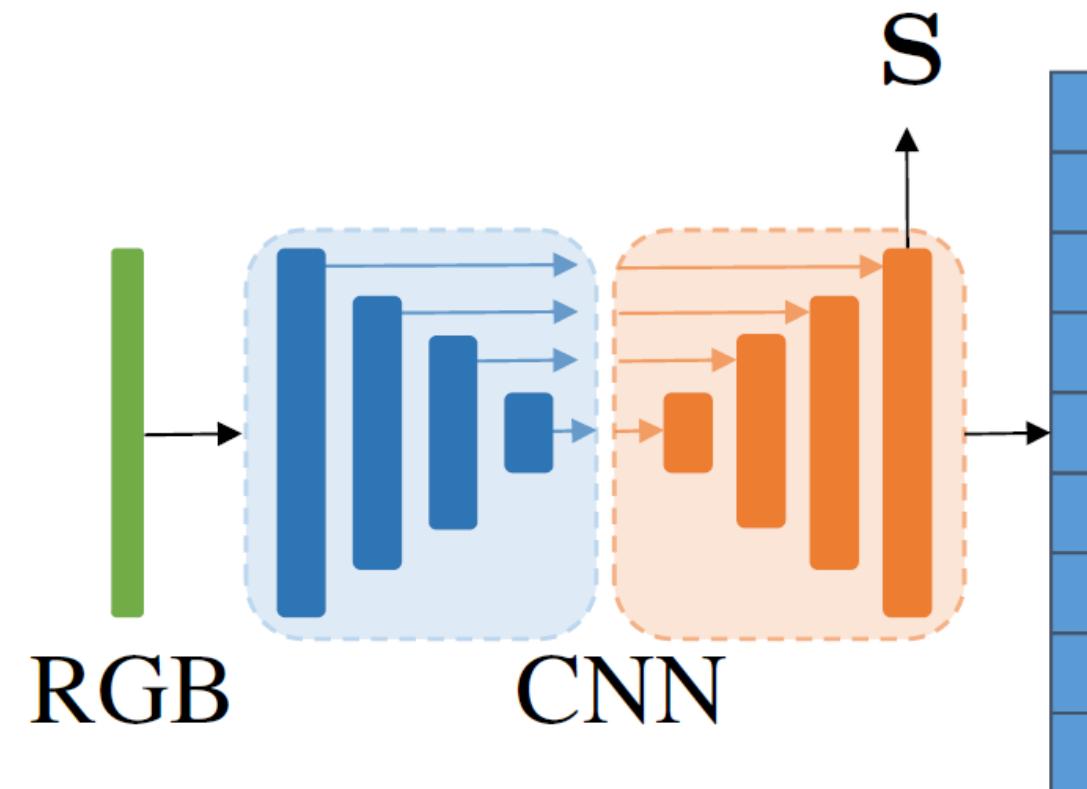
# SQL<sup>depth</sup>

## DepthNet with the Self Query Layer



# SQL<sup>depth</sup>

## U-net



### Input:

Imagen RGB  $I \in \mathbb{R}^{3 \times H \times W}$ .

### Patche:

Feature map  $F \in \mathbb{R}^{C \times \frac{h}{p} \times \frac{w}{p}}$ , donde  $p$  es el tamaño del patch.

ViT input  $F \in \mathbb{R}^{N \times C}$ , donde  $N = \frac{h}{p} \times \frac{w}{p}$ .

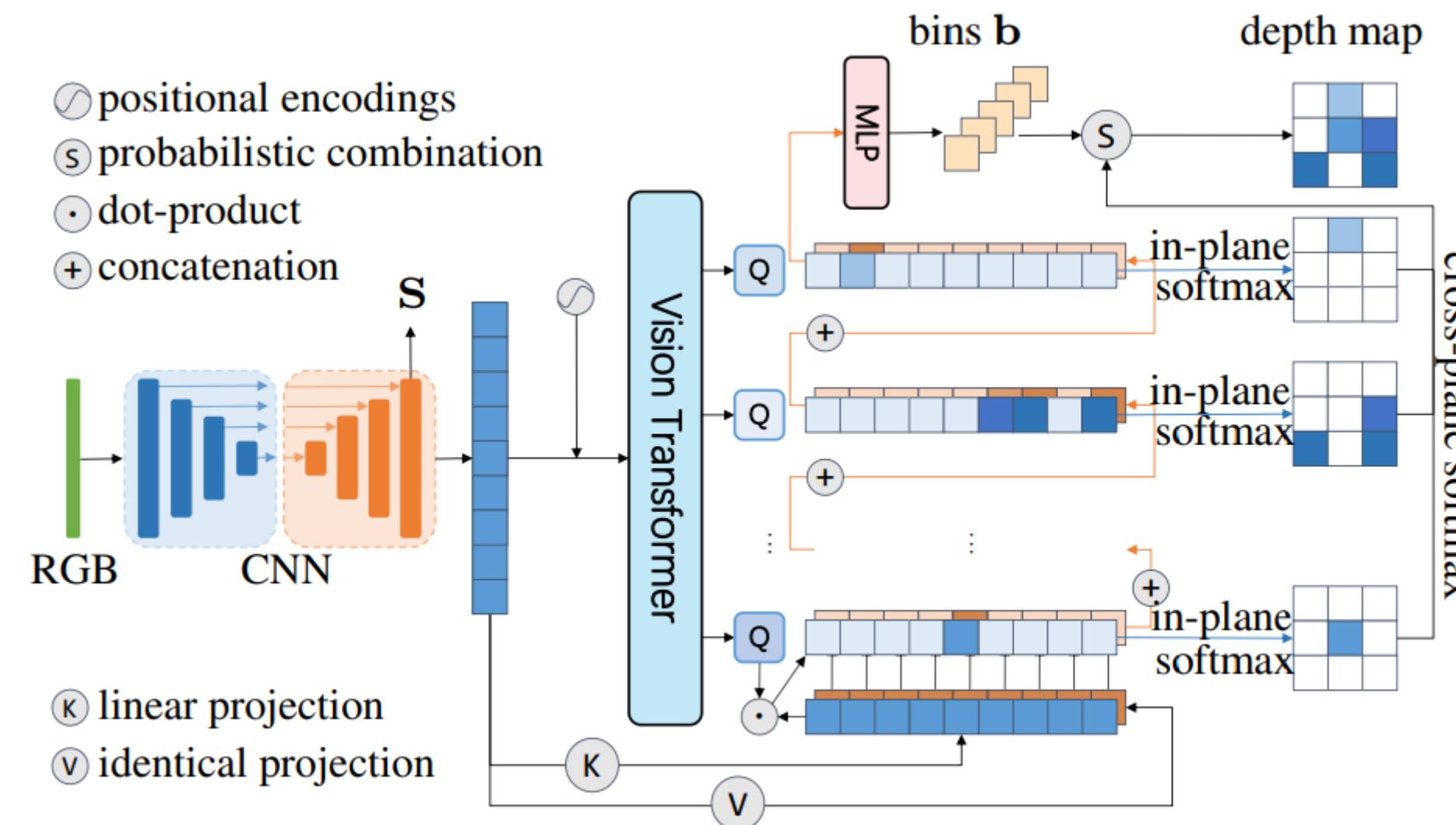
### Queries:

Feature map  $Q \in \mathbb{R}^{C \times Q}$ , el cual abstrae estructuras espaciales importantes.



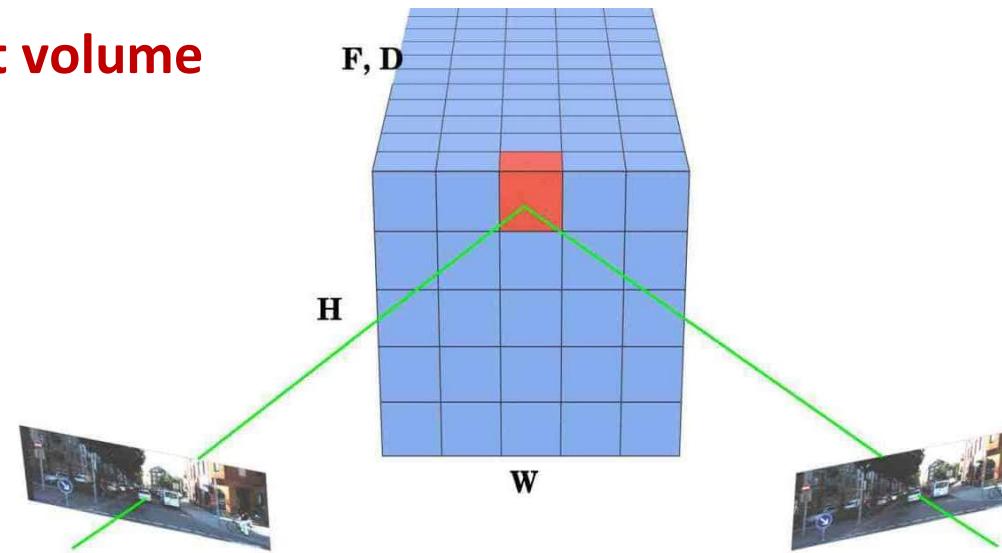
# SQL<sup>depth</sup>

## DepthNet with the Self Query Layer



# Self-Query Layer

**Cost volume**



Para cada píxel en  $I_L$ , buscamos su posible correspondencia en  $I_R$  desplazándolo en diferentes niveles  $d$ .

$$V_{cost}(x, y, d) = \text{Similitud}(F_L(x, y), F_R(x - d, y))$$

donde:

- $F_L, F_R$  son los feature maps de las imágenes.
- $d$  es el desplazamiento (disparidad en visión estéreo).
- $V_{cost} \in R^{H \times W \times D}$ , cada posición almacena una similitud entre píxeles de ambas imágenes.



**Self-cost volume**

En lugar de comparar píxeles entre dos imágenes, se compara cada píxel con representaciones globales de proto-objetos en la imagen.

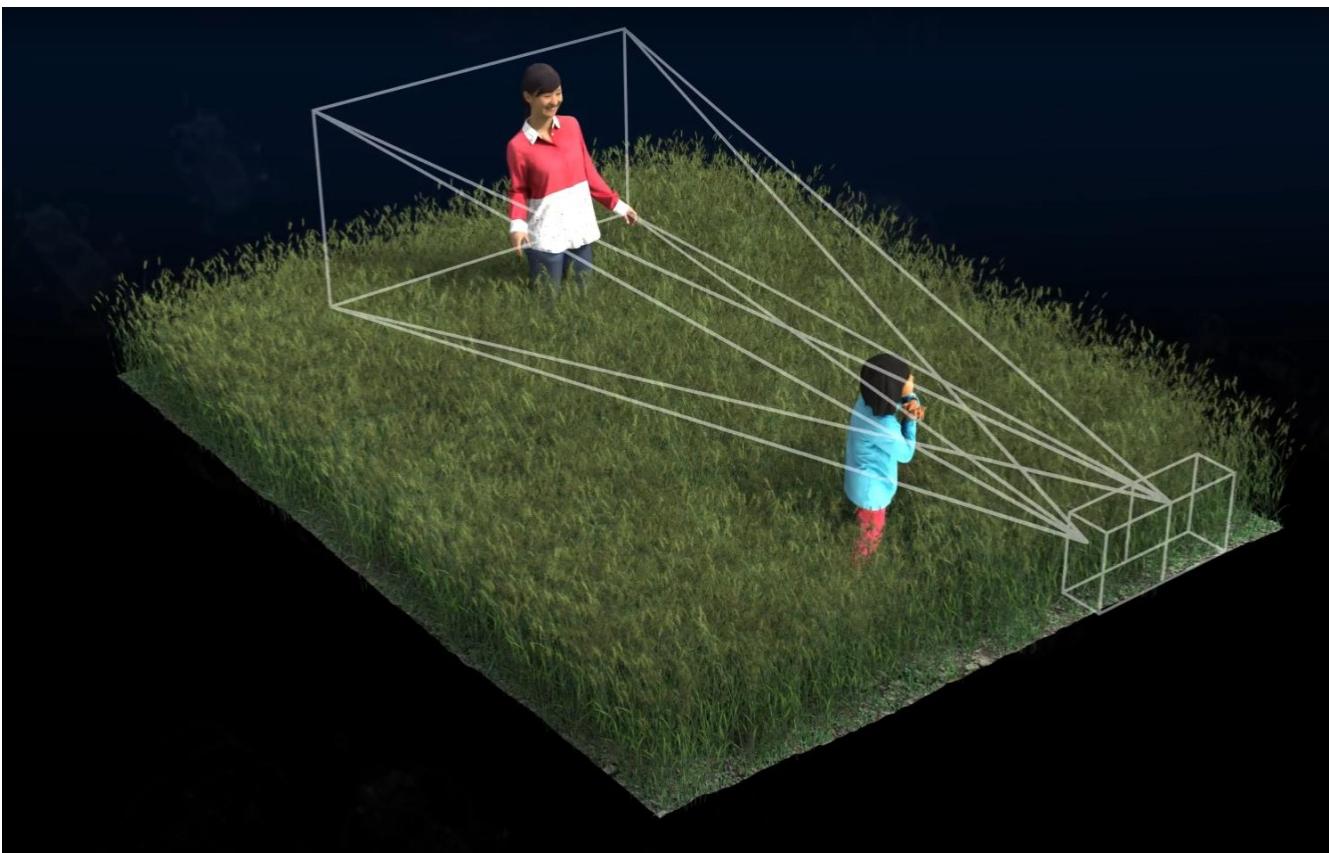
$$V(i, j, k) = Q_k^T \cdot S_{i,j}$$

donde:

- $S_{i,j}$  es la representación de características en el píxel  $(i, j)$ .
- $Q_k$  es el vector de características del proto-objeto  $k$  (obtenido de ViT).
- $V \in R^{h \times w \times Q}$  indica la relación de similitud entre el píxel y cada proto-objeto.

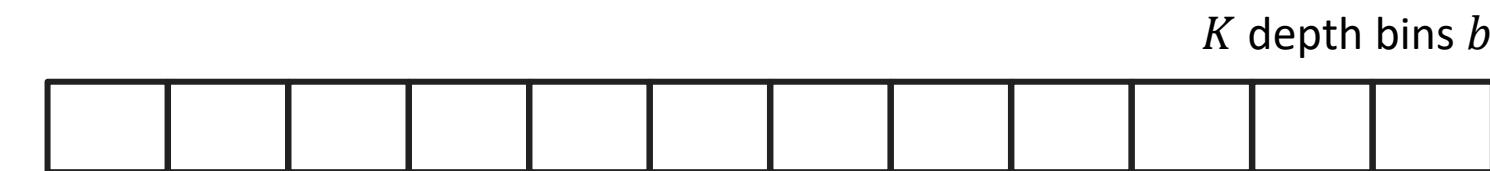
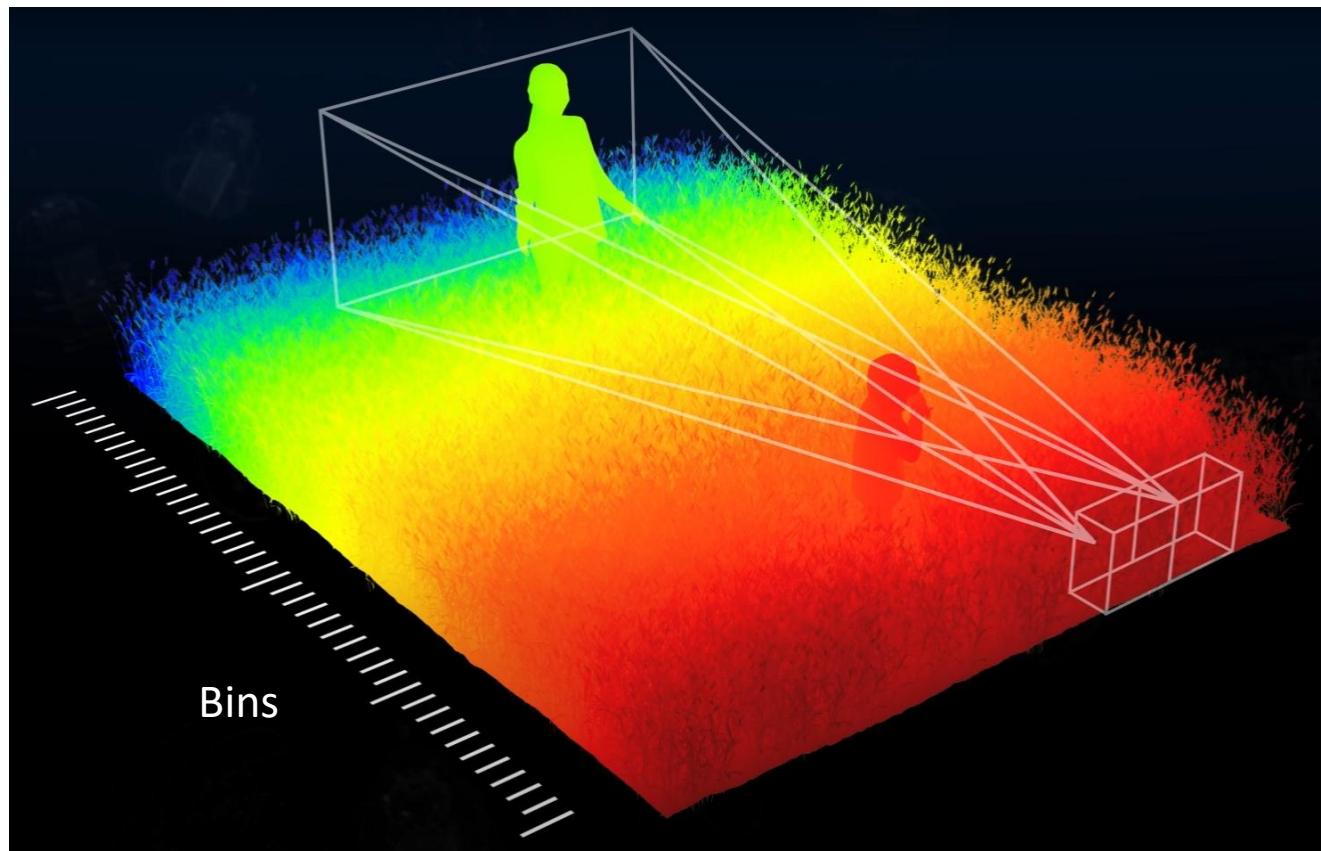
# SQL<sup>depth</sup>

## Depth bins



# SQL<sup>depth</sup>

## Depth bins



$$b = \text{MLP} \left( \bigoplus_{k=1}^Q \sum_{i,j} \text{Softmax}(V_k)_{i,j} \cdot S_{i,j} \right)$$

Center depth of the  $i$ -th bin:  $c(b_k) = d_{min} + (d_{max} - d_{min}) \left( \frac{b_k}{2} + \sum_{t=1}^{k-1} b_t \right)$

Probabilidad de que el píxel  $(i, j)$  pertenezca al bin  $k$

$$p_{i,j,k} = \text{softmax}(\hat{V}_{i,j,k})$$

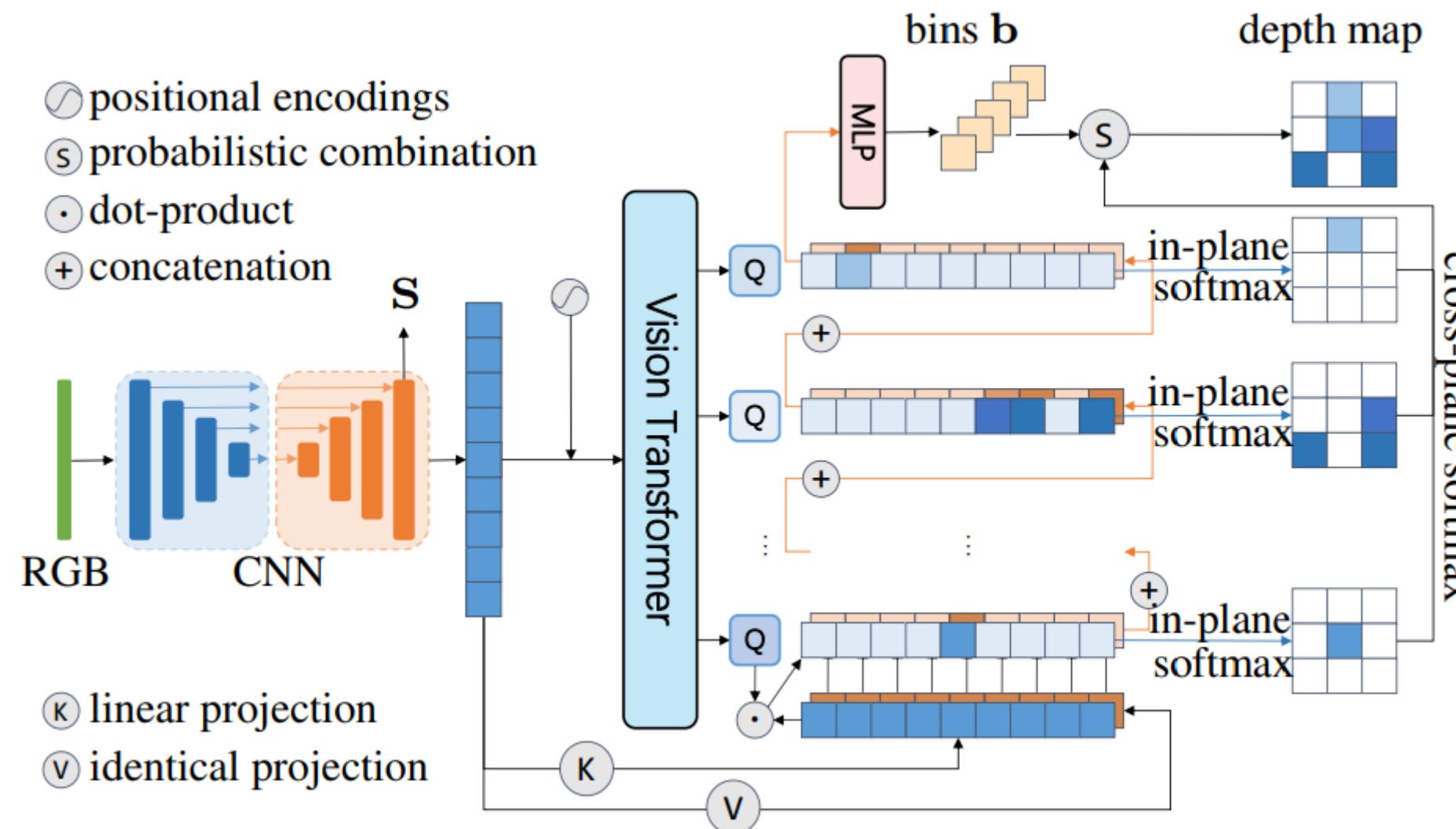
Profundidad estimada para el píxel  $(i, j)$

$$\tilde{d}_{i,j} = \sum_{k=1}^K c(b_k) \cdot p_{i,j,k}$$



# SQL<sup>depth</sup>

## DepthNet with the Self Query Layer



# SQLdepth

**Loss function**  $L = \mu L_{photo} + \lambda L_s$

**Photometric loss:**  $L_{photo} = \min_{t'} pe(I_t, I_{t' \rightarrow t}), \quad I_{t'} \in \{t-1, t+1\}$

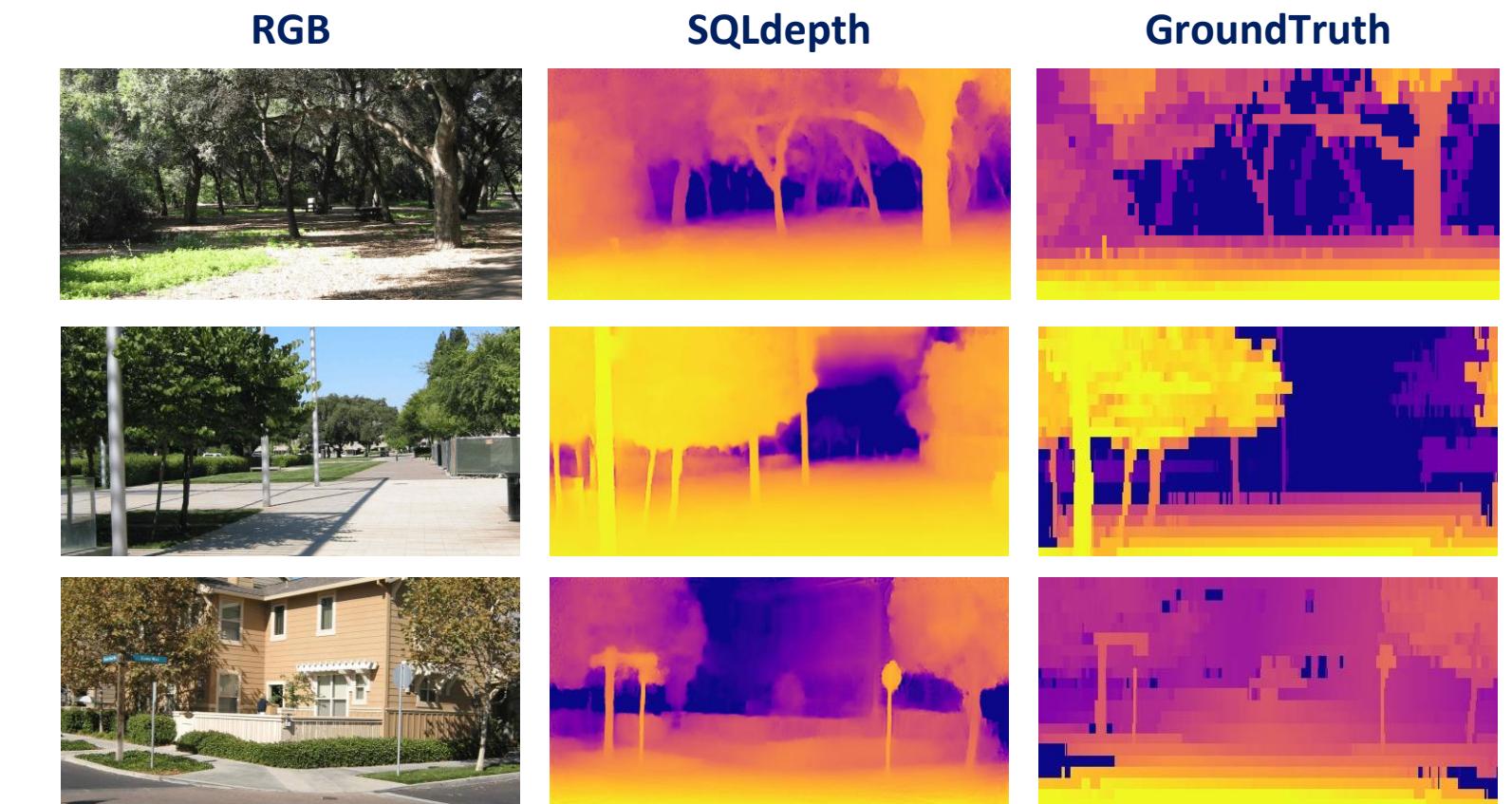
**Auto-masking:**  $\mu = \left[ \min_{t'} pe(I_t, I_{t' \rightarrow t}) < \min_{t'} pe(I_t, I_{t'}) \right]$

**Photometric error:**

$$pe(I_a, I_b) = \frac{\alpha}{2} (1 - SSIM(I_a, I_b)) + (1 - \alpha) |I_a - I_b|_1$$

**Edge-aware smooth loss:**

$$L_s = |\partial_x d_t^*| e^{-|\partial_x I_t|} + |\partial_y d_t^*| e^{-|\partial_y I_t|}$$



# Structural similarity index measure (SSIM)

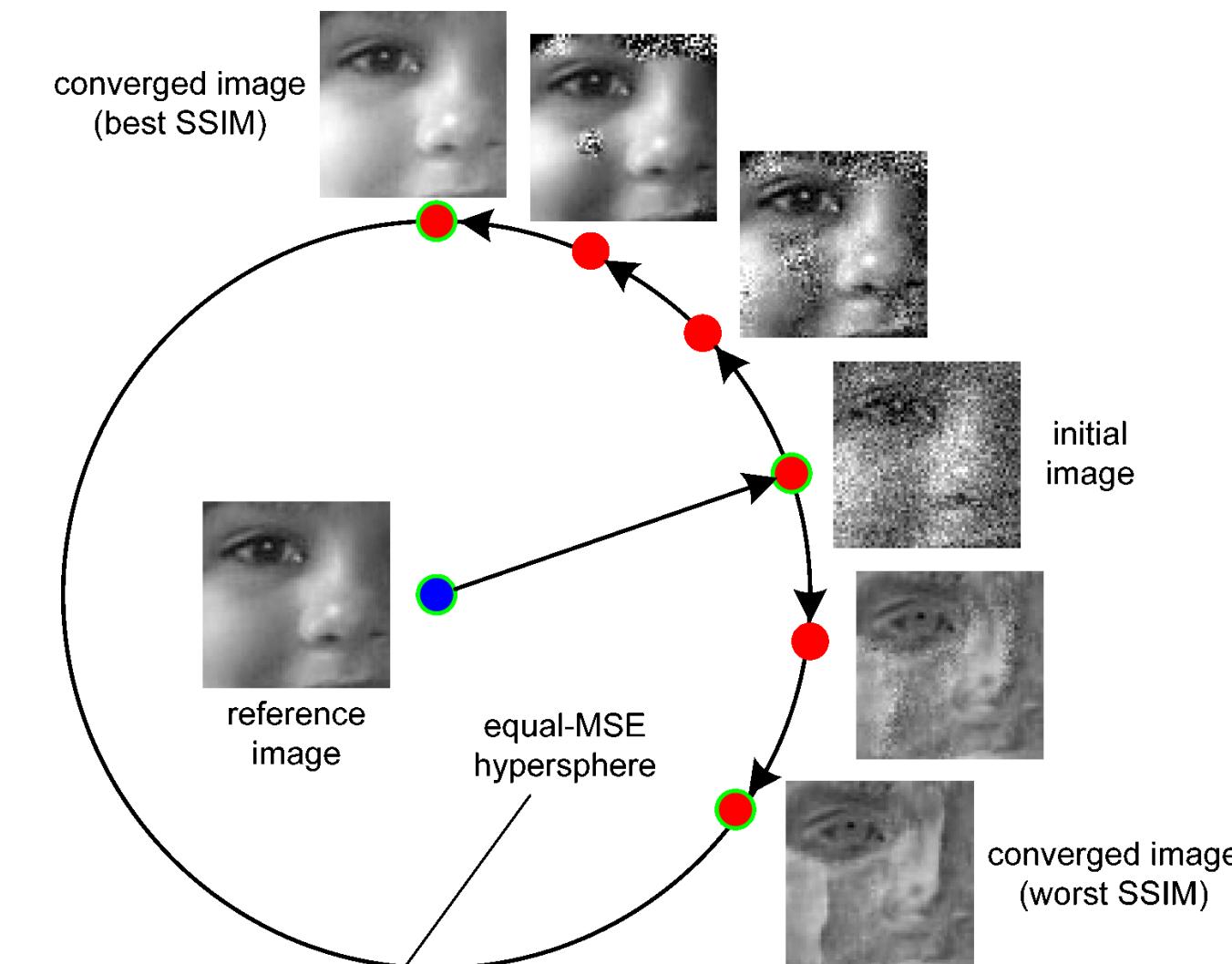
$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

Donde:

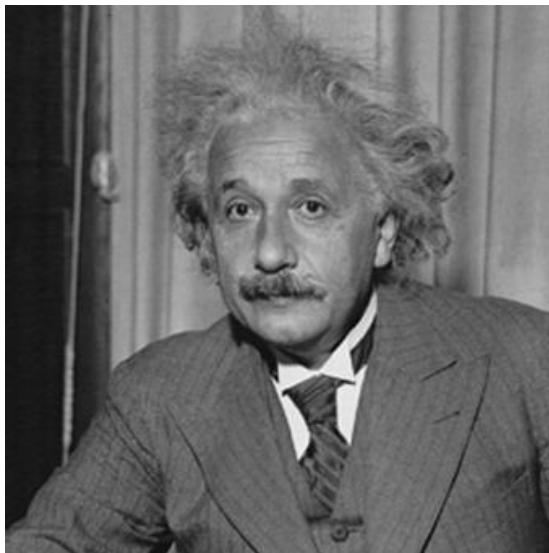
- $\mu_x$  y  $\mu_y$  son las medias locales de las imágenes  $x$  e  $y$ .
- $\sigma_x^2$  y  $\sigma_y^2$  son las varianzas locales de  $x$  e  $y$ , respectivamente.
- $\sigma_{xy}$  es la covarianza entre  $x$  e  $y$ .
- $C_1$  y  $C_2$  son constantes de estabilización para evitar divisiones por cero, generalmente definidas como:

$$C_1 = (K_1 L)^2, \quad C_2 = (K_2 L)^2$$

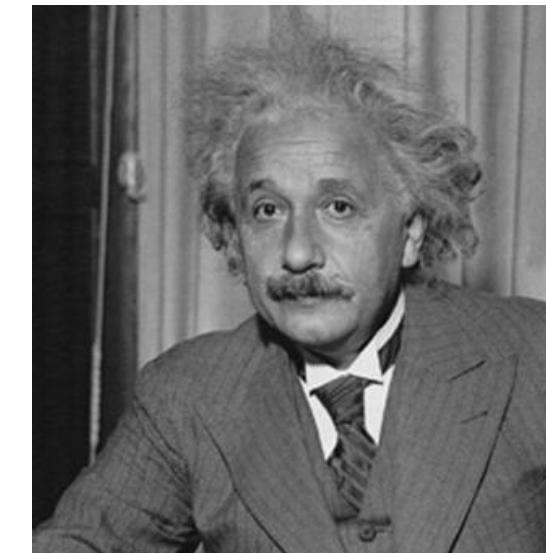
donde  $L$  es el rango dinámico de los valores de píxeles (por ejemplo, 255 para imágenes de 8 bits), y  $K_1$  y  $K_2$  son pequeñas constantes típicamente 0.01 y 0.03, respectivamente.



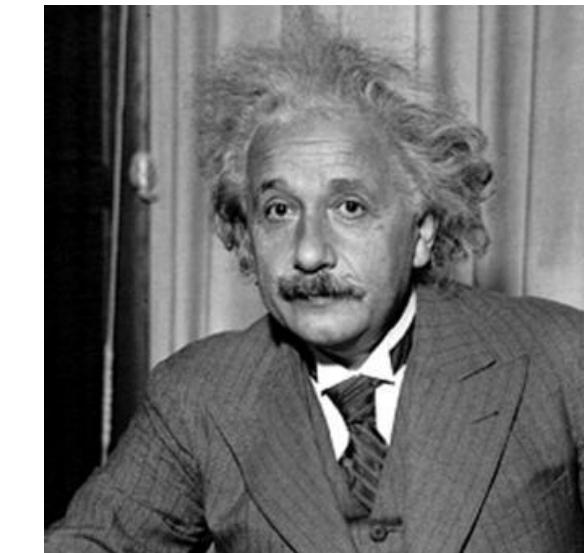
# Structural similarity index measure (SSIM)



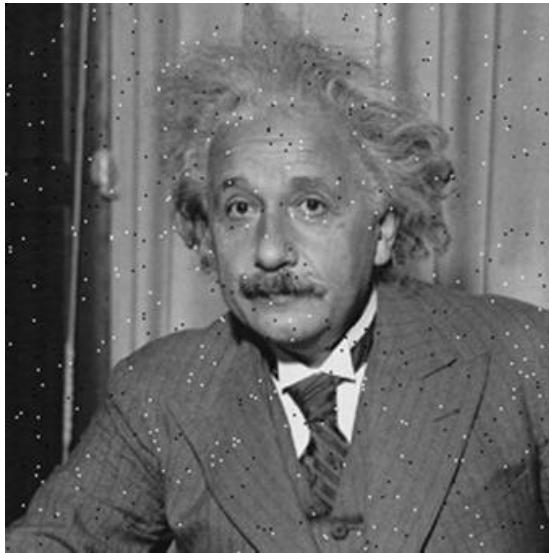
Original, MSE = 0, SSIM = 1



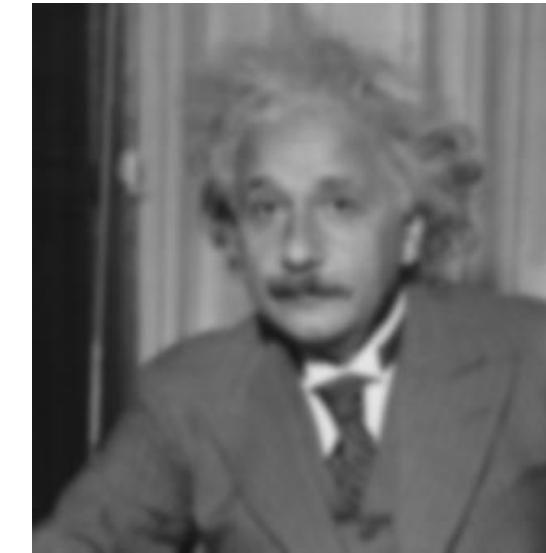
MSE = 144, SSIM = 0.988



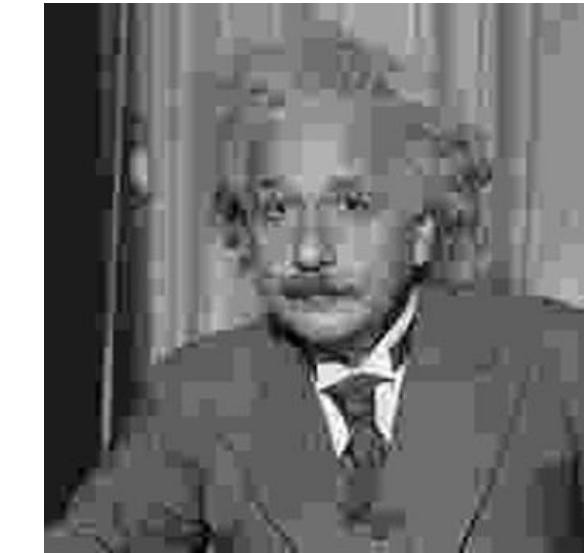
MSE = 144, SSIM = 0.913



MSE = 144, SSIM = 0.840



MSE = 144, SSIM = 0.694



MSE = 142, SSIM = 0.662



Zhou Wang et al. (2004) "Image quality assessment: from error visibility to structural similarity".  
IEEE transactions on image processing, 13(4), 600-612.

**TRANSFORMATEC**

# GRACIAS

*Victor Flores Benites*