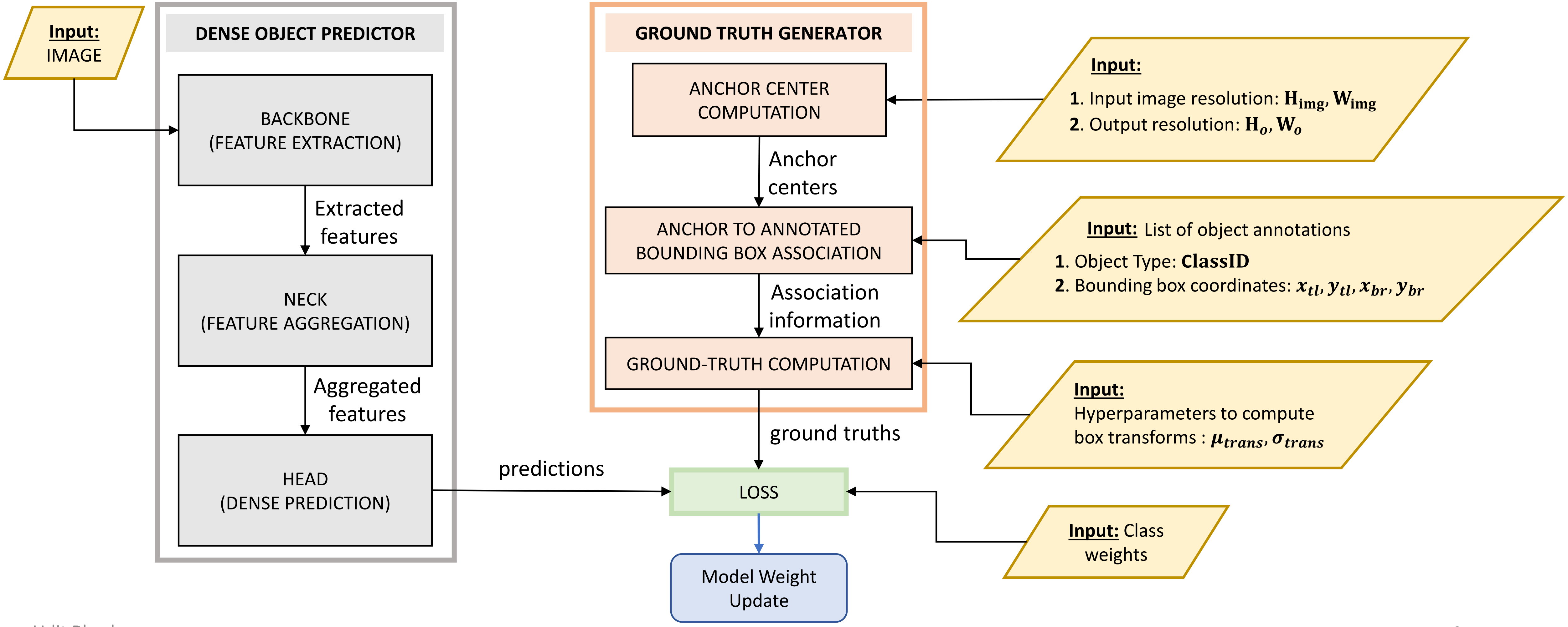
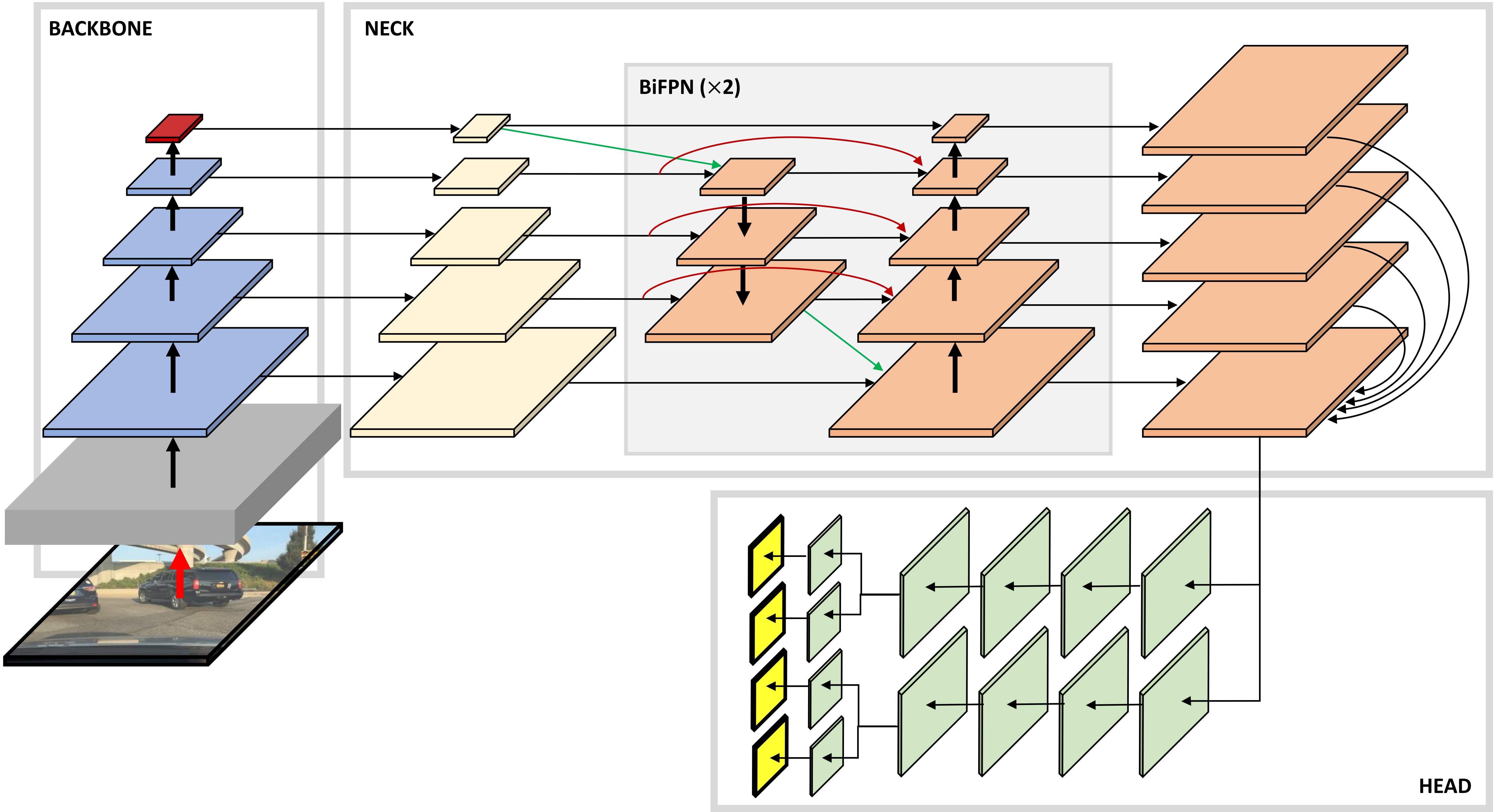
Anchor Free Single Stage 2D Object Detector

Table of Contents

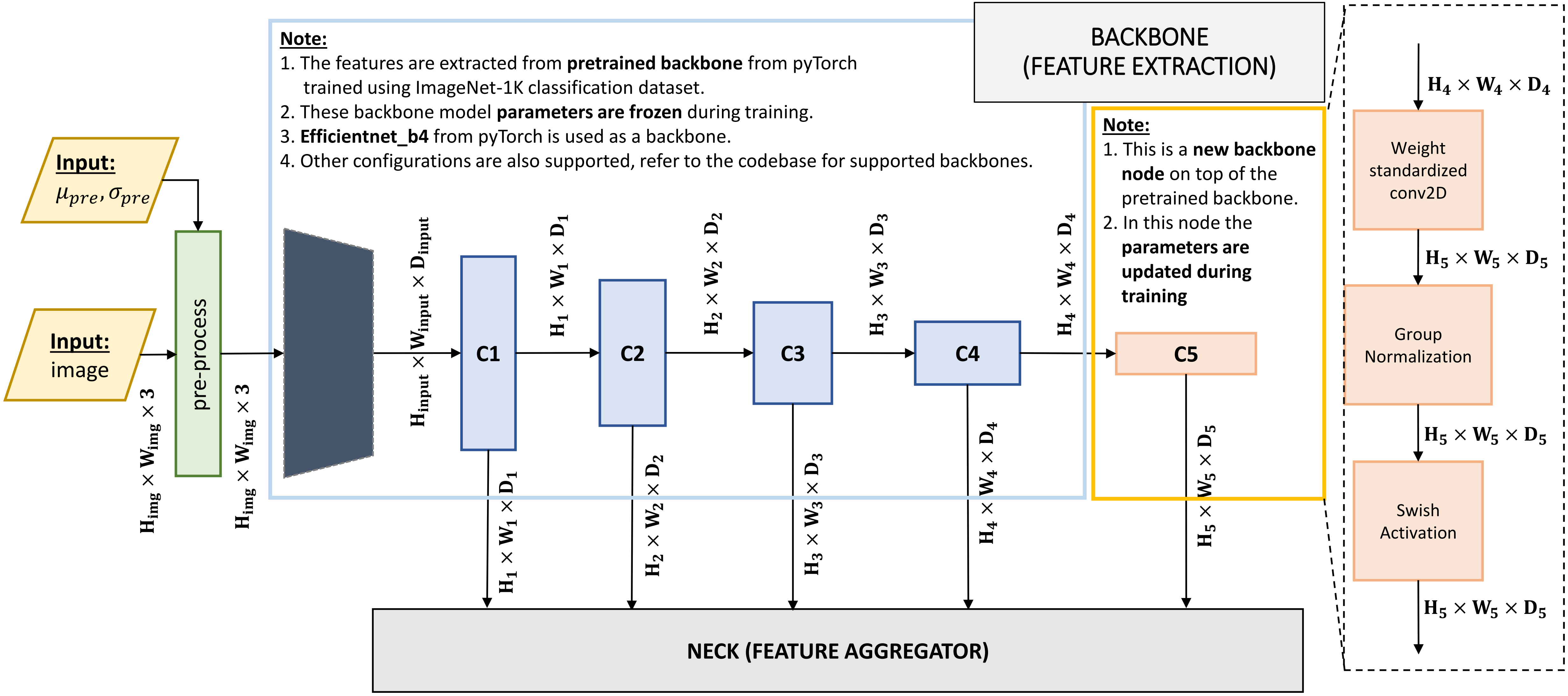
- ARCHITECTURE
  - High Level Design 1
  - High Level Design 2
  - Backbone
  - Feature Aggregator (Data Flow)
  - Feature Aggregator (Formulas)
  - Network Head
  - Architecture Summary
- GROUND-TRUTH
  - Anchor Centre Computation
  - Anchor to Annotated Bounding Box Association
  - Association Visualization
  - Box Offsets & Box Proposals
  - Centerness Score Formula
  - Centerness Score Visualization
  - Objectness & Classification Score
- LOSS FUNCTIONS
  - Loss Function Table
- NMS & SCORE THRESHOLD
  - Non Maximal Suppression Threshold Determination
  - ROC for NMS Threshold determination
  - Prediction Score Threshold Determination
  - ROC for Prediction Score Threshold determination

CONCEPT LEVEL ARCHITECTURE (DENSE PREDICTOR)

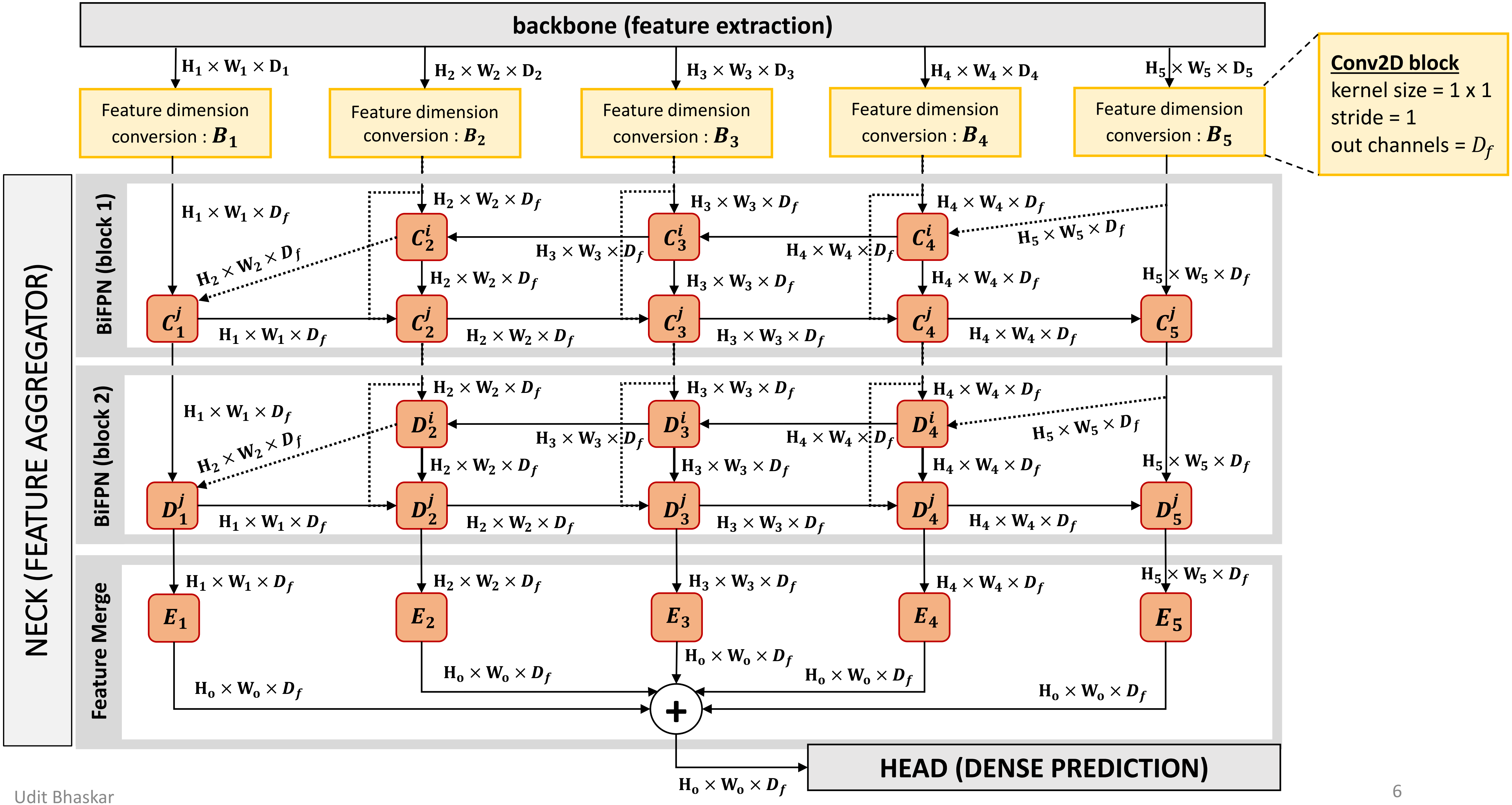




BACKBONE
NECK
BiFPN (×2)
HEAD



BACKBONE
(FEATURE EXTRACTION)
Note:
1. The features are extracted from pretrained backbone from pyTorch trained using ImageNet-1K classification dataset.
2. These backbone model parameters are frozen during training.
3. Efficientnet_b4 from pyTorch is used as a backbone.
4. Other configurations are also supported, refer to the codebase for supported backbones.
Input: μ_{pre}, σ_{pre}
Input: image
pre-process
$H_{img} \times W_{img} \times 3$
$H_{img} \times W_{img} \times 3$
$H_{input} \times W_{input} \times D_{input}$
C1
$H_1 \times W_1 \times D_1$
C2
$H_2 \times W_2 \times D_2$
C3
$H_3 \times W_3 \times D_3$
C4
$H_4 \times W_4 \times D_4$
Note:
1. This is a new backbone node on top of the pretrained backbone.
2. In this node the parameters are updated during training
C5
$H_5 \times W_5 \times D_5$
$H_4 \times W_4 \times D_4$
Weight standardized conv2D
$H_5 \times W_5 \times D_5$
Group Normalization
$H_5 \times W_5 \times D_5$
Swish Activation
$H_5 \times W_5 \times D_5$
NECK (FEATURE AGGREGATOR)



backbone (feature extraction)
H1 × W1 × D1
H2 × W2 × D2
H3 × W3 × D3
H4 × W4 × D4
H5 × W5 × D5
Feature dimension conversion : B1
Feature dimension conversion : B2
Feature dimension conversion : B3
Feature dimension conversion : B4
Feature dimension conversion : B5
Conv2D block
kernel size = 1 x 1
stride = 1
out channels = Df
NECK (FEATURE AGGREGATOR)
BiFPN (block 1)
H1 × W1 × Df
H2 × W2 × Df
H3 × W3 × Df
H4 × W4 × Df
H5 × W5 × Df
C2i
C3i
C4i
C1j
C2j
C3j
C4j
C5j
BiFPN (block 2)
D2i
D3i
D4i
D1j
D2j
D3j
D4j
D5j
Feature Merge
E1
E2
E3
E4
E5
Ho × Wo × Df
+
HEAD (DENSE PREDICTION)

Top down connections

$$c_4^i = f_4^i\left(w_1' B_4 + w_2' \mathrm{interp}(B_5)\right)$$

$$c_3^i = f_3^i\left(w_3' B_3 + w_4' \mathrm{interp}(c_4^i)\right)$$

$$c_2^i = f_2^i\left(w_5' B_2 + w_6' \mathrm{interp}(c_3^i)\right)$$

Bottom up connections

$$c_1^j = f_1^j\left(w_7' B_1 + w_8' \mathrm{interp}(c_2^i)\right)$$

$$c_2^j = f_2^j\left(w_9' B_2 + w_{10}' c_2^i + w_1' \mathrm{interp}(c_1^j)\right)$$

$$c_3^j = f_3^j\left(w_{11}' B_3 + w_{12}' c_3^i + w_{13}' \mathrm{interp}(c_2^j)\right)$$

$$c_4^j = f_4^j\left(w_{13}' B_4 + w_{14}' c_4^i + w_{16}' \mathrm{interp}(c_3^j)\right)$$

$$c_5^j = f_5^j\left(w_{15}' B_5 + w_{16}' \mathrm{interp}(c_4^j)\right)$$

Feature merge

$$E_1 = f_1^e\left(\mathrm{interp}(D_2^i)\right)$$

$$E_2 = f_2^e\left(\mathrm{interp}(D_1^j)\right)$$

$$E_3 = f_3^e\left(\mathrm{interp}(D_2^j)\right)$$

$$E_4 = f_4^e\left(\mathrm{interp}(D_3^j)\right)$$

$$E_5 = f_5^e\left(\mathrm{interp}(D_4^j)\right)$$

$$E = E_1 + E_2 + E_3 + E_4 + E_5$$

FEATURE AGGREGATOR FORMULAS

Weight Computation for merging three feature maps

$$w_i' = \frac{ReLU(w_i)}{ReLU(w_i) + ReLU(w_j) + ReLU(w_k) + \varepsilon}$$

$$w_j' = \frac{ReLU(w_j)}{ReLU(w_i) + ReLU(w_j) + ReLU(w_k) + \varepsilon}$$

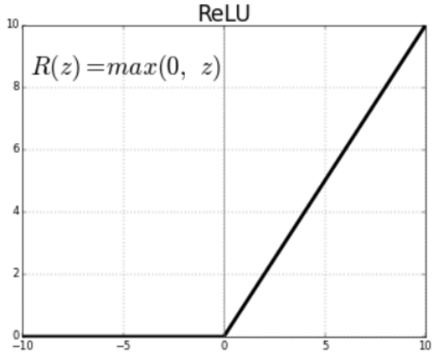
$$w_k' = \frac{ReLU(w_k)}{ReLU(w_i) + ReLU(w_j) + ReLU(w_k) + \varepsilon}$$

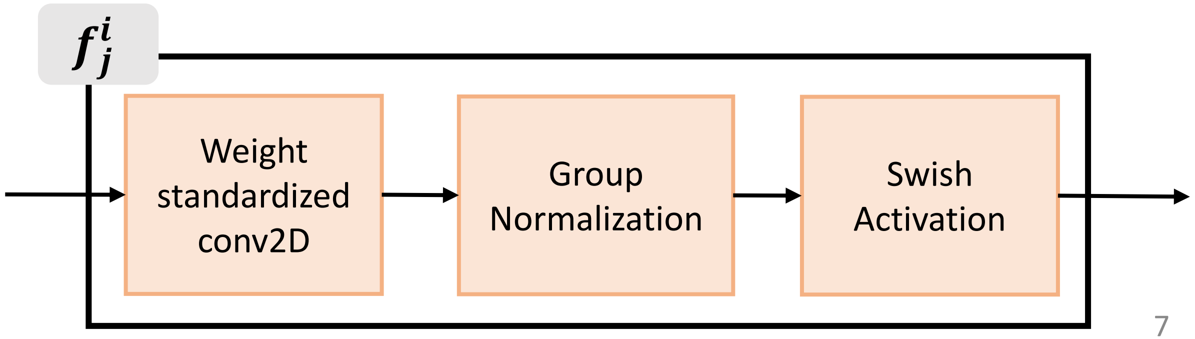
note:
$\boldsymbol{w_i, w_j, w_k}$ are **trainable** parameters

Weight Computation for merging two feature maps

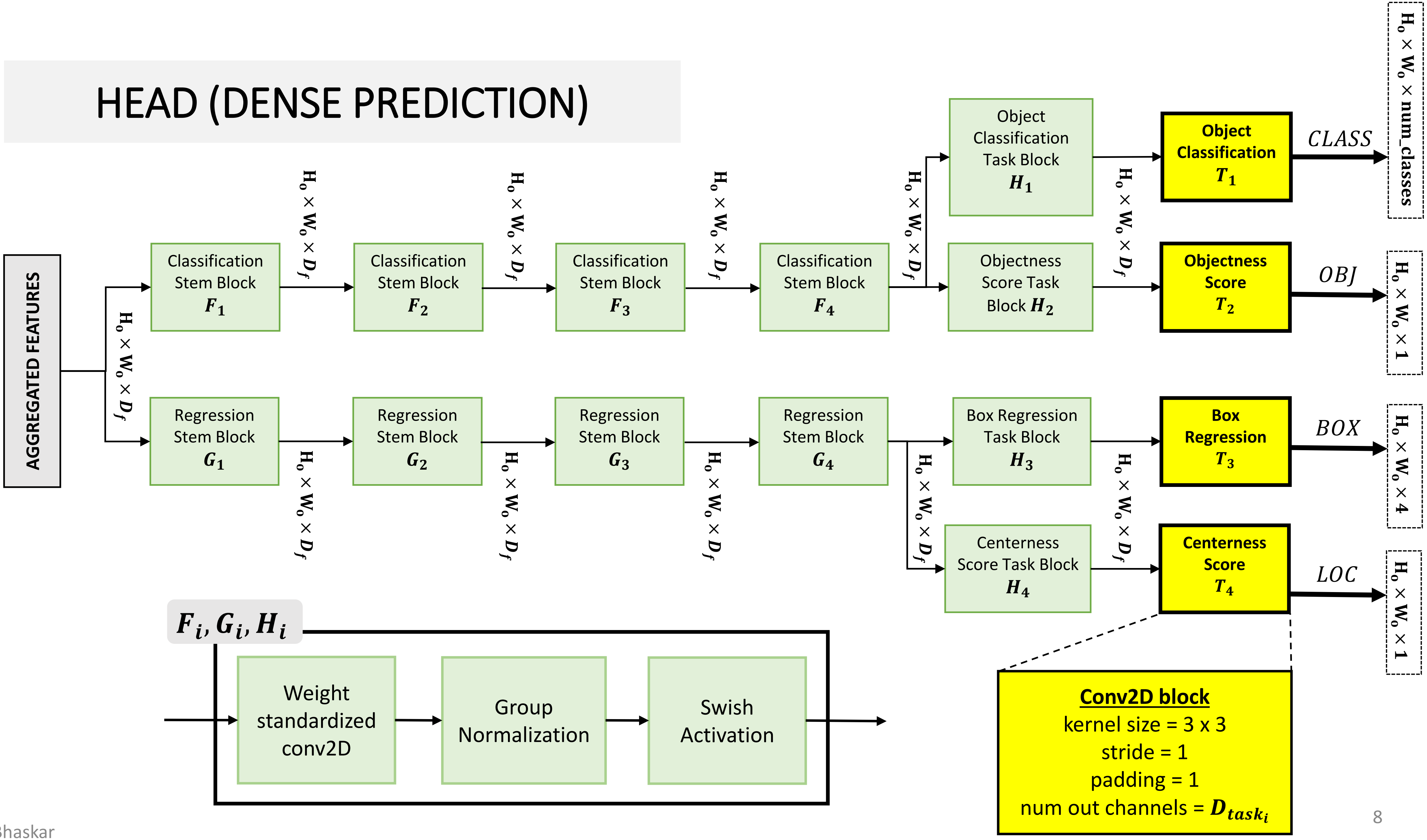
$$w_i' = \frac{ReLU(w_i)}{ReLU(w_i) + ReLU(w_j) + \varepsilon}$$

$$w_j' = \frac{ReLU(w_j)}{ReLU(w_i) + ReLU(w_j) + \varepsilon}$$

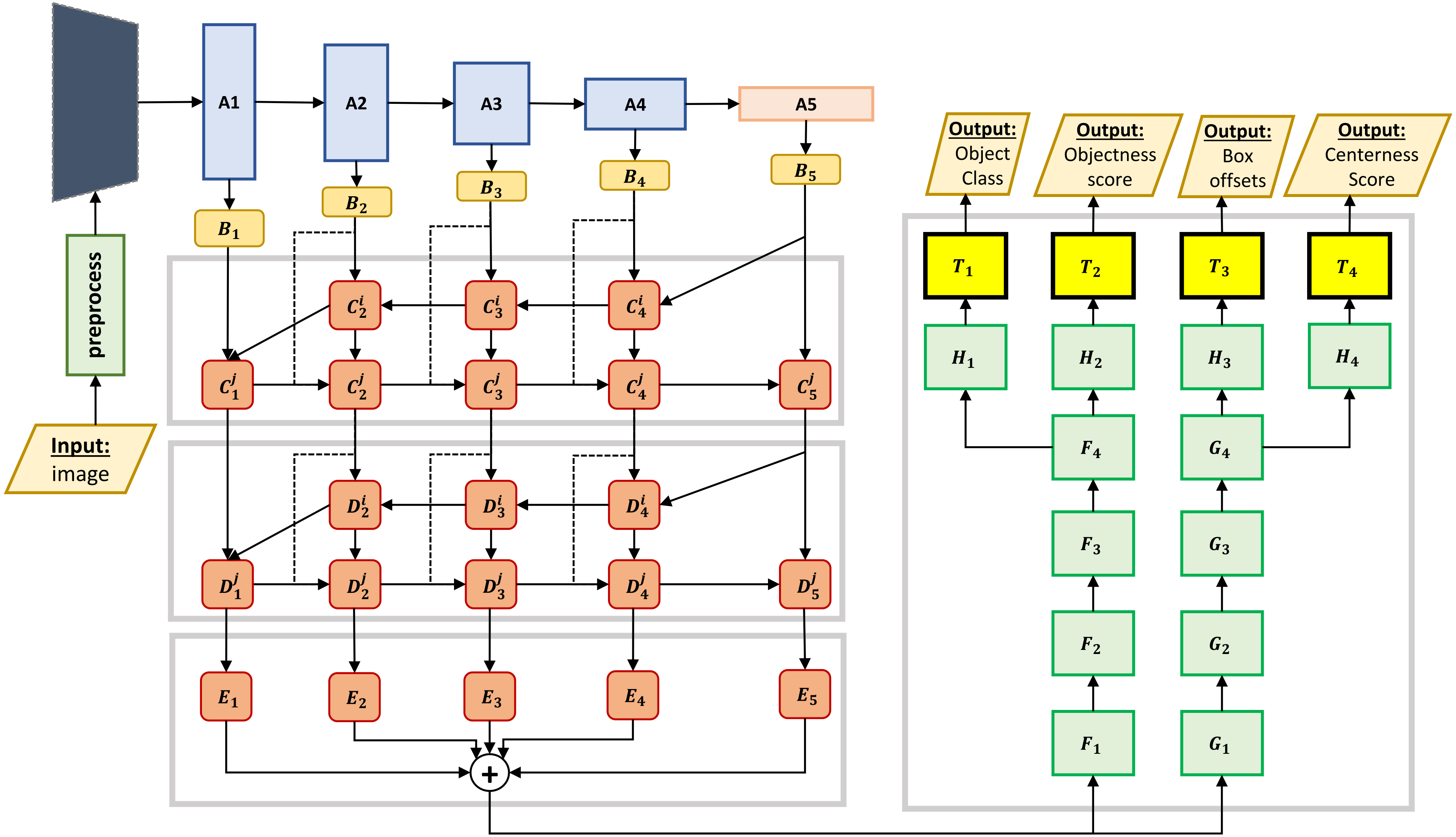




HEAD (DENSE PREDICTION)



ARCHITECTURE SUMMARY



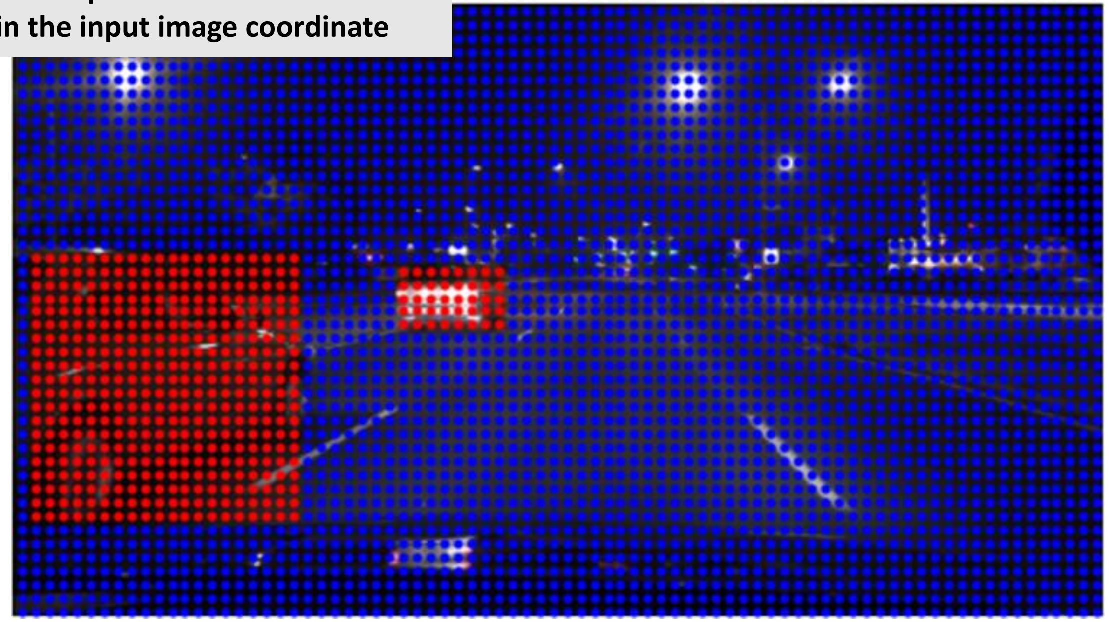
TOC

ANCHOR CENTER COMPUTATION

pytorch code snippet

```
xcoord = torch.arange(start=0, end=width,  step=1, dtype=torch.float32, device=device) + 0.5
ycoord = torch.arange(start=0, end=height, step=1, dtype=torch.float32, device=device) + 0.5
xcoord *= stride_width
ycoord *= stride_height
xcoord = xcoord.unsqueeze(0).repeat(height, 1)
ycoord = ycoord.unsqueeze(-1).repeat(1, width)
grid_coords = torch.stack((xcoord, ycoord), dim=-1).reshape(-1, 2)
```

Visual depiction of anchor centers in the input image coordinate



Input: $\mathbf{H_o, W_o}$

Generate a vector of x and y grid indexes in the output resolution $(\boldsymbol{H_o, W_o})$

$$x_{index} = (0,1,2,3,4, \dots, W_o - 1)$$
$$y_{index} = (0,1,2,3,4, \dots, H_o - 1)$$

Compute the grid center coordinates in the output resolution

$$x_{index} = x_{index} + 0.5$$
$$y_{index} = y_{index} + 0.5$$

Input : $\mathbf{H_{img}, W_{img}}$

Transform the grid coordinate from the output resolution to the input image resolution

$$s_w = \frac{W_{image}}{W_o}$$
$$s_h = \frac{H_{image}}{H_o}$$
$$x_{coord} = s_w * x_{index}$$
$$y_{coord} = s_h * y_{index}$$

Generate the whole grid by using tensor repeat and stack operations.

Output: *anchor center coordinates*

TOC

ANCHOR TO ANNOTATED BOUNDING BOX ASSOCIATION

Input:
Thresholds
$T_w, T_h,$
T_r^{low}, T_r^{high}

Input: List of object annotations
1. Object Type: **ClassID**
2. Bounding box coordinates: $x_{tl}, y_{tl}, x_{br}, y_{br}$

Select prominent ground truth annotations. Some of the annotated bounding boxes are either **invalid** or are **too small**, such annotations are removed by thresholding on the box aspect ratio and box dimension

$$\mathbf{w^i = x_2 - x_1\ , h^i = y_2 - y_1\ , ratio^i = w^i/h^i}$$

$$\mathbf{flag = \left(w^i \geq T_w\right) \& \left(h^i \geq T_h\right) \& \left(ratio^i \geq T_r^{low}\right) \& \left(ratio^i \leq T_r^{high}\right)}$$

if the **flag** is true the annotations are selected for association

Input:
anchor center coordinates

Each of the anchor centres are associated with the ground truth box using the following rules:
1) The anchor centre coordinates shall **lie within the annotated bounding box**.
2) If an anchor is associated with multiple boxes from rule 1 then choose that box where the **distance between the center of the box and the associated anchor coordinate is smallest.**
3) The **unassociated anchor centres** are marked as **negative** or **background.**

Compute association of the anchor centre coordinates with **all** the annotated bounding boxes. Mark the **associated anchor** centres as **'ignored'** and the **unassociated anchors** as **'negative'** or background.

Optionally one can **ignore the associated anchor coordinates near the boundary of the groundtruth box** to avoid 'confusing' the detector during training. To do that shrink the associated box by a certain fraction and compute if the previously associated anchor coordinates are outside the box. **If outside then mark it as 'negative'.**

Extract the positive anchors

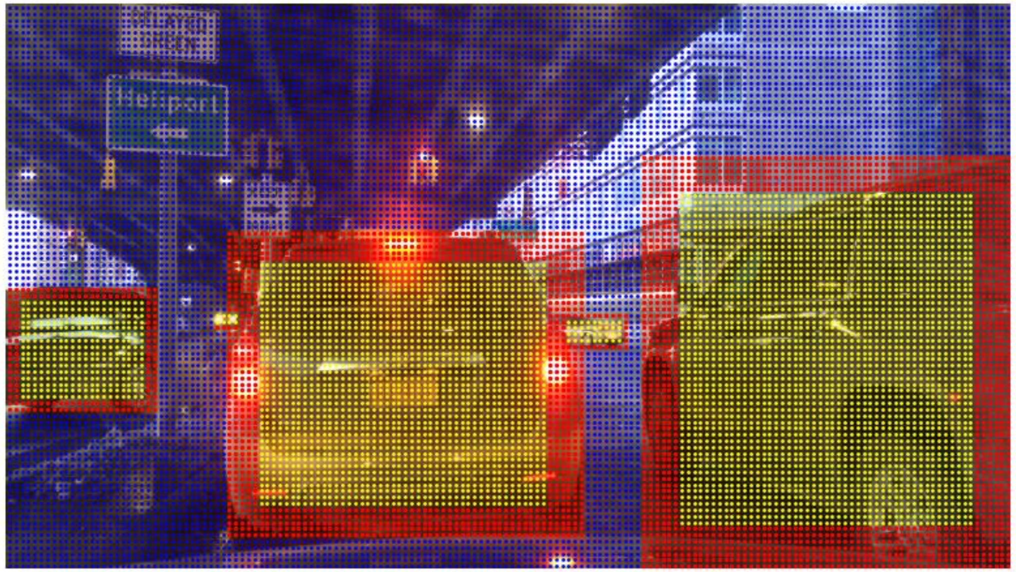
Overlay the extracted **'positive' anchors on top** of the **'ignored' anchors**

Output:
anchor to groundtruth associations

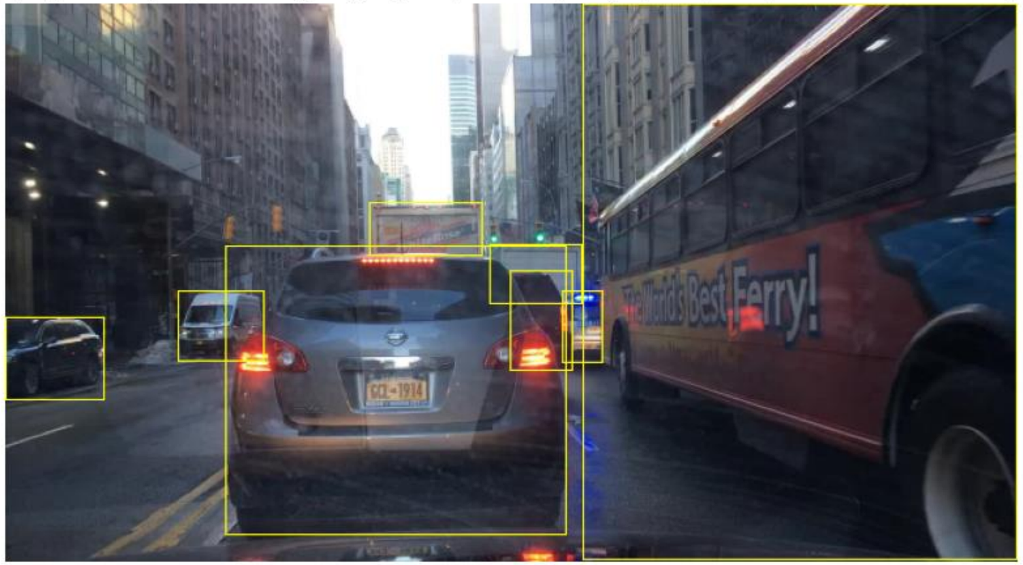
TOC

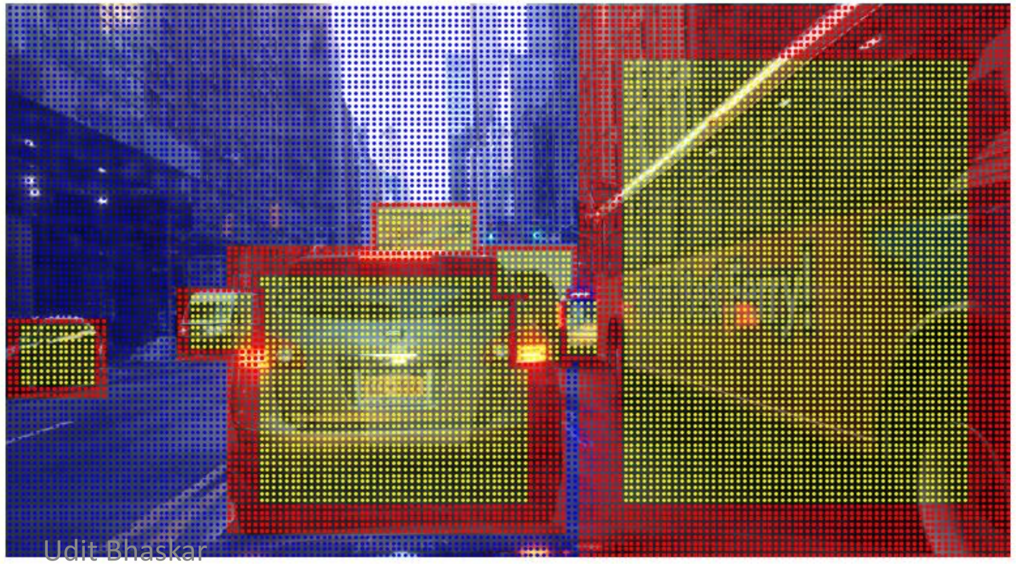
ASSOCIATION VISUALIZATION

../../..\../dataset/bdd/bdd100k_images_100k/bdd100k/images/100k/train\00091078-7cff8ea6.jpg



../../..\../dataset/bdd/bdd100k_images_100k/bdd100k/images/100k/train\00091078-59817bb0.jpg





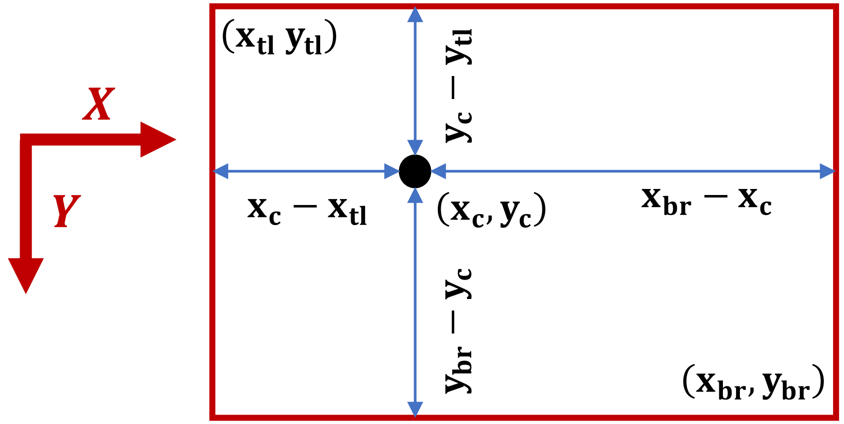
Positive

Ignored

Negative

BOX OFFSETS & BOX PROPOSALS

	Normalized Ground truth box offset	Proposal from predicted box offset
Box top left x coordinate	$t^x_{tl} = \dfrac{\log(x_c - x_{tl}) - \mu^x_{tl}}{\sigma^x_{tl}}$	$x_{tl} = x_c - \exp\left(\mu_{t^x_{tl}} + \sigma_{t^x_{tl}} * t^x_{tl}\right)$
Box top left y coordinate	$t^y_{tl} = \dfrac{\log(y_c - y_{tl}) - \mu^y_{tl}}{\sigma^y_{tl}}$	$y_{tl} = y_c - \exp\left(\mu_{t^y_{tl}} + \sigma_{t^y_{tl}} * t^y_{tl}\right)$
Box bottom right x coordinate	$t^x_{br} = \dfrac{\log(x_{br} - x_c) - \mu^x_{br}}{\sigma^x_{br}}$	$x_{br} = x_c + \exp\left(\mu_{t^x_{br}} + \sigma_{t^x_{br}} * t^x_{br}\right)$
Box bottom right y coordinate	$t^y_{br} = \dfrac{\log(y_{br} - y_c) - \mu^y_{br}}{\sigma^y_{br}}$	$y_{br} = y_c + \exp\left(\mu_{t^y_{br}} + \sigma_{t^y_{br}} * t^y_{br}\right)$



Note:

- The neural network **learns to predict** the offsets: $t^x_{tl}, t^y_{tl}, t^x_{br}, t^y_{br}$
- The **ground truth** for $t^x_{tl}, t^y_{tl}, t^x_{br}, t^y_{br}$ is computed from the assignment of ground truth bounding box to anchor centres
- $\mu_{t^x_{tl}}, \mu_{t^y_{tl}}, \mu_{t^w_{br}}, \mu_{t^h_{br}}$ and $\sigma_{t^x_{tl}}, \sigma_{t^x_{tl}}, \sigma_{t^w_{br}}, \sigma_{t^h_{br}}$ are the statistic of the **unnormalized** ground truth offsets
- The **ground truth** for $t^x_{tl}, t^y_{tl}, t^w_{br}, t^h_{br}$ is **normalized** such that the **ground truth offsets** have **zero mean** and **unit variance**
- $(x_c, y_c) \rightarrow$ anchor centre coordinates
- $(x_{tl}, y_{tl}, x_{br}, y_{br}) \rightarrow$ propoposal box corner coordinates computed from anchor centres and predicted offsets.

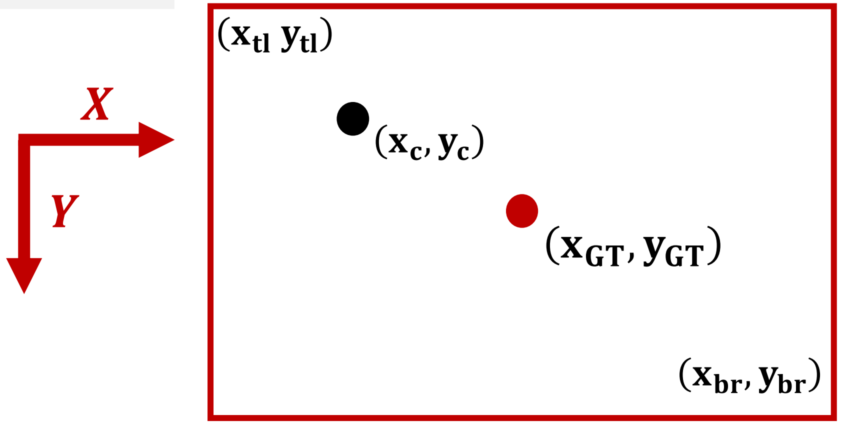
TOC

CENTERNESS SCORE

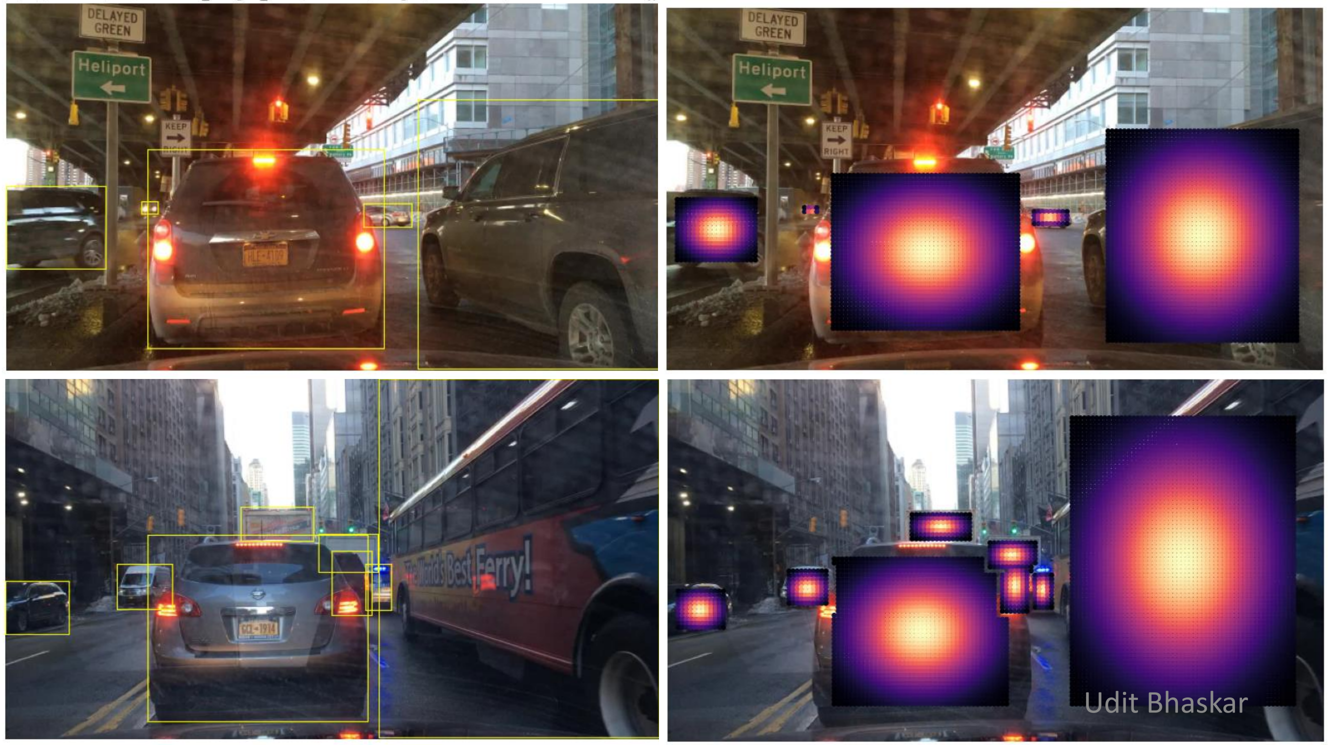
$$x_{GT} = \frac{x_{tl} + x_{br}}{2} \qquad y_{GT} = \frac{y_{tl} + y_{br}}{2}$$

$$W_{GT} = x_{br} - x_{tl} \qquad H_{GT} = y_{br} - y_{tl}$$

$$centerness_{GT} = \exp\left(-\frac{1}{2}\left(\left(\frac{x_c - x_{GT}}{s_w W_{GT}}\right)^2 + \left(\frac{y_c - y_{GT}}{s_h H_{GT}}\right)^2\right)\right)$$



../../..\../dataset/bdd/bdd100k_images_100k/bdd100k/images/100k/train\00091078-7cff8ea6.jpg



Note:

- The neural network **learns to predict** the centerness score which is in the range $[0, 1]$
- The **ground truth** for centerness is computed from the assignment of ground truth bounding box to anchor centres.
- It is hypothesised that the predicted outputs from the anchors that are more towards the centre of the object are more accurate.
- $(s_w, s_h) \rightarrow$ scaling factors to **adjust the spread** of the centerness ground-truth probability

OBJECTNESS & CLASSIFICATION SCORE

Input:
anchor to groundtruth associations

Compute Ground Truth for objectness score

Check if the anchor is associated with a ground truth object.

- If **associated** with the **ground truth**, objectness is set as **1**.
- If the anchor center is **associated** with the **background class**, set it as **0**.
- The **ignored anchors are not considered for training**

Output:
groundtruth objectness

Compute Ground Truth for classification score

Compute **one-hot** representation of the associated class label.
Anchors that are associated with **positive classes** are considered for **training**.

Output:
groundtruth class

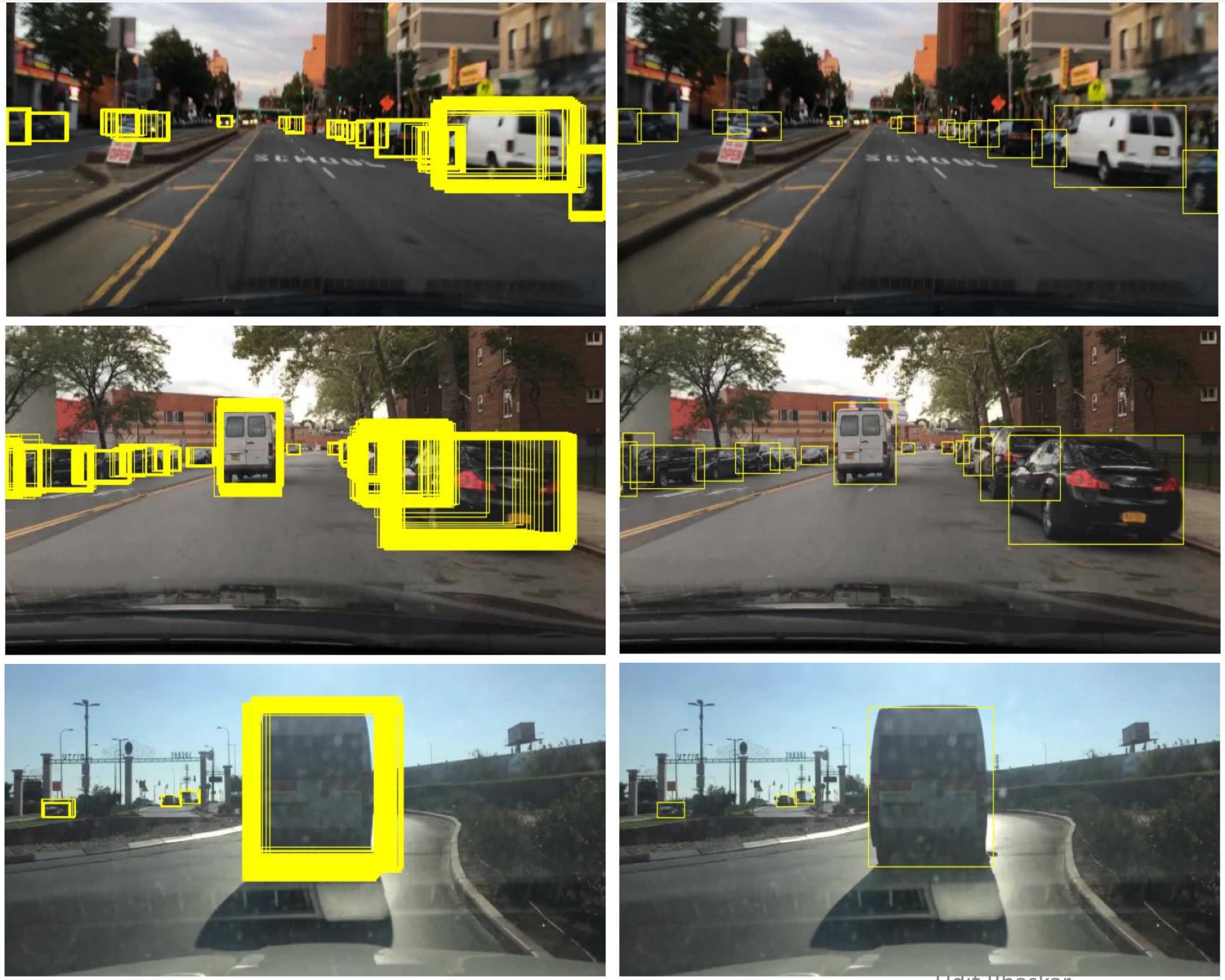
LOSS FUNCTIONS

TASK	LOSS FUNCTION
Object Classification	Class Weighted Cross Entropy Loss
Objectness	Focal Loss
Box Offset Regression	Smooth L1 Loss
Centerness Score Regression	Binary Cross Entropy Loss

TOC

NON MAXIMAL SUPRESSION :

Use ROC plot to determine the threshold for non maximal suppression



NMS Threshold determination

Inputs: bounding box predictions from the positive anchors

Inputs: NMS threshold to be considered **[0.1, 0.2, ... 0.9, 0.99]**

compute a **list of predicted bounding** boxes after applying the nms. For each **NMS threshold** do the following

Inputs: ground truth bounding box (annotations)

Using **greedy association** perform a bipartite matching between the set the **predicted** and **ground truth box** coordinates

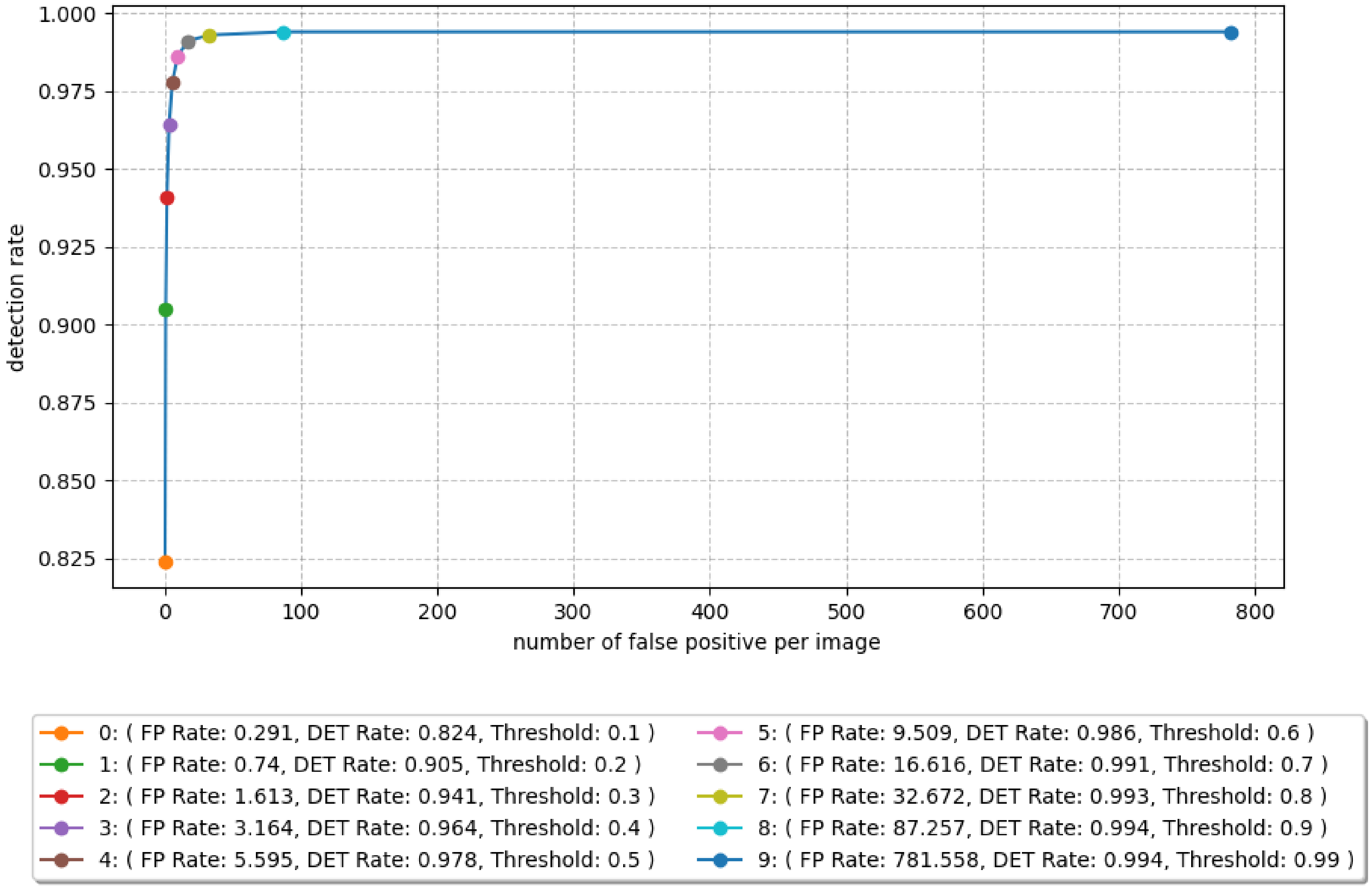
Compute **the total number of True positives and False Positives**. The predictions that are associated with the GT box are considered as true positives and the predictions that are not associated as False positives.

$$DetectionRate = \frac{total\ number\ of\ TPs}{tosl\ number\ of\ GT\ boxes}.$$

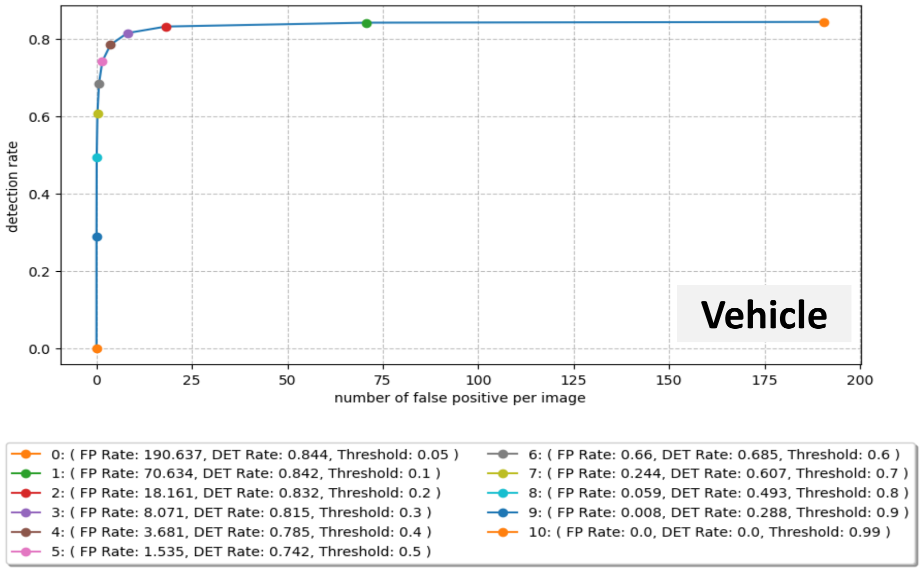
$$FalsePositiveRate = \frac{total\ number\ of\ FPs}{tosl\ number\ of\ images}.$$

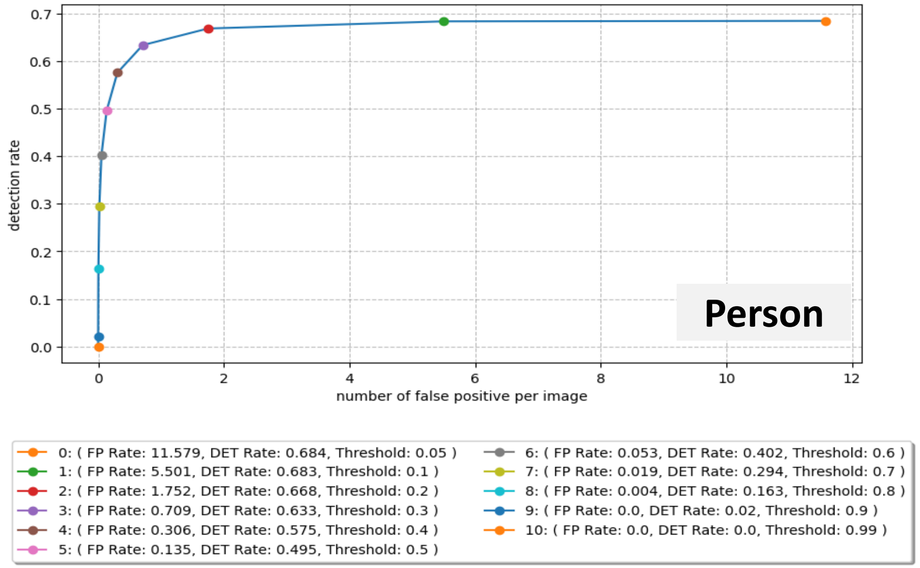
Output: ROC[thresholds]

ROC for NMS Threshold determination



ROC for Score Threshold determination





Inputs:
predicted bounding boxes from all anchors

Inputs:
1. detection thresholds to be considered
   [0.05, 0.1, 0.2, ... 0.9, 0.99]
2. Object class

compute a **list of predicted bounding** boxes after applying the nms.
For each **detection threshold** do the following

compute score for each prediction $score = \sqrt{prob_{cntr} * prob_{cls} * prob_{obj}}$
Select prediction **If** $score \geq Det_{thr}$

Using **greedy association** perform a bipartite matching between the set the **selected** and **ground truth box** coordinates

Inputs: ground truth bounding box (annotations)

Compute **the total number of True positives and False Positives**.
The predictions that are associated with the GT box are considered as true positives and the predictions that are not associated as False positives.

$$DetectionRate = \frac{total\ number\ of\ TPs}{tosl\ number\ of\ GT\ boxes}.$$

$$FalsePositiveRate = \frac{total\ number\ of\ FPs}{tosl\ number\ of\ images}.$$

Output: ROC[thresholds]