



# Image Segmentation Based on Deep Neural Network and Active Contour Model

Alex Xu

June 18, 2024

# Content

## 1 Basic theory

- Overview
- Mathematical Derivation
- Dive into Deep Learning

## 2 Methodology

- From R-CNN to YOLO
- The Greatest Works
- Introduction to Combination

## 3 Experiment

- YOLOv8 With EMA and WIoU
- Deep Snake, Flexible Design

# Background

- Overview of image segmentation.

# Background

- Overview of image segmentation.
- Traditional: Accurate segmentation by Active Contour Model.

# Background

- Overview of image segmentation.
- Traditional: Accurate segmentation by Active Contour Model.
- Innovation: Deep learning methods.

# Active Contour Model

## Curve evolution

We define curve as

$$C(\theta, t) = (x(\theta, t), y(\theta, t)) \quad (1)$$

Then we have curvature

$$\frac{\partial C(\theta, t)}{\partial t} = F(k)N \quad (2)$$

As constant evolution

$$\frac{\partial C(\theta, t)}{\partial t} = V_0 N \quad (3)$$

As curvature evolution

$$\frac{\partial C(\theta, t)}{\partial t} = \alpha k N \quad (4)$$

# Active Contour Model

## Level-set method

In curve  $C = \{(x, y), \phi(x, y) = c\}$  where  $\phi(x, y)$  defined as level-set function, we introduce time  $t$  and apply partial derivatives

$$\frac{d\phi}{dt} = \frac{\partial\phi}{\partial t} + \frac{\partial\phi}{\partial x} \cdot \frac{\partial x}{\partial t} + \frac{\partial\phi}{\partial y} \cdot \frac{\partial y}{\partial t} = \frac{\partial\phi}{\partial t} + \nabla\phi \cdot \frac{\partial(x, y)}{\partial t} = 0 \quad (5)$$

$\nabla\phi = (\frac{\partial\phi}{\partial x}, \frac{\partial\phi}{\partial y})$  is the gradient of level set function  $\phi(x, y)$ , and  $\frac{\partial(x, y)}{\partial t} = \frac{\partial C}{\partial t} = (\frac{\partial x}{\partial t}, \frac{\partial y}{\partial t}) = \vec{V}$ .  
Then

$$\frac{\partial\phi}{\partial t} = -\nabla\phi \cdot \vec{V} = -|\nabla\phi| \cdot \frac{\vec{V}}{|\nabla\phi|} = |\nabla\phi| \cdot \left(-\frac{\nabla\phi}{|\nabla\phi|}\right) \cdot \vec{V} = |\nabla\phi| \cdot \vec{U} \cdot \vec{V} \quad (6)$$

# Active Contour Model

## Calculus of variations

The 1D energy function can be written as

$$E(\phi) = \int_{x_0}^{x_1} F(x, \phi, \phi') dx \quad (7)$$

In order to apply Taylor's Formula, we introduce a very small  $v(x)$

$$F(x, \phi + v, \phi' + v') = F(x, \phi, \phi') + \frac{\partial F}{\partial \phi} \cdot v + \frac{\partial F}{\partial \phi'} \cdot v' + \dots \quad (8)$$

Integrate both sides and collate to get

$$E(\phi + v) - E(\phi) = \int_{x_0}^{x_1} \left( v \frac{\partial F}{\partial \phi} + v' \frac{\partial F}{\partial \phi'} \right) dx \quad (9)$$

# Active Contour Model

## Calculus of variations

As  $\phi(x_0) + v(x_0) = a, \phi(x_1) + v(x_1) = b,$

$$\int_{x_0}^{x_1} v' \frac{\partial F}{\partial \phi'} dx = \int_{x_0}^{x_1} \frac{\partial F}{\partial \phi} dv = v \frac{\partial F}{\partial \phi'} \Big|_{x_0}^{x_1} - \int_{x_0}^{x_1} v \frac{d}{dx} \frac{\partial F}{\partial \phi'} dx = - \int_{x_0}^{x_1} v \frac{d}{dx} \frac{\partial F}{\partial \phi'} dx \quad (10)$$

Ignoring very small changes in the independent variable, we have

$$\frac{\partial F}{\partial \phi} - \frac{d}{dx} \left( \frac{\partial F}{\partial \phi'} \right) = 0 \quad (11)$$

This is the Euler equation for 1D energy functional. Similarly, The Euler equation for 2D is

$$\frac{\partial F}{\partial \phi} - \frac{\partial F}{\partial \phi_x} - \frac{\partial F}{\partial \phi_y} = 0 \quad (12)$$

# Active Contour Model

## Calculus of variations

Take (11) as an example. To solve the Euler equation, we introduce a small  $v(\cdot)$

$$E(\phi, t + \Delta t) = E(\phi, t) + \Delta t \int_{x_0}^{x_1} \frac{\partial \phi}{\partial t} \left( \frac{\partial F}{\partial \phi} - \frac{d}{dx} \left( \frac{\partial F}{\partial \phi'} \right) \right) dx \quad (13)$$

Let

$$\frac{\partial \phi}{\partial t} = - \left( \frac{\partial F}{\partial \phi} - \frac{d}{dx} \left( \frac{\partial F}{\partial \phi'} \right) \right) = \frac{d}{dx} \left( \frac{\partial F}{\partial \phi'} \right) - \frac{\partial F}{\partial \phi} \quad (14)$$

Then

$$\Delta E = E(\phi, t + \Delta t) - E(\phi, t) = -\Delta t \int \left( \frac{\partial F}{\partial \phi} - \frac{d}{dx} \left( \frac{\partial F}{\partial \phi'} \right) \right)^2 dx \quad (15)$$

The solution to the Euler equation of an energy functional tends to stabilize as the curve evolves, corresponding to the gradient flow of the given variation.

# Active Contour Model

## Chan Vese Model

Chan and Vese proposed the Chan-Vese active contour model. Given a grayscale image  $I(x)$ , the target  $\Omega_1$  and background  $\Omega_2$  are detected with characteristic uniformity and a clear difference, represented by the approximate grayscale constants  $c_1$  and  $c_2$ . The energy functional is defined as

$$\begin{aligned} F(c_1, c_2, \phi) = & \lambda_1 \cdot \int_{\Omega} (I(x) - c_1)^2 \cdot H(\phi(x)) dx + \lambda_2 \cdot \int_{\Omega} (I(x) - c_2)^2 \cdot (1 - H(\phi(x))) dx \\ & + \mu \cdot \int_{\Omega} \delta_0(\phi(x)) |\nabla \phi(x)| dx + v \cdot \int_{\Omega} H(\phi(x)) dx \end{aligned} \quad (16)$$

Where  $\lambda_1, \lambda_2 > 0, \mu, v \geq 0$ .  $H(\phi)$  is the unit step function(Heaviside function).  $\delta_0$  is its derivative, from which the Dirac function can be derived.

$$H(z) = \begin{cases} 0 & z < 0 \\ 1 & z \geq 0 \end{cases}, \delta_0(z) = \frac{d}{dz} H(z) \quad (17)$$

# Active Contour Model

## Chan Vese Model

Substituting it into the Euler-Lagrange equation, the level set gradient flow of the Chan-Vese model is solved.

$$\frac{\partial \phi}{\partial t} = -\frac{\partial F}{\partial \phi} = \delta_\varepsilon \left[ \mu \operatorname{div} \left( \frac{\nabla \phi}{|\nabla \phi|} \right) - v - \lambda_1 (I(x) - c_1)^2 + \lambda_2 (I(x) - c_2)^2 \right] \quad (18)$$

where

$$c_1 = \frac{\int I(x) \cdot H(\phi(x)) dx}{\int H(\phi(x)) dx}, \quad c_2 = \frac{\int I(x) \cdot (1 - H(\phi(x))) dx}{\int (1 - H(\phi(x))) dx} \quad (19)$$

## Convolutional Neural Networks

- convolutional layer
- pooling layer
- flatten layer
- fully connected layer
- activation layer

# Neural Network

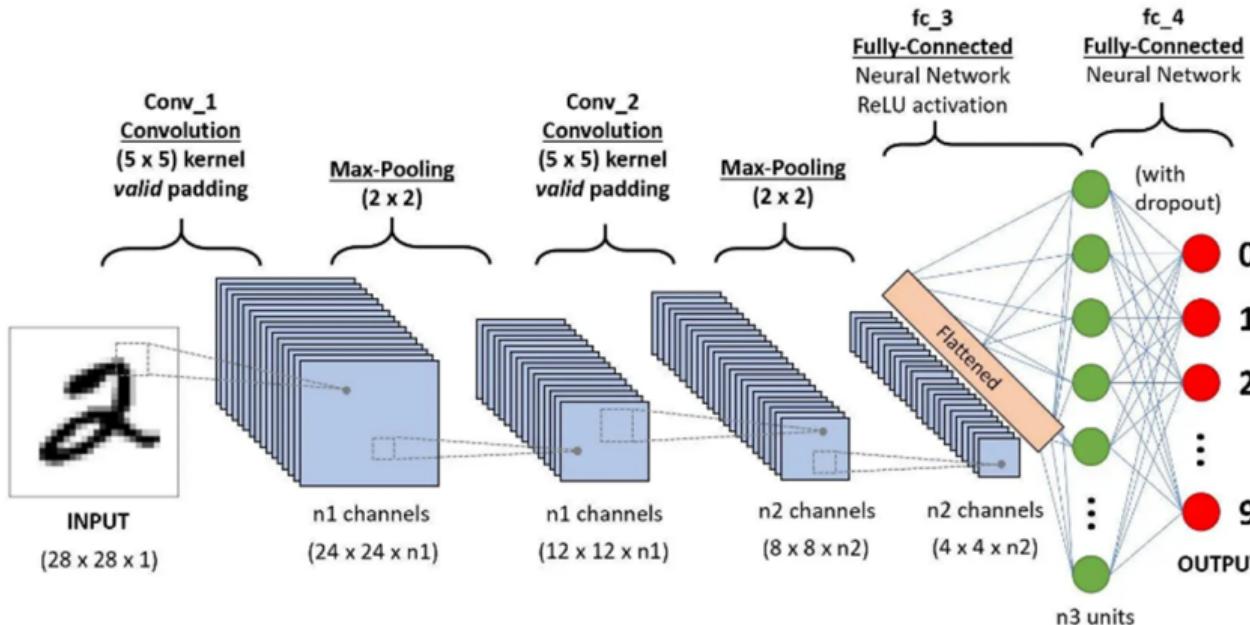


Figure 1: Handwritten number recognition pipeline

# Loss Functions

## Softmax

$$\text{Softmax}(x) = \frac{e^{x_i}}{\sum_i e^{x_i}}$$

(20)

# Loss Functions

## Softmax

$$\text{Softmax}(x) = \frac{e^{x_i}}{\sum_i e^{x_i}}$$

(20)

## Sigmoid

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

(21)

# Loss Functions

Softmax

$$\text{Softmax}(x) = \frac{e^{x_i}}{\sum_i e^{x_i}}$$

(20)

Sigmoid

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

(21)

tanh

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

(22)

# Loss Functions

Softmax

$$\text{Softmax}(x) = \frac{e^{x_i}}{\sum_i e^{x_i}} \quad (20)$$

Sigmoid

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}} \quad (21)$$

tanh

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (22)$$

ReLU

$$\text{ReLU}(x) = \max(0, x) \quad (23)$$

# Optimizations

## Methods

- Batch normalization

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} \quad (24)$$

- Stochastic gradient descent

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta, x^{(i)}, y^{(i)}) \quad (25)$$

# Classic Models

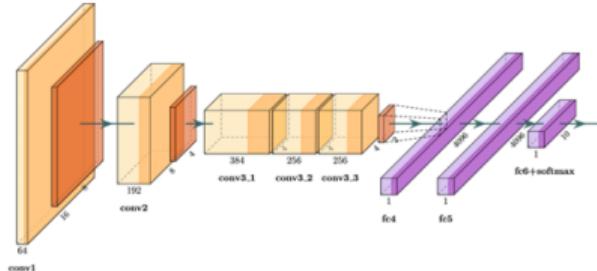


Figure 2: AlexNet

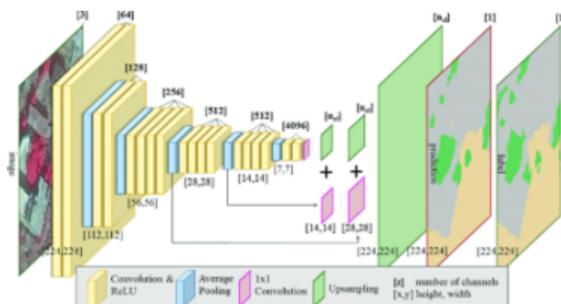


Figure 3: FCN

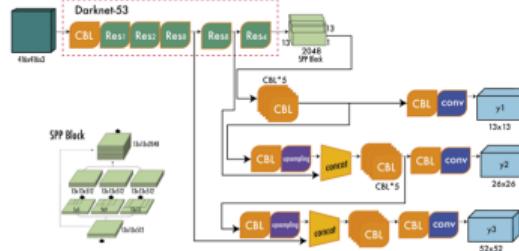


Figure 4: DarkNet

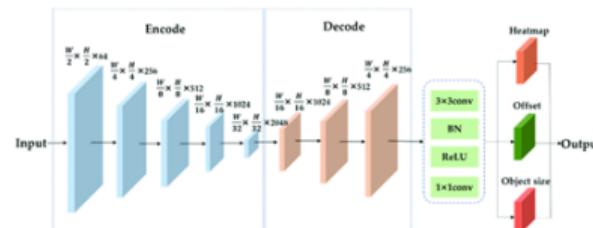


Figure 5: CenterNet

# Content

## 1 Basic theory

- Overview
- Mathematical Derivation
- Dive into Deep Learning

## 2 Methodology

- From R-CNN to YOLO
- The Greatest Works
- Introduction to Combination

## 3 Experiment

- YOLOv8 With EMA and WIoU
- Deep Snake, Flexible Design

# R-CNN Series

R-CNN is an object detection model that uses selective search for region proposals, feeds these into a pre-trained CNN for feature extraction, and then classifies and refines the regions using SVMs and bounding box regressors.

It laid the foundation for deep learning-based object detection but was computationally intensive.

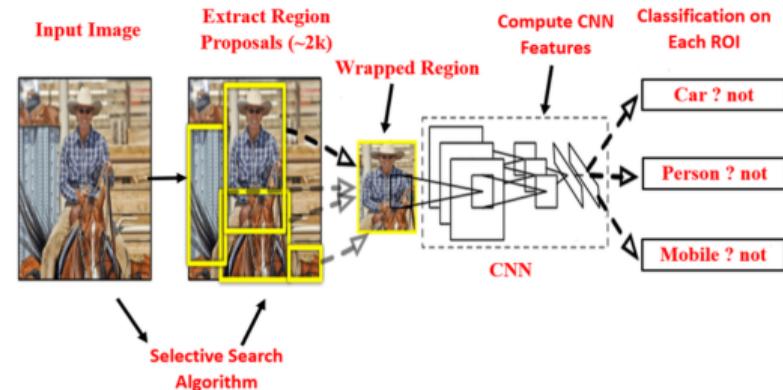


Figure 6: R-CNN

# R-CNN series

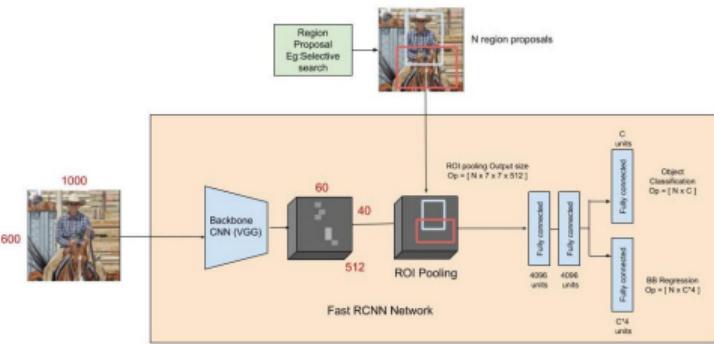


Figure 7: Fast R-CNN

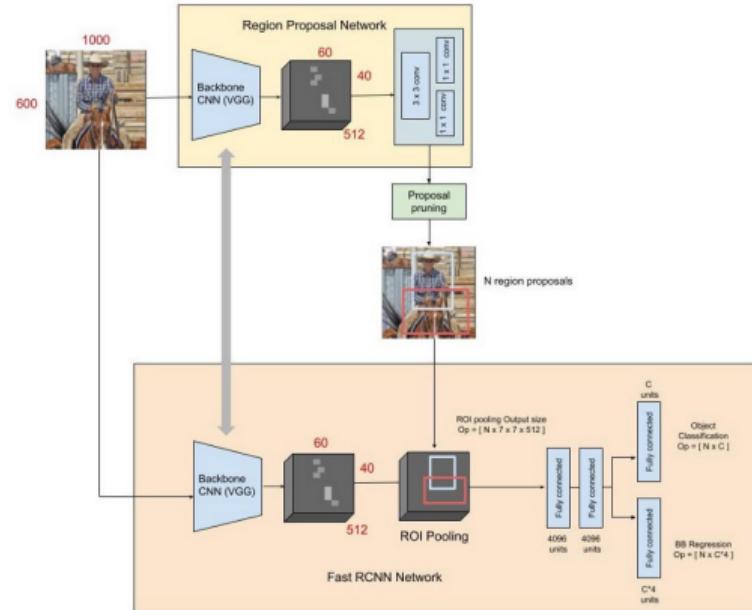


Figure 8: Faster R-CNN

# You Only Look Once

## Innovations

- Unified detection framework
- Real-time detection
- Global contextual prediction
- Spatially separated bounding boxes
- End-to-end training

## YOLO: You Only Look Once

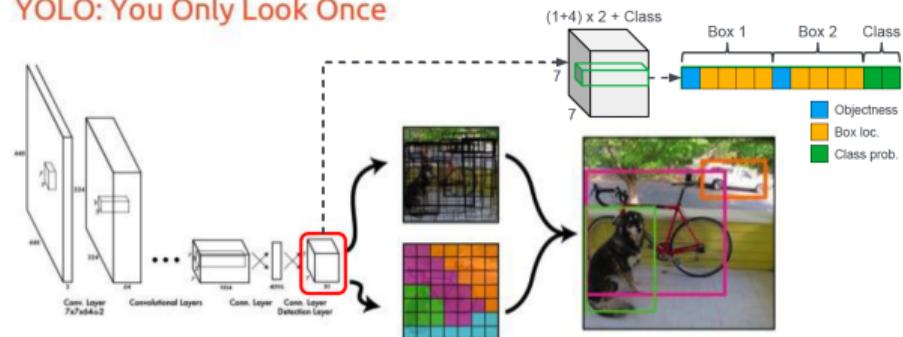


Figure 9: YOLOv1 architecture

# Loss Function Improvement

## YOLOv1

$$\begin{aligned} \text{Loss} = & \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B I_{ij}^{\text{obj}} \left[ \left( x_i^j - \hat{x}_i^j \right)^2 + \left( y_i^j - \hat{y}_i^j \right)^2 \right] \\ & + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B I_{ij}^{\text{obj}} \left[ \left( \sqrt{w_i^j} - \sqrt{\hat{w}_i^j} \right)^2 + \left( \sqrt{h_i^j} - \sqrt{\hat{h}_i^j} \right)^2 \right] \\ & + \sum_{i=0}^{S^2} \sum_{j=0}^B I_{ij}^{\text{obj}} \left( C_i^j - \hat{C}_i^j \right)^2 \\ & + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B I_{ij}^{\text{noobj}} \left( C_i^j - \hat{C}_i^j \right)^2 \\ & + \sum_{i=0}^{S^2} I_{ij}^{\text{obj}} \sum_{c \in \text{classes}} \left( P_i^j(c) - \hat{C}_i^j(c) \right)^2 \end{aligned} \tag{26}$$



# Loss Function Improvement

## YOLOv3

$$\begin{aligned} \text{Loss} = & \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B I_{ij}^{\text{obj}} \left[ \left( x_i^j - \hat{x}_i^j \right)^2 + \left( y_i^j - \hat{y}_i^j \right)^2 \right] \\ & + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B I_{ij}^{\text{obj}} \left[ \left( \sqrt{w_i^j} - \sqrt{\hat{w}_i^j} \right)^2 + \left( \sqrt{h_i^j} - \sqrt{\hat{h}_i^j} \right)^2 \right] \\ & - \sum_{i=0}^{S^2} \sum_{j=0}^B I_{ij}^{\text{obj}} \left[ \hat{C}_i^j \log \left( C_i^j \right) + \left( 1 - \hat{C}_i^j \right) \log \left( 1 - C_i^j \right) \right] \quad (27) \\ & - \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B I_{ij}^{\text{noobj}} \left[ \hat{C}_i^j \log \left( C_i^j \right) + \left( 1 - \hat{C}_i^j \right) \log \left( 1 - C_i^j \right) \right] \\ & - \sum_{i=0}^{S^2} I_{ij}^{\text{obj}} \sum_{c \in \text{classes}} \left[ \hat{P}_i^j \log \left( P_i^j \right) + \left( 1 - \hat{P}_i^j \right) \log \left( 1 - P_i^j \right) \right] \end{aligned}$$



# Masterpieces

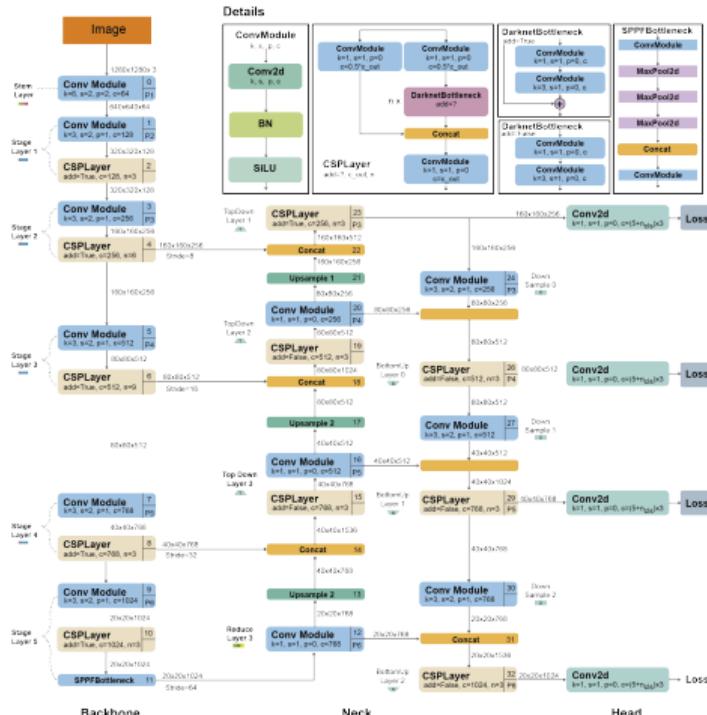


Figure 10: YOLOv5 architecture

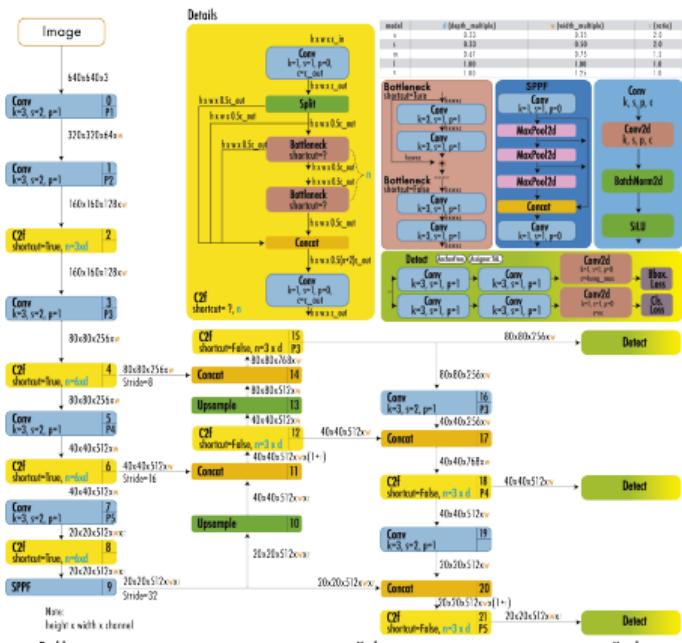


Figure 11: YOLOv8 architecture

# Applications

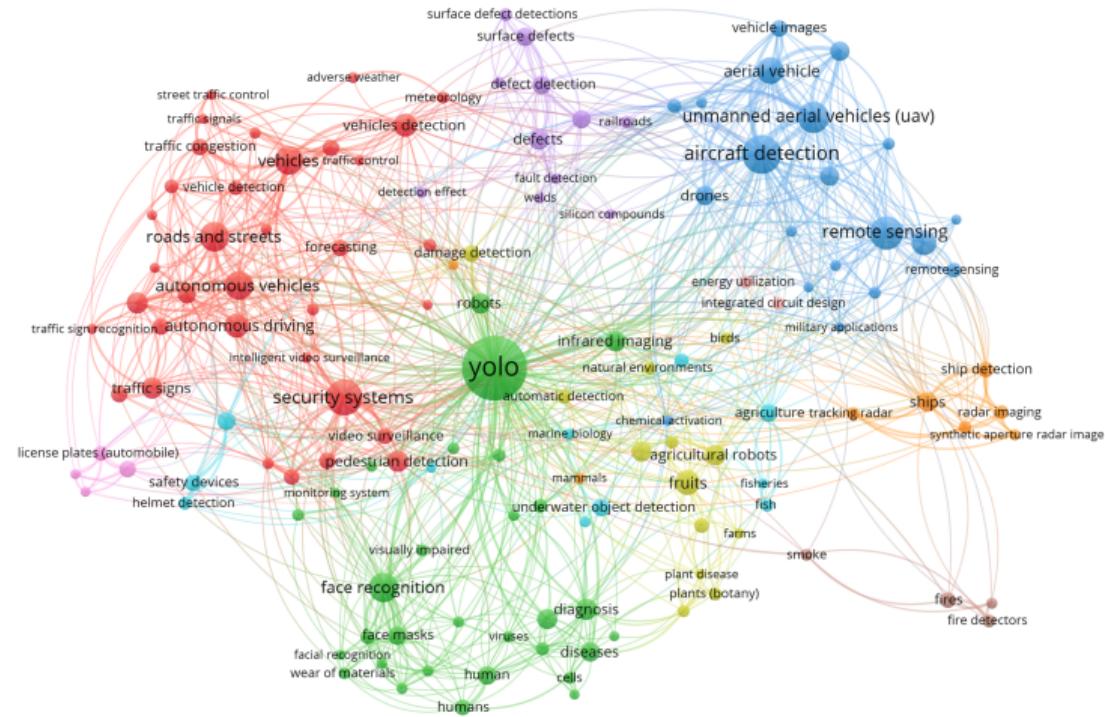


Figure 12: Applications on YOLO

# Deep Snake

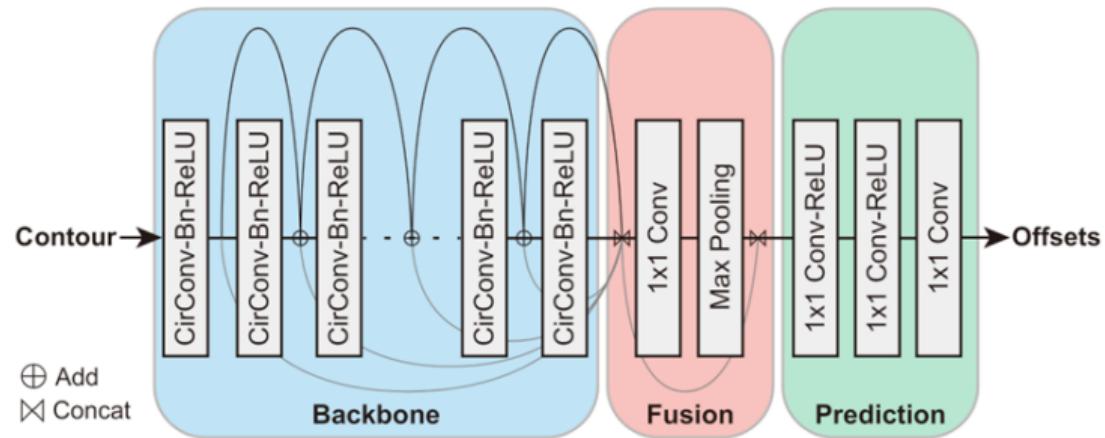


Figure 13: Deep Snake architecture

# Content

## 1 Basic theory

- Overview
- Mathematical Derivation
- Dive into Deep Learning

## 2 Methodology

- From R-CNN to YOLO
- The Greatest Works
- Introduction to Combination

## 3 Experiment

- YOLOv8 With EMA and WIoU
- Deep Snake, Flexible Design

# Inprovement: Efficient Multi-scale Attention

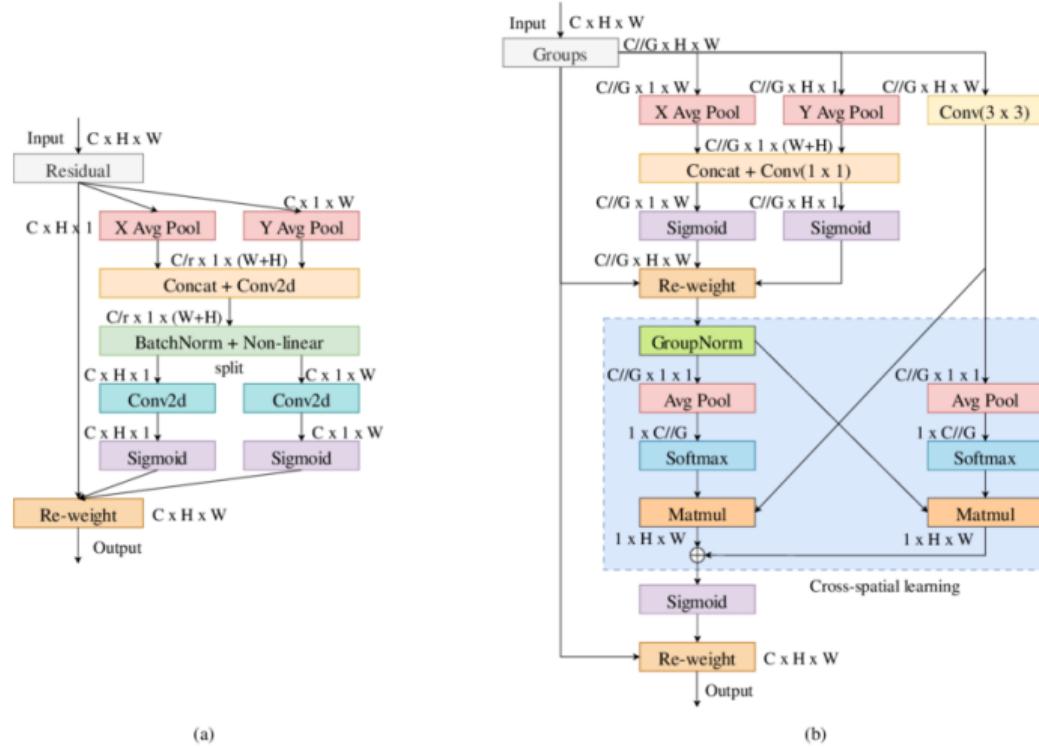


Figure 14: Structure of CA and EMA

# Inprovement: Efficient Multi-scale Attention

## Pooling output comparison

In coordinate attention(CA), the 1D global average-pooling for encoding information along the horizontal dimension direction in  $C$  at height  $H$  can be denoted by

$$z_c^H(H) = \frac{1}{W} \sum_{0 \leq i \leq W} x_c(H, i) \quad (28)$$

The pooling output in  $C$  at width  $W$  can be formulated as

$$z_c^W(W) = \frac{1}{H} \sum_{0 \leq j \leq H} x_c(j, W) \quad (29)$$

Efficient multi-scale attention(EMA) employs 2D global average pooling to aggregate cross-spatial information, encoding it into the  $1 \times 1$  branch output. Outputs of the least branch will be transformed to the correspond dimension shape before the joint activation mechanism of channel features, i.e.,  $\mathbb{R}_1^{1 \times C//G} \times \mathbb{R}_3^{C//G \times HW}$ .

The 2D global pooling operation is

$$z_c = \frac{1}{H \times W} \sum_j^H \sum_i^W x_c(i, j) \quad (30)$$

# Improvement: Wise-IoU

In YOLOv2, we have bounding box regression(BBR) loss expressed as

$$\mathcal{L}(\vec{B}, \vec{B}_{gt}) = \left\| \vec{B} - \vec{B}_{gt} \right\| \quad (31)$$

But it cannot shield the inference of the size of the bounding box. So we come up with intersection over union(IoU)

$$\mathcal{L}_{IoU} = 1 - IoU = 1 - \frac{W_i H_i}{S_i} \quad (32)$$

In order to prevent circumstances like  $W_i = 0$  or  $H_i$ , we add penalty term  $\mathcal{R}_i$

$$\mathcal{L}_i = \mathcal{L}_{IoU} + \mathcal{R}_i \quad (33)$$

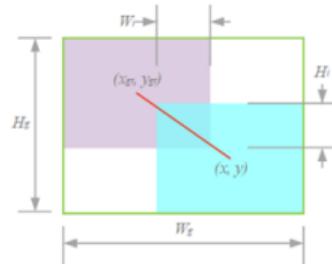


Figure 15: An IoU example

# Improvement: Wise-IoU

## Distance-IoU

Consider normalized distance between central points of two bounding boxes

$$\mathcal{R}_{\text{DIOU}} = \frac{(x - x_{gt})^2 + (y - y_{gt})^2}{W_g^2 + H_g^2} \quad (34)$$

# Improvement: Wise-IoU

## Distance-IoU

Consider normalized distance between central points of two bounding boxes

$$\mathcal{R}_{\text{D}\text{IoU}} = \frac{(x - x_{gt})^2 + (y - y_{gt})^2}{W_g^2 + H_g^2} \quad (34)$$

## Complete-IoU

Add the consideration of aspect ratio

$$\mathcal{R}_{\text{C}\text{IoU}} = \mathcal{R}_{\text{D}\text{IoU}} + \alpha v, \alpha = \frac{v}{\mathcal{L}_{\text{IoU}} + v} \quad (35)$$

# Improvement: Wise-IoU

## Efficient-IoU

Increase the punishment for distance metric

$$\mathcal{R}_{\text{EIoU}} = \mathcal{R}_{\text{DIoU}} + \frac{(x - x_{gt})^2}{W_g^2} + \frac{(y - y_{gt})^2}{H_g^2} \quad (36)$$

# Inprovement: Wise-IoU

## Efficient-IoU

Increase the punishment for distance metric

$$\mathcal{R}_{\text{EIoU}} = \mathcal{R}_{\text{DIoU}} + \frac{(x - x_{gt})^2}{W_g^2} + \frac{(y - y_{gt})^2}{H_g^2} \quad (36)$$

## Scylla-IoU

We define angle cost as

$$\Lambda = \sin \left( 2 \sin^{-1} \frac{\min(|x - x_{gt}|, |y - y_{gt}|)}{\sqrt{(x - x_{gt})^2 + (y - y_{gt})^2} + \varepsilon} \right) \quad (37)$$

Then the distance cost comes to be

$$\Delta = \frac{1}{2} \sum_{t=w,h} (1 - e^{-\gamma \rho_t}), \gamma = 2 - \Lambda \quad (38)$$

Therefore, we have

$$\mathcal{R}_{\text{SIoU}} = \Delta + \Sigma \quad (39)$$

# Improvement: Wise-IoU

## Wise-IoU

Based on SIoU and Elou, we reconsider low-quality examples and geometric factors, and construct Wise-IoU with 2 layers of attention mechanism

$$\mathcal{L}_{\text{WIoU}} = \mathcal{R}_{\text{WIoU}} \mathcal{L}_{\text{IoU}} \quad (40)$$

where

$$\mathcal{R}_{\text{WIoU}} = \exp \left( \frac{(x - x_{gt})^2 + (y - y_{gt})^2}{(W_g^2 + H_g^2)^*} \right) \quad (41)$$

$\mathcal{R}_{\text{WIoU}} \in [1, e]$  which will significantly amplify  $\mathcal{L}_{\text{IoU}}$  of the ordinary-quality anchor box.

$\mathcal{L}_{\text{IoU}} \in [0, 1]$ , which will significantly reduce  $\mathcal{R}_{\text{WIoU}}$  of the high-quality anchor box and its focus on the distance between central points when the anchor box coincides well with the target box.  
Superscript \* indicates  $W_g, H_g$  are detached from the computational graph.

# Experiment: Where to add EMA?

	Precision(B) (%)	Recall(B) (%)	mAP50(B) (%)	mAP50-95(B) (%)
YOLOv8	<b>60.047</b>	68.864	66.999	56.038
YOLOv8+EMA(neck)	58.934	<b>70.791</b>	66.75	57.047
YOLOv8+EMA(backbone)	59.085	70.489	<b>68.187</b>	<b>57.669</b>

	Precision(M) (%)	Recall(M) (%)	mAP50(M) (%)	mAP50-95(M) (%)
YOLOv8	<b>61.783</b>	69.719	68.568	55.745
YOLOv8 +EMA(neck)	59.51	71.393	68.149	56.008
YOLOv8 +EMA(backbone)	60.575	<b>72.123</b>	<b>69.953</b>	<b>57.032</b>

# Experiment: How's the combination?

	Precision(B) (%)	Recall(B) (%)	mAP50(B) (%)	mAP50-95(B) (%)
YOLOv8	<b>60.047</b>	68.864	66.999	56.038
YOLOv8+WIoU	57.03	76.804	66.26	52.148
YOLOv8+EMA	59.085	70.489	68.187	57.669
YOLOv8+EMA+WIoU	59.276	<b>74.3</b>	<b>69.169</b>	<b>58.039</b>

	Precision(M) (%)	Recall(M) (%)	mAP50(M) (%)	mAP50-95(M) (%)
YOLOv8	<b>61.783</b>	69.719	68.568	55.745
YOLOv8+WIoU	58.219	77.826	68.119	53.059
YOLOv8+EMA	60.575	72.123	69.953	57.032
YOLOv8+EMA+WIoU	60.184	<b>75.724</b>	<b>70.888</b>	<b>57.467</b>



# Assessment: EMA and WIoU

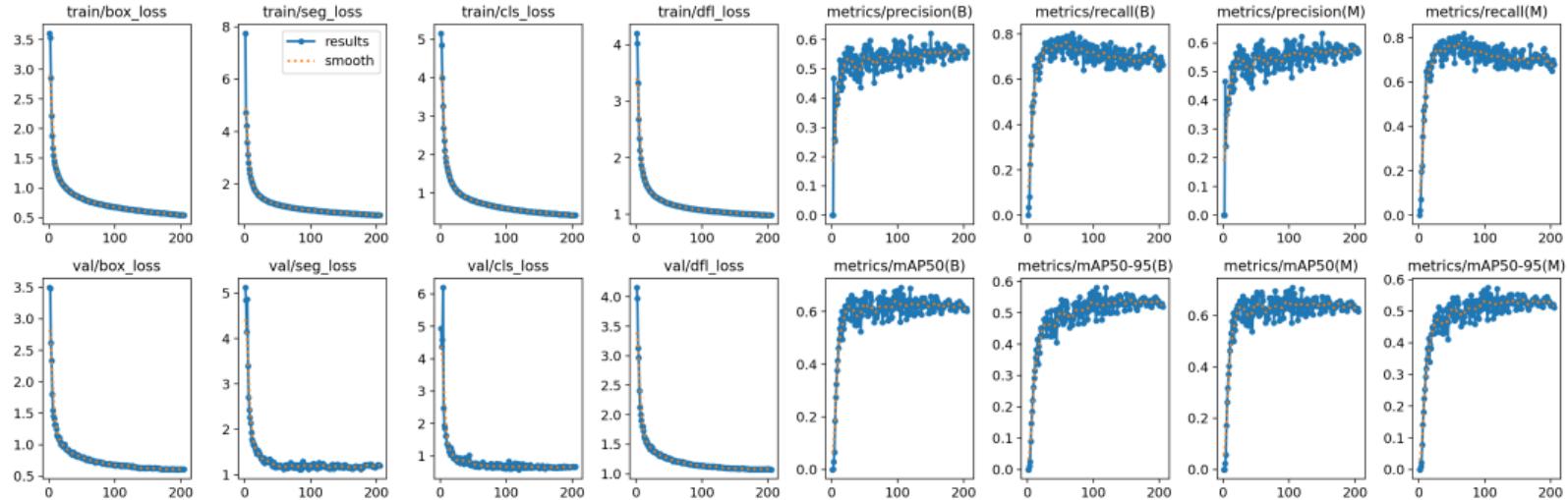


Figure 16: Results

# Assessment: EMA and WIoU

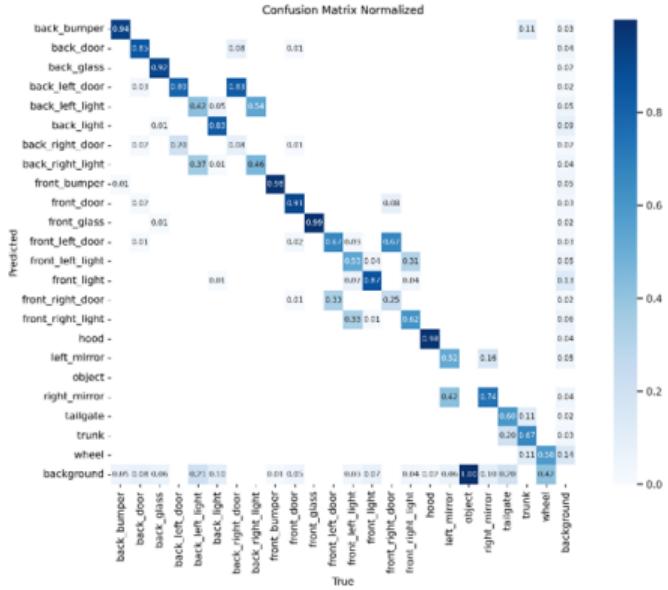


Figure 17: Original YOLOv8 confusion matrix

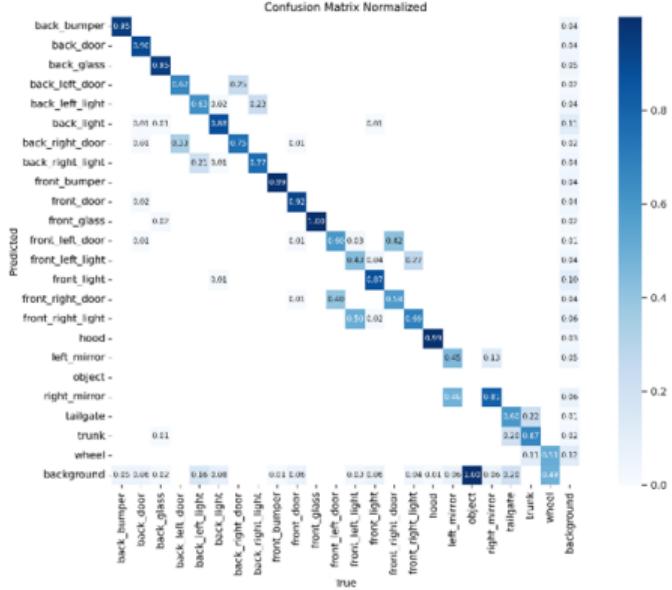


Figure 18: Improved YOLOv8 confusion matrix

# Results: EMA and WIoU



Figure 19: Original labels

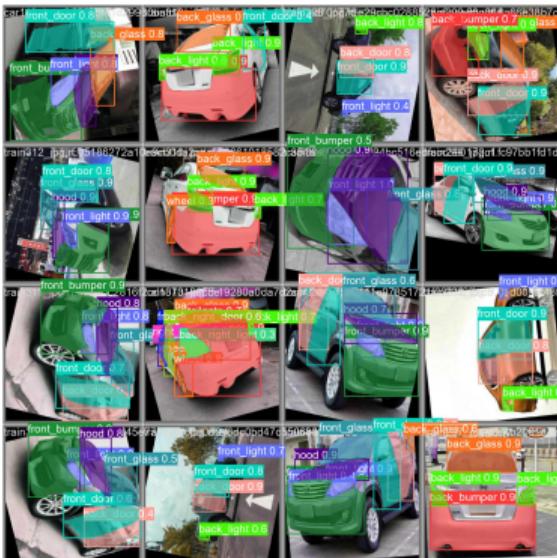


Figure 20: YOLOv8 with EMA and WIoU segmentation

# Improvement: Deep Layer Aggregation

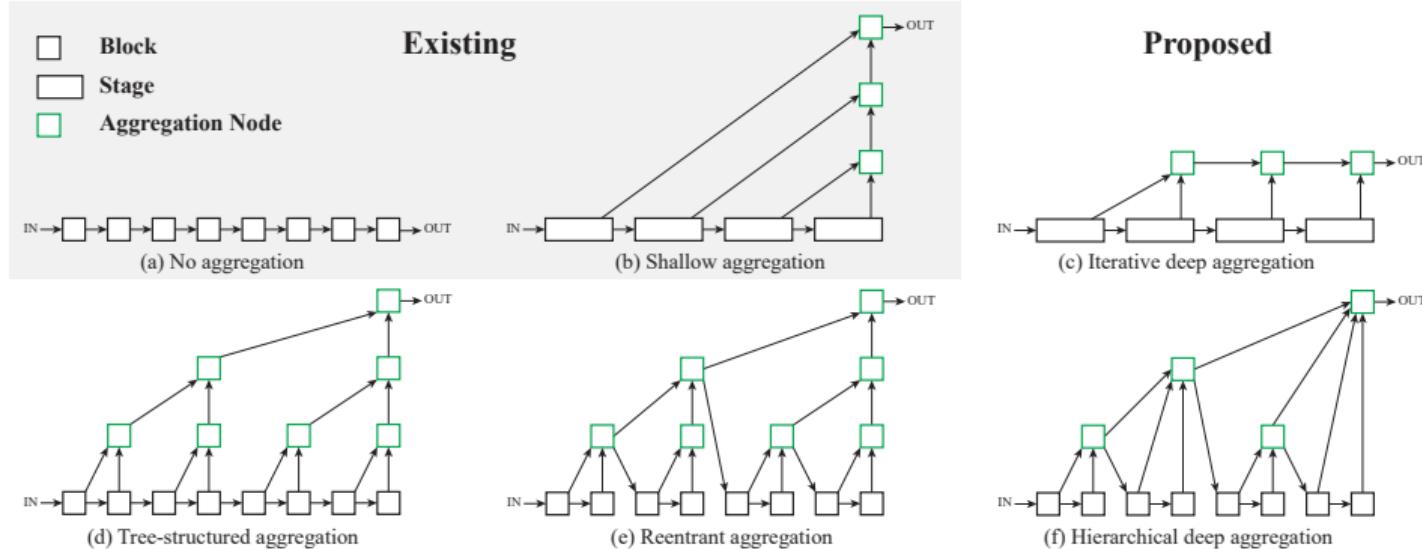


Figure 21: Different ways to aggregate layers

# Improvement: Deep Layer Aggregation

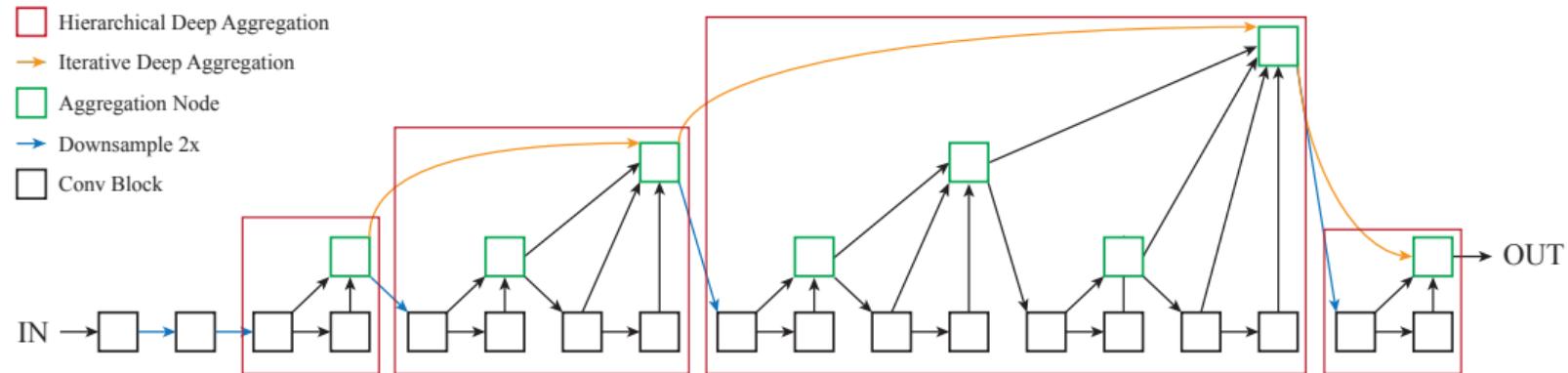


Figure 22: DLA network architecture

# Experiment: Deep Snake

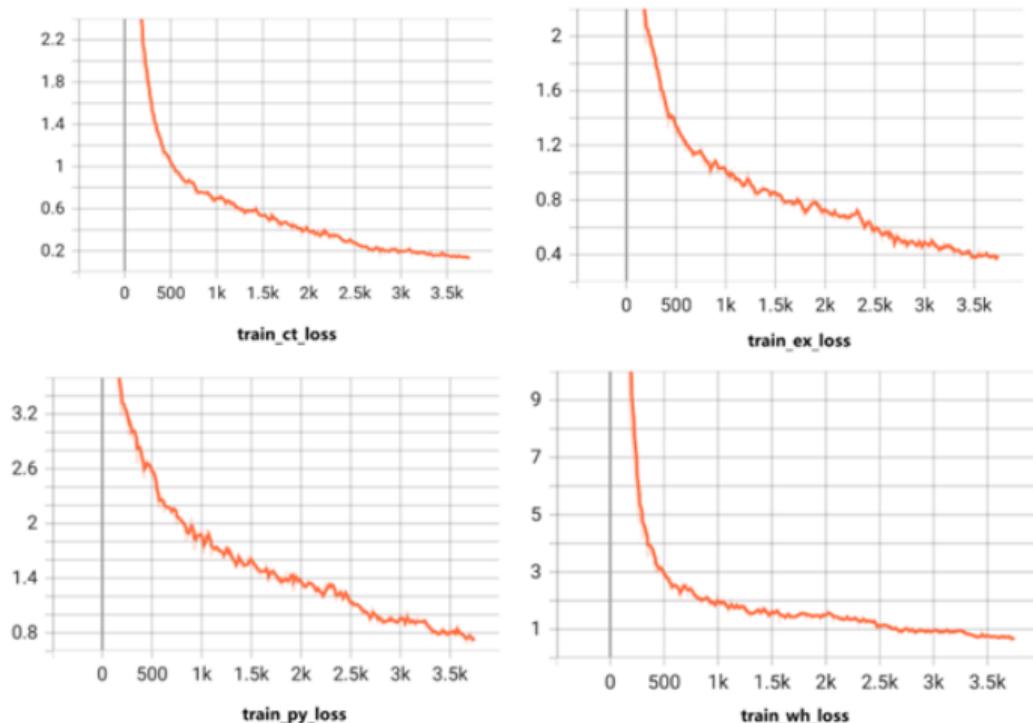


Figure 23: Loss during training

# Experiment: Deep Snake

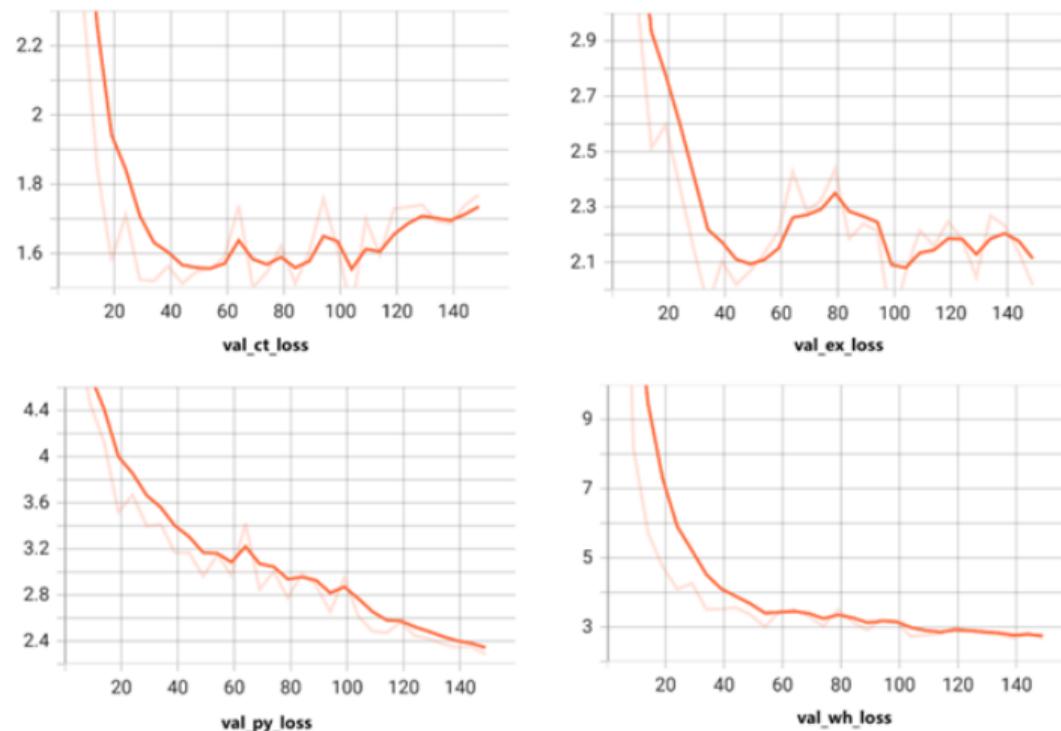


Figure 24: Loss during validation

# Experiment: Deep Snake

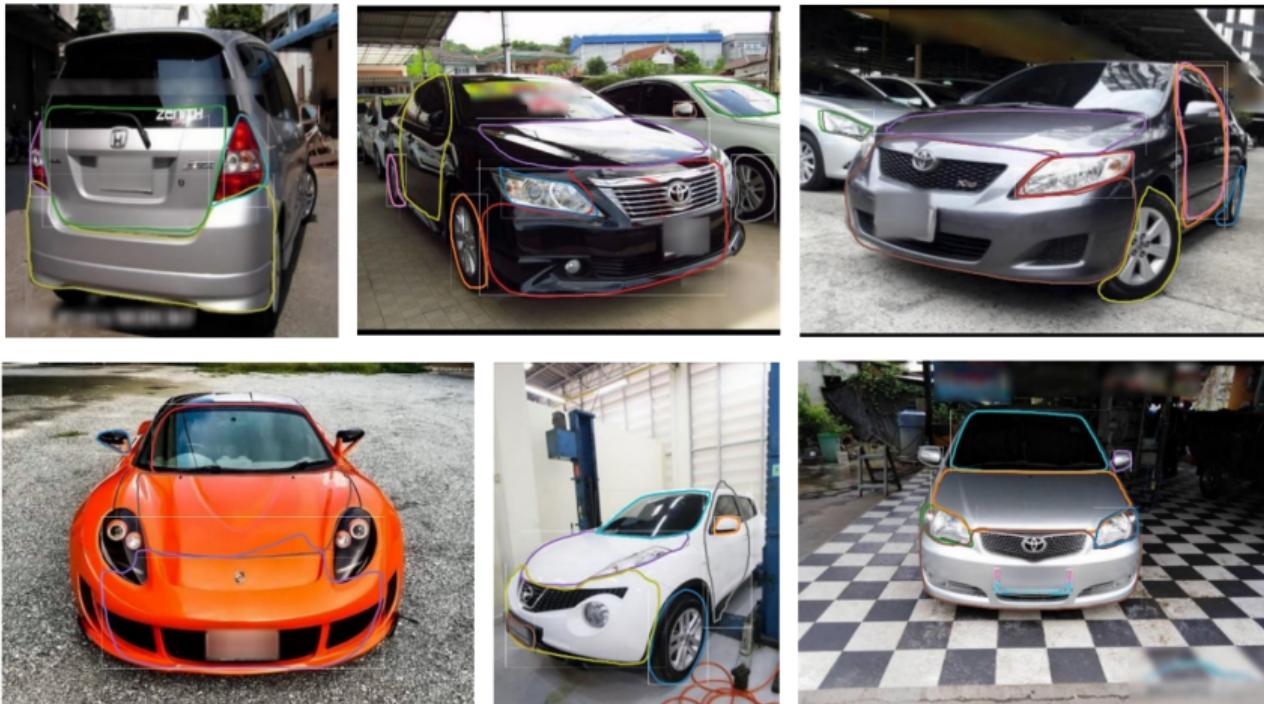


Figure 25: Sample pictures segmented by Deep Snake



THANK YOU