Intelligent Traffic System [1]

Pham Thanh Phong
Founder
ITM Vision
phongpham663@gmail.com

Ha Quang Phuoc
AI Engineer
ITM Vision
hqphuoc129@gmail.com

February 28, 2021

[1]Special thanks for the guidance of Msc.Vo Phi Son

CONTENTS

List of Figures

Chapter 1

OVERVIEW ITS SYSTEM

1.1 ITS

1.2 Hardware for Inference

Chapter 2

OBJECT DETECTION

2.1 Overview

2.1.1 Traditional Methods

Traditional object detection methods are built on handcrafted features and shallow trainable architectures.

Features used in traditional methods: color feature, HOG feature, edge feature, optical flow features, texture features,...

The pipeline of traditional object detection models can be mainly divided into three stages: informative region selection, feature extraction, and classification.

2.1.2 Deep Learning Based Method

2.1.2.1 ANN

An ANN consists of interconnected groups of nodes. Each group is called a layer and each node is called a neuron. The connection between neurons, replicating synapses in biological brains, transfers information between layers. A very simple ANN can be seen in where two inputs are going through a hidden layer in order to produce an output.

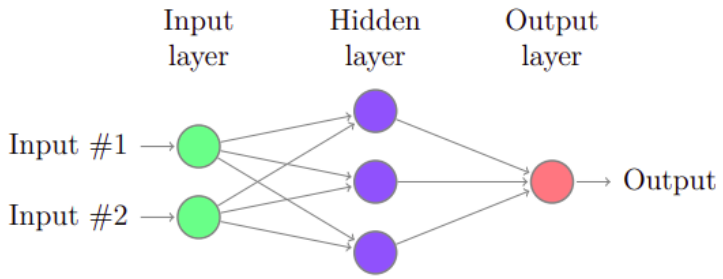

Figure 2.1: Example of an Artificial Neural Network with one input

2.1.2.2 Neurons

The artificial neurons in neural networks is a mathematical model of a biological neuron, modelled to replicate the function of being excitable depending on input signals. The equation of a neuron, visualized as one of the blue circles, is simply a weighted sum of all the inputs plus a bias term, i.e.

$$s(x) = \sum_{i=0}^{n} w_i x_i + b_i \tag{2.1}$$

where n is the number of inputs and w_i the weights in the neuron and b_i the bias term. The

combination of weights and input signals, giving the neuron its value together with the bias term, represents the excitability.

2.1.2.3 Activation Functions

The activation functions are essential components of ANNs, used to perform nonlinear mappings of the input data and are typically applied element-wise to all neurons in a hidden layer. This section describes a few commonly used activation functions and their properties. The activation functions commonly used in the intermediate layers of ANNs are presented.

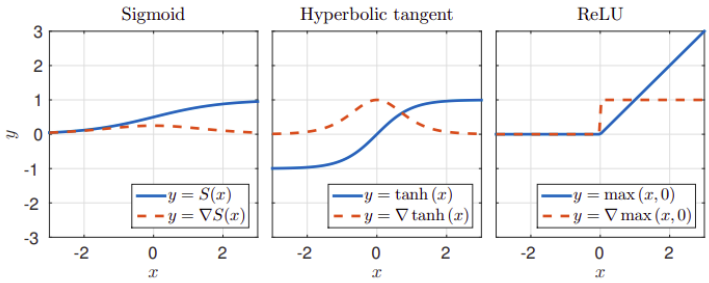

Figure 2.2: The sigmoid, hyperbolic tangent and ReLU activation functions and their derivatives

1. **Sigmoid**

   The sigmoid activation functions is defined as:

   $$S(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

   It takes values from 0 to 1 and is linear close to the origin. It has a "squashing" property such that large positive or negative input values result in values close to 1 or 0 respectively.

2. **Hyperbolic Tangent**

   The hyperbolic tangent:

   $$tan(x) = \frac{sin(x)}{cos(x)} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.3}$$

   is also commonly used as activation function in ANNs. It has very similar properties to the sigmoid function except tan(x) takes values from -1 to 1.

3. **Rectified Linear Unit**

   The Rectified Linear Unit (ReLU) is an activation function commonly used with deep neural networks, and is simply given by:

   $$f(x) = max(x, 0) \tag{2.4}$$

   Compared to the commonly used sigmoid activation function, the ReLU has the advantage of not saturating for large inputs. While sigmoidal functions take values in the range (0, 1), ReLUs take values in $[0, \infty)$ making them less prone to be either “on” or “off”.

4. **Softmax**

   The softmax function is a normalising function commonly used as an activation function for the last layer of a neural network. Given an input vector x with values x_i, the corresponding output element y_i is calculated.

   $$y_i = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \tag{2.5}$$

By construction, the elements of the output vector sums to 1. This property is useful when the output of the neural network should encode probabolities for different cases, such as in classification tasks.

2.1.2.4 CNNs Overview

2.1.2.5 Categorize

Thanks to Deep Neural Network, a more significant gain is obtained with the introduction of regions with convolutional neural network (CNN) features (R-CNN). DNNs, or the most representative CNNs, act in a quite different way from traditional approaches. They have deeper architectures with the capacity to learn more complex features than the shallow ones. Also, the expressivity and robust training algorithms allow to learn informative object representations without the need to design features manually.

Since the proposal of R-CNN, a great deal of improved models have been suggested, including fast R-CNN that jointly optimizes classification and bounding box regression tasks, faster R-CNN that takes an additional subnetwork to generate region proposals, and you only look once (YOLO) that accomplishes object detection via a fixed-grid regression.

2.1.2.6 Region Proposal Based Method

Generate region proposals at first and then classify each proposal into different object categories. Frameworks may be included such as: R-CNN, Faster R-CNN, region-based fully convolutional network R-FCN, feature pyramid networks (FPN), and Mask R-CNN.

Region proposal-based frameworks are composed of several correlated stages, including region proposal generation, feature extraction with CNN, classification, and bounding box regression, which are usually trained separately. Even in the recent end-to-end module Faster R-CNN, an alternative training is still required to obtain shared convolution parameters between RPN and detection network. As a result, the time spent in handling different components becomes the bottleneck in the real-time application.

2.1.2.7 Regression / Classification Based Method

Regression/classification based framework solves object detection problem by regarding it as regression or classification problem. Some frameworks may be accounted for examples are: MultiBox, AttentionNet, G-CNN, YOLO, Single Shot MultiBox Detector (SSD), YOLOv2, YOLOv3.

One-step frameworks based on global regression/classification, mapping straightly from image pixels to bounding box coordinates and class probabilities, can reduce time expense.

2.2 Comparison YOLO SSD Faster R-CNN

2.2.1 Faster-RCNN

2.2.1.1 Architecture

The architecture of Faster R-CNN is shown in the next figure. It consists of 2 modules:

- **RPN:** For generating region proposals.
- **Faster R-CNN:** For detecting objects in the proposed regions.

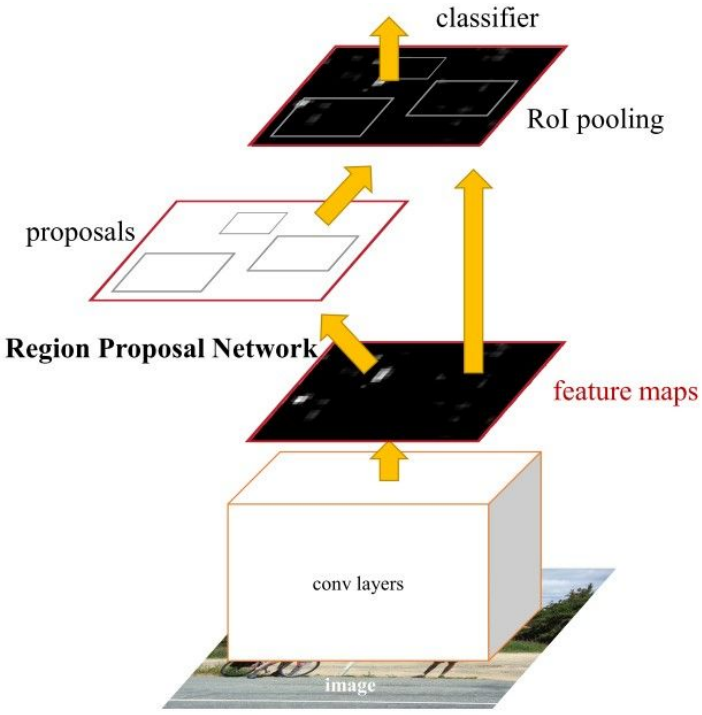

Figure 2.3: Pipeline of Faster R-CNN

2.2.1.2 Region Proposal Generation

The RPN module is responsible for generating region proposals. It applies the concept of attention in neural networks, so it guides the Fast R-CNN detection module to where to look for objects in the image.

The network slides over the conv feature map and fully connects to an n×n spatial window. A low-dimensional vector (512-dimensional for VGG16) is obtained in each sliding window and fed into two sibling FC layers, namely, box-classification layer (cls) and box-regression layer (reg). This architecture is implemented with an n × n conv layer followed by two sibling 1 × 1 conv layers. To increase nonlinearity, ReLU is applied to the output of the n × n conv layer.

The cls layer outputs a vector of 2 elements for each region proposal. If the first element is 1 and the second element is 0, then the region proposal is classified as background. If the second element is 1 and the first element is 0, then the region represents an object. In other words, it represents a binary classifier that generates the objectness score for each region proposal.

RPN produces better region proposals compared to generic methods like Selective Search and EdgeBoxes, which are implemented in RCNN and Fast - RCNN.

The RPN processes the image using the same convolutional layers used in the Fast R-CNN detection network. Thus, the RPN does not take extra time to produce the proposals compared to the algorithms like Selective Search and reduce training time as the RPN and the Fast R-CNN can be merged into a single network.

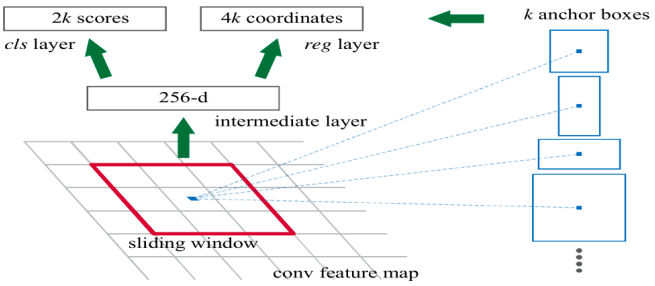

Figure 2.4: Architecture of RPN

2.2.1.3 Detection Head

1. **Anchors**

   The feature map of the last shared convolution layer is passed through a rectangular sliding window of size n*n where n = 3 for the VGG-16 net. For each window, K region proposals are generated. Each proposal is parametrized according to a reference box which is called an anchor box. The 2 parameters of the anchor boxes are scale and aspect ratio. Anchors of three scales and three aspect ratios are adopted, therefore K regions are produced from each region proposals. The multi-scale anchors are key to share features across the RPN and the Fast R-CNN detection network.

   For training the RPN, each anchor is given a positive or negative **objectness score** based on the Intersection-over Union (IoU).

   The next 4 conditions use the IoU to determine whether a positive or a negative **objectness score** is assigned to an anchor. Based on this, classification label is produced.

   - An anchor that has an IoU overlap higher than **0.7** with any ground-truth box is given a positive objectness label.
   - If there is no anchor with an IoU overlap higher than **0.7**, then assign a positive label to the anchor(s) with the highest IoU overlap with a ground-truth box.
   - A negative **objectness score** is assigned to a **non-positive** anchor when the IoU overlap for all ground-truth boxes is less than **0.3**. A negative objectness score means the anchor is classified as background.
   - Anchors that are neither positive nor negative do not contribute to the training objective.

2. **Loss Function**

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_c ls(p_i, p_i*) + \lambda \frac{1}{N_{reg}} \sum_i p_i * L_{reg}(t_i, t_i*) \tag{2.6}$$

   Where p_i is the predicted probability of the i^{th} anchor being an object. The ground truth label

   p_i* is 1 if the anchor is positive, otherwise 0. t_i stores four parameterized coordinates of the predicted boudaring box while t_i* is related to the ground truth box overlapping with a positive anchor. $L_c ls$ is a binary log loss and $L_r eg$ is a smoothed L_1 loss. These two terms are normalized with the mini-batch size ($N_c ls$) and the number of anchor locations ($N_r eg$).

2.2.1.4 Observations

1. **Pros**

   With the proposal of Faster R-CNN, region proposal-based CNN architectures for object detection can really be trained in an end-to-end way.

2. **Cons**

However, the alternate training algorithm is very time-consuming and RPN produces object-like regions (including backgrounds) instead of object instances and is not skilled in dealing with objects with extreme scales or shapes.

2.2.2 SSD

2.2.2.1 Feature Extraction

SSD uses VGG16 to extract feature maps. VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It was one of the famous model submitted to ILSVRC-2014. It makes the improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. It is one of the most preferred choices in the community for extracting features from images. The configuration of the VGGNet is publicly available and has been used in many other applications and challenges as a baseline feature extractor.

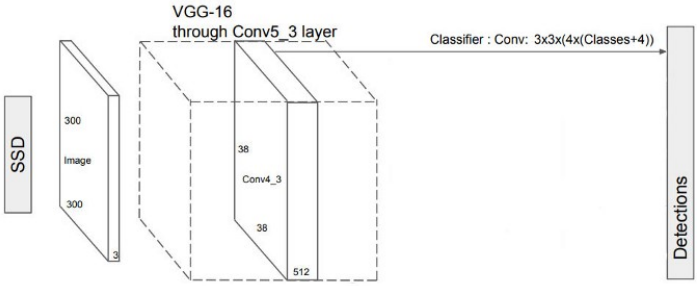

Figure 2.5: Feature Extractor of SSD

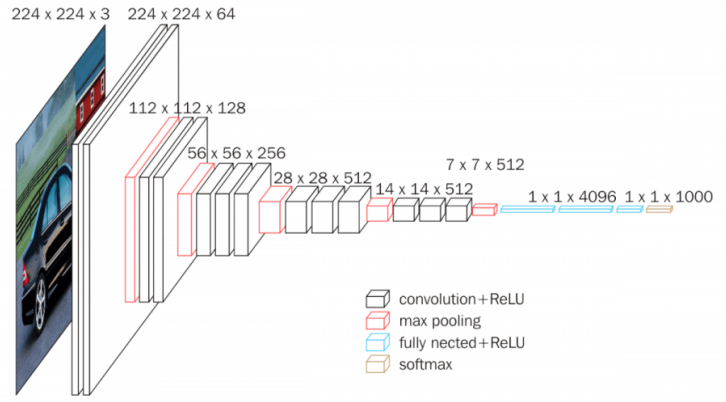

Figure 2.6: Architecture of VGG16

2.2.2.2 Multi-scale feature maps for detection

Given a specific feature map, instead of fixed grids adopted in YOLO, SSD uses bounding box regression technique,it takes the advantage of a set of default anchor boxes with different aspect ratios and scales to discretize the output space of bounding boxes. To handle objects with various sizes, the network fuses predictions from multiple feature maps with different resolutions.

SSD adds several feature layers to the end of VGG16 backbone network to predict the offsets to default anchor boxes and their associated confidences. Final detection results are obtained by conducting NMS on multiscale refined bounding boxes.

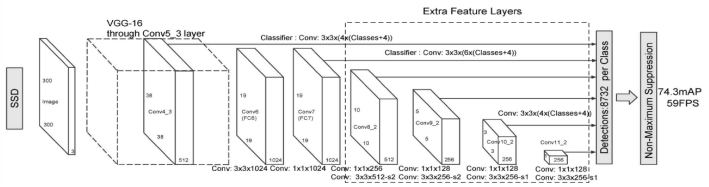

Figure 2.7: Architecture of SSD

2.2.2.3 Default Boundary Box

The default boundary boxes are equivalent to anchors in Faster R-CNN. Default boundary boxes are chosen manually. SSD defines a scale value for each feature map layer. Combining the scale value with the target aspect ratios, we compute the width and the height of the default boxes. For layers making 6 predictions, SSD starts with 5 target aspect ratios: 1, 2, 3, 1/2, and 1/3. Then the width and the height of the default boxes are calculated as:

$$w = scale.\sqrt{aspectratio} \tag{2.7}$$

$$h = \frac{scale}{\sqrt{aspectratio}} \tag{2.8}$$

Then SSD adds an extra default box with scale:

$$scale = \sqrt{scale.scaleatnextlevel} \tag{2.9}$$

Where aspect ratio is equal to 1.

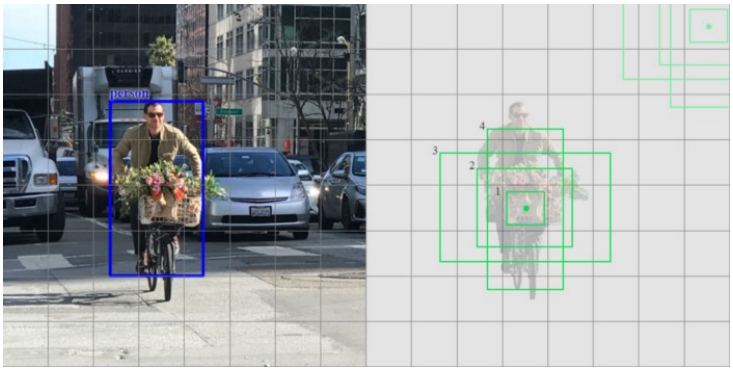

Figure 2.8: Default Boundary Box

2.2.2.4 Matching Strategy

SSD predictions are classified as positive matches or negative matches. SSD only uses positive matches in calculating the localization cost (the mismatch of the boundary box). If the corresponding default boundary box (not the predicted boundary box) has an IoU greater than 0.5 with the ground truth, the match is positive. Otherwise, it is negative. Once we identify the positive matches, we use the corresponding predicted boundary boxes to calculate the cost. This matching strategy nicely partitions what shape of the ground truth that a prediction is responsible for.

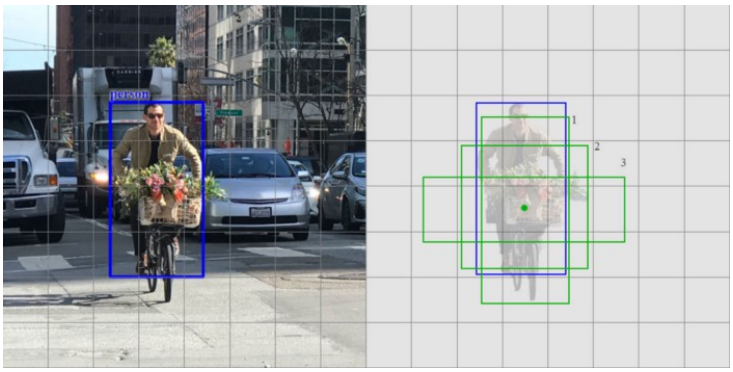

Figure 2.9: Matching with ground truth

2.2.2.5 Loss Function

There are two types of loss functions here: Confidence loss and Location loss.

- There are two types of loss functions here: Confidence loss and Location loss.

  The localization loss between the predicted box **l** and the ground truth box **g** is defined as the smooth L_1 loss with c_x, c_y as the offset to the default bounding box **d** of width **w** and height **h**.

$$L_loc(x,l,g) = \sum_{i \in Pos}^{N} \sum_{m \in c_x, c_x, w, h} x_{ij}^{k} smooth_{L_1}(l_i^m - \hat{g}_j^m) \tag{2.10}$$

$$\hat{g}_j^{c_x} = \frac{(g_j^{c_x} - d_i^{c_x})}{d_i^w}; \hat{g}_j^{c_y} = \frac{(g_j^{c_y} - d_i^{c_y})}{d_i^h} \tag{2.11}$$

$$\hat{g}_j^w = \log(\frac{g_j^w}{d_i^w}); \hat{g}_j^h = \log(\frac{g_j^h}{d_i^h}) \tag{2.12}$$

$$x_{ij}^p = \begin{cases} 1 & \text{if IoU} > 0.5 \text{ between default box i and ground true box j on class p} \\ 0 & \text{Otherwise} \end{cases} \tag{2.13}$$

- The confidence loss is a measure of the confidence that an algorithm quantifies if a bounding box in an image consists of any object or class. For every positive match prediction, we penalize the loss according to the confidence score of the corresponding class. For negative match predictions, we penalize the loss according to the confidence score of the class "0": class "0" classifies no object is detected.The alpha term balances the contribution of the location losses. The main objective in neural neutral is to analyse the parameters that reduce the loss predictions.

  It is calculated as the softmax loss over multiple classes confidences c (class score).

$$L_{conf}(x,c) = -\sum_{i \in Pos}^{N} x_{ij}^p \log(\hat{c}_i^0) where \hat{c}_i^p = \frac{exp(c_i^p)}{(\sum)_p exp(c_i^p)} \tag{2.14}$$

  where N is the number of matched default boxes.

- The final loss function is computed as:

$$L(x,c,l,g) = \frac{1}{N}(L_{conf}(x,c) + \propto L_{loc}(x,l,g)) \tag{2.15}$$

  where N is the number of positive matches and α is the weight for the localization loss.

2.2.2.6 Hard negative mining

SSD still requires negative sampling so it can learn what constitutes a bad prediction. So, instead of using all the negatives, we sort those negatives by their calculated confidence loss. SSD picks the negatives with the top loss and makes sure the ratio between the picked negatives and positives is at most 3:1. This leads to faster and more stable training.

2.2.2.7 Observations

1. **Pros**

   SSD is a single-shot detector. It has no delegated region proposal network and predicts the boundary boxes and the classes directly from feature maps in one single pass.

   To improve accuracy, SSD introduces:

   - Small convolutional filters to predict object classes and offsets to default boundary boxes.
   - Separate filters for default boxes to handle the difference in aspect ratios.
   - Multi-scale feature maps for object detection.

   SSD can be trained end-to-end for better accuracy. SSD makes more predictions and has better coverage on location, scale, and aspect ratios. With the improvements above, SSD can lower the input image resolution to 300×300 with a comparative accuracy performance. Integrating with hard negative mining, data augmentation, and a larger number of carefully chosen default anchors, SSD significantly outperforms the Faster R-CNN in terms of accuracy on PASCAL VOC and COCO while being three times faster.

2. **Cons**
   Shallow layers in a neural network may not generate enough high level features to do prediction for small objects. Therefore, SSD does worse for smaller objects than bigger objects.

   The need of complex data augmentation also suggests it needs a large number of data to train. For example, SSD does better for Pascal VOC if the model is pretrained on COCO dataset.

2.2.3 YOLO

2.2.3.1 Network Architecture

The whole system can be divided into two major components: Feature Extractor and Detector; both are multi-scale. When a new image comes in, it goes through the feature extractor first so that we can obtain feature embeddings at three (or more) different scales. Then, these features are feed into three (or more) branches of the detector to get bounding boxes and class information.

Figure 2.10: Major components

2.2.3.2 Feature Extraction

The feature extractor YOLO V3 uses is called Darknet-53. Darknet-53 contains 53 layers and borrows the ideas of skip connections to help the activations to propagate through deeper layers without gradient diminishing from ResNet. But the Darknet-53 claims to be more efficient than ResNet101 or ResNet152.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 2.11: Feature Extraction of YOLOv3

Inside the block, there's just a bottleneck structure (1x1 followed by 3x3) plus a skip connection. If the goal is to do multi-class classification as ImageNet does, an average pooling and a 1000 ways fully connected layers plus softmax activation will be added. However in the case of object detection, the classification head won't be included, instead, the "detection" head will be added to this feature extractor. Features from last three residual blocks are used in the later detection.

2.2.3.3 Multi-scale Detector

1. **IOU (Intersection Over Union)**

   Intersection over Union is an evaluation metric used to measure the accuracy of an object detector on a particular dataset.

   Intersection over Union is a ratio. In the numerator we compute the area of overlap between the predicted bounding box and the ground-truth bounding box The denominator is the area of union, or more simply, the area encompassed by both the predicted bounding box and the ground-truth bounding box. Dividing the area of overlap by the area of union yields our final

score — the Intersection over Union.

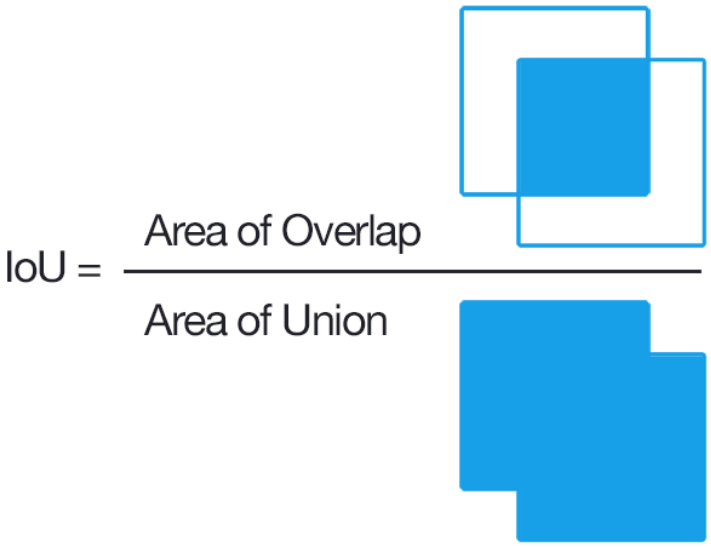

Figure 2.12: IOU formula

2. **Anchor Box**

The goal of object detection is to get a bounding box and its class. Bounding box usually represents in a normalized xmin, ymin, xmax, ymax format.

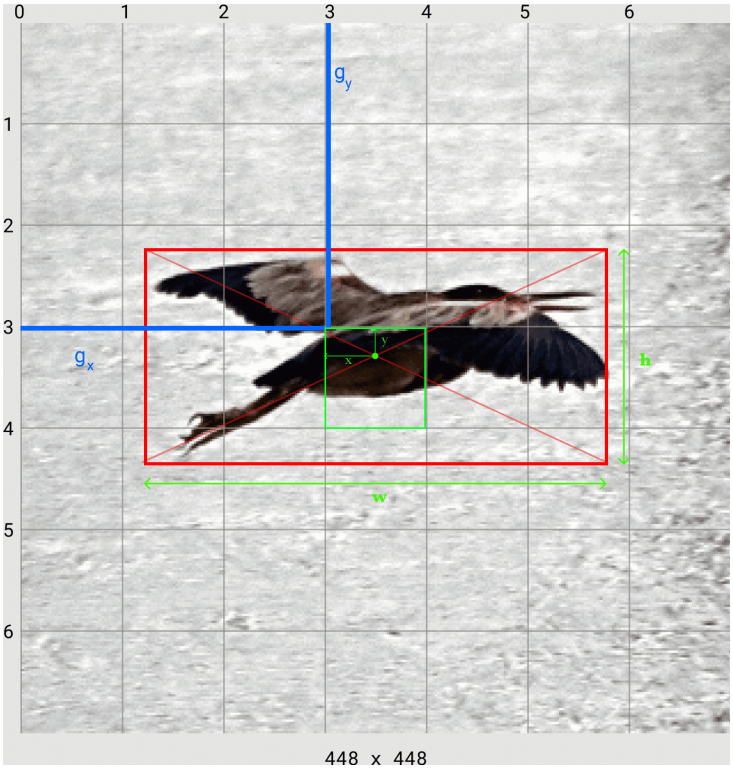

Figure 2.13: Example for anchor box

The input image is divided into an S x S grid of cells. For each object that is present on the image, one grid cell is said to be "responsible" for predicting it. Anchor boxes are assigned to each cell of

the grid. And once we defined those anchors, we can determine how much does the ground truth box overlap with the anchor box and pick the one with the best IOU and couple them together.

In YOLO v3, we have three anchor boxes per grid cell. And we have three scales of grids:

- The location offset against the anchor box: tx,ty,tw,th. This has 4 values.
- The confidence score to indicate if this box contains an object. This has 1 value. The confidence score is defined as $P_{object} * IOU_{pred}^{truth}$ which indicates how likely there is exist objects ($P_{object} \geq 0$) and show confidence of its prediction (IOU_{pred}^{truth}).
- The conditional class probabilities $P(Class_i|Object)$ to tell us which class this box belongs to. This has number of values according to number of classes.

3. **Loss Function**

During training the following loss function introduced in YOLO paper is optimize.

$$\lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(x_i - \hat{x_i})^2 + (y_i - \hat{y_i})^2] \tag{2.16}$$

$$\lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w_i}})^2 + (\sqrt{h_i} - \sqrt{\hat{h_i}})^2] \tag{2.17}$$

$$\sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (C_i - \hat{C_i})^2 \tag{2.18}$$

$$\lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} (C_i - \hat{C_i})^2 \tag{2.19}$$

$$\sum_{i=0}^{s^2} \mathbb{1}_{i}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p_i}(c))^2 \tag{2.20}$$

The loss function contain 4 parts : centroid loss, anchor box's width and height loss, confidence loss, classification loss.

❒**Centroid Loss:** the smaller this loss is, the closer the centroids of predictions and ground truth are. Since this is a regression problem, we use mean square error here. Besides, if there're no object from the ground truth for certain cells, we don't need to include the loss of that cell into the final loss. Therefore we also multiple by object mask. Object mask is either 1 or 0, which indicates if there's an object or not.

❒**Anchor's box width and height loss:** this term penalizes the bounding box with inacurate height and width. The square root is present so that erors in small bounding boxes are more penalizing than errors in big bounding boxes.

❒**Confidence loss:** tries to make the confidence score equal to the IOU between the object and the prediction when there is one object or close to 0 when there is no object in the cell.

❒**Classification loss:** is calculated by binary cross-entropy loss.

In YOLOv3, they made some modifications to the loss function, confidence loss now uses binary cross-entropy loss to calculated instead of mean square error. The classification loss is now changed into multi-label classification instead of multi-class classification, therefore independent logistic classifiers replaced the softmax for better class prediction.Because some dataset may contains labels that are related to each other.

2.2.3.4 Post Processing

The final component in this detection system is a post-processor. In YOLO, non maximum suppression is used to eleminate duplicate results.

2.2.3.5 Modifications of YOLO

For older version of YOLO, spatial constraint is one major drawback of the algorithm as a grid can analyse only up to two blocks and these two blocks can contain only one class. This results in a reduction in the detection of the nearby objects. It is quite difficult to detect small images in a group image with the help of this algorithm. As it uses the last-stage feature map, which has only the coarse information as it passes through the neural network, the accuracy has a limitation.

Bochkovskiy et al proposed the YOLOv4algorithm with significant changes from the previous version, and much better accuracy. Published in April 2020 it is the latest, most advanced iteration of YOLO, and the first developed by the original authors (Redmon et al). To achieve better accuracy, they designed a deeper and complex network, where they used Dense Block. It contains multiple convolutional layers, with batch normalization, ReLU, after which convolution takes place.

For the backbone of the feature extraction, they used the CSPDarknet-53, which uses the CSP connections along with Darknet-53 from the previous YOLOv3. Spatial Pyramid Pooling (SPP) is used as the neck over CSPDarknet-53 as it increases the receptive field, differentiates the most significant feature and does not cause a reduction in speed. In place of the Feature Pyramid Network (FPN) in YOLOv3, here they used Path Aggregation Network (PANet). For the head, they used the original YOLOv3 network.

Apart from the architecture, it consists of a training strategy to get better accuracy without the extra cost to hardware, which is termed as Bag of Freebies. With these, we can get better performance for "free". Another set of strategies which give better results at low cost, but not completely free, were termed as the Bag of Specials.

With those modification, YOLO is the most commonly used algorithm that is used for detecting natural images as this is the most efficient algorithm for new domains and inputs that are unexpected.

2.2.4 Comparision YOLO, SSD and Faster R-CNN

2.2.4.1 Pascal VOC 2007/2012

As YOLO is not skilled in producing object localizations of high IoU, it obtains a very poor result on VOC 2012. However, with the complementary information from Fast R-CNN (YOLO+FRCN) and the aid of other strategies, such as anchor boxes, BN, and fine-grained features, the localization errors are corrected (YOLOv2).

2.2.4.2 Microsoft COCO

Overall, region proposal-based methods, such as Faster R-CNN and R-FCN, perform better than regression/ classification-based approaches, namely, YOLO and SSD, due to the fact that quite a lot of localization errors are produced by regression/classification-based approaches.

2.2.4.3 Time analysis

Regression-based models can usually be processed in real time at the cost of a drop in accuracy compared with region proposal-based models. Also, region proposal-based models can be modified into real-time systems.

2.2.4.4 Application in Vehicle Detection System

Lecheng Ouyang et al implemented vehicle target detection based on YOLOv3 in complex scenes and showed advantages over traditional target detection algorithms in accuracy and speed. They showed how YOLOv3 can be used for vehicle detection, and it gives an accuracy of 89.16% at 21fps on the VOC dataset.

Jeong-ah Kim et al has put three algorithm into test by the the vehicle type classification for the vehicle type recognition was based on the classification of the Korea Expressway Corporation. They found out that Faster R-CNN may not suitable for real-time application, SSD's accuracy is low and sometimes fails to detect a vehicle while YOLOv4 yeilds the most efficient results.[7]

2.2.4.5 Other Experiments

Deepa et al compared all three models for real-time tennis ball tracking and from their work, they concluded that SSD is much efficient and comparatively more accurate accurate algorithm with less computation speed for this particular task of detecting the tennis ball tosses. It worth noticing that, the YOLO version they used is version 1, which performs badly with small objects like tennis ball. This is no longer a shortage to the latest version of YOLO [6].

According to [**], when they compared the YOLOv3 algorithm with the SSD, it was shown by evaluation that YOLOv3 had better performance than SSD. If the SSD has an input resolution at 300*300, it would have the same inference as the YOLOv3 at the input resolution 416*416, the precision of YOLOv3 is higher. YOLOv3 is faster than SSD, which is applicable to real-time.

2.2.4.6 Conclusion

Choosing YOLOv4 for ITS system is considered most efficient choice so far, therefore we use YOLOv4 to operate as a detector for our system.

2.3 Metrics and Evaluations

2.3.1 Average Precision(AP) and Mean Average Precision(mAP)

- Recall is the Ratio of the correct predictions and the total number of correct items in the set. It is expressed as percentage of the total correct(positive) items correctly predicted by the model. In other words, recall indicates how good is the model at picking the correct items.

$$Recall = \frac{TP}{TP + FN} \tag{2.21}$$

- Precision is measured over the total predictions of the model. It is the ratio between the correct predictions and the total predictions. In other words, precision indicates how good the model is at whatever it predicted.

$$Precision = \frac{TP}{TP + FP} \tag{2.22}$$

- To get True Positives(TP) and False Positives(FP), we use IoU. Using IoU, we now have to identify if the detection(a Positive) is correct(True) or not(False). The most commonly used threshold is 0.5 — i.e. If the IoU is greater than 0.5, it is considered a True Positive, else it is considered a false positive. The COCO evaluation metric recommends measurement across various IoU thresholds, but for simplicity, we will stick to 0.5, which is the PASCAL VOC metric.

- Since every part of the image where we didn't predict an object is considered a negative, measuring "True" negatives(TN) is a bit futile. So we only measure "False" Negatives(FN) ie. the objects that our model has missed out.

- The average precision (AP) is a way to summarize the precision-recall curve into a single value representing the average of all precisions. The AP is calculated according to the next equation. Using a loop that goes through all precisions/recalls, the difference between the current and next recalls is calculated and then multiplied by the current precision. In other words, the AP is the weighted sum of precisions at each threshold where the weight is the increase in recall.

$$AP = \sum_{n=0}^{k=n-1} [Recalls(k) - Recalls(k+1)] * Precisions(k) \tag{2.23}$$

$$Recalls(n) = 0, Precisions(n) = 1, n = number\ of\ thresholds. \tag{2.24}$$

- The mean Average Precision or mAP score is calculated by taking the mean AP over all classes and/or overall IoU thresholds, depending on different detection challenges that exist.

$$AP = \frac{1}{n} \sum_{n=1}^{k=n} AP_k \tag{2.25}$$

$$AP_k = the\ AP\ of\ class\ k, n = the\ number\ of\ classes \tag{2.26}$$

Chapter 3

MULTIPLE OBJECT TRACKING

3.1 Overview

3.2 MOT Components Overview

3.3 Detection Based Tracking and End to End Tracking

3.4 Low Level Feature Method

3.5 Deep SORT

3.5.1 Feature Extraction and Embedding

3.5.2 Affinity Computation and Data Association

3.6 Metrics and Evaluations

3.6.1 Conventional Metrics

3.6.2 Update Metrics

Chapter 4

INFERENCE

4.1 Architecture Optimization

4.2 Algorithm Optimization

4.3 TensorRT Framework

4.4 Post-Processing

4.4.1 Speed Estimation

4.4.2 Anomaly Detection

Chapter 5

MODEL TRAINING

5.1 Dataset

5.2 Loss Function

5.3 Hyperparameters

Chapter 6

SOFTWARE DESIGN

6.1 System Overview

6.2 Hardware Design

6.3 Software Design

6.3.1 Dependencies

6.3.2 User Interface

6.3.3 Administration