University of Science

Vietnam National University,

Ho Chi Minh City

Seminar Report

Student

21127676 – Phan Huu Phuoc

21127592 – Nguyen Minh Dat

21127191 – Nguyen Nhat Truyen

21127675 – Nguyen Hoang Phuoc

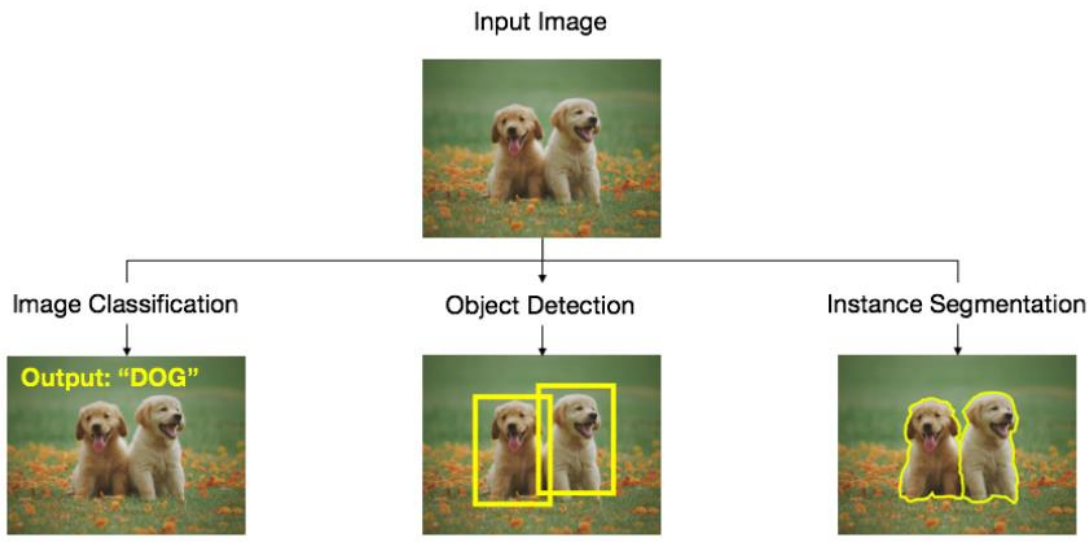
Class: 21KHMT
Theoretical Instructor: Dr. Le Hoang Thai
Practical Instructor: Dr. Truong Tan Khoa, Dr. Duong Thai Bao

Table of contents

I. Introduction

Common Computer Vision tasks:

- **Image Classification**: Image classification is a computer vision task that involves assigning a label or category to an input image. The goal is to train a model that can accurately classify images into predefined classes or categories.
- **Object Detection**: Object detection is a computer vision task that involves identifying and localizing objects within an image. Unlike image classification, object detection not only determines the class of the object but also detects its location by drawing bounding boxes around the objects.
- **Image Segmentation**: Image segmentation is a computer vision task that involves dividing an image into multiple coherent regions or segments. Each segment represents a distinct object or region within the image. The goal is to assign a label to each pixel, indicating which segment or object it belongs to.



Face Detection problem:

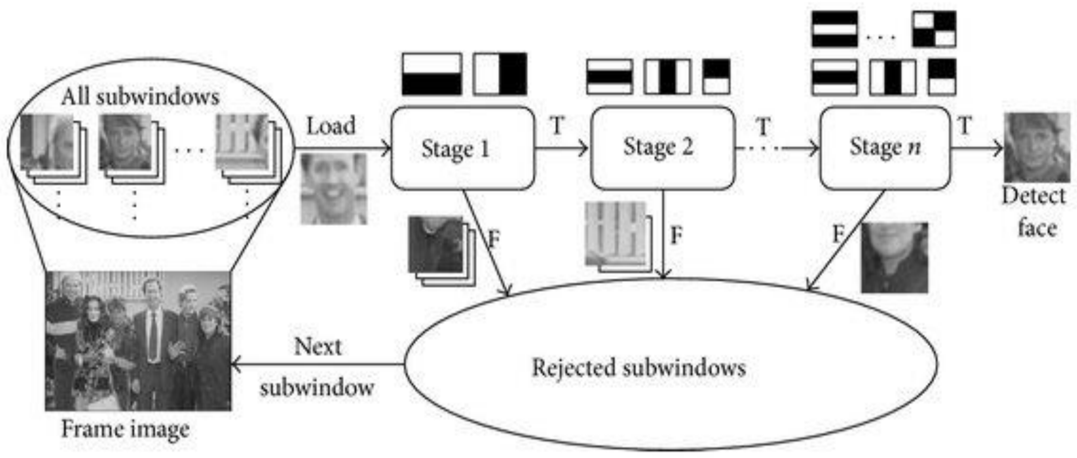
- A significant advancement in computer vision is the development of **automatic face detection**. This technology allows computers to analyze digital images and videos, automatically identifying and pinpointing the location of human faces within them.
- Face detection acts as a crucial building block for a variety of **applications**. It forms the foundation for systems that can recognize individuals based on their facial features, track the movement of people in videos, and even analyze facial expressions to gauge emotions.

II. Related work

Object detection in computer vision has come a long way. While Haar cascades, introduced in 2001 by Viola and Jones, were a significant breakthrough, recent advancements in deep learning have led to even more powerful methods.

A. Haar cascades

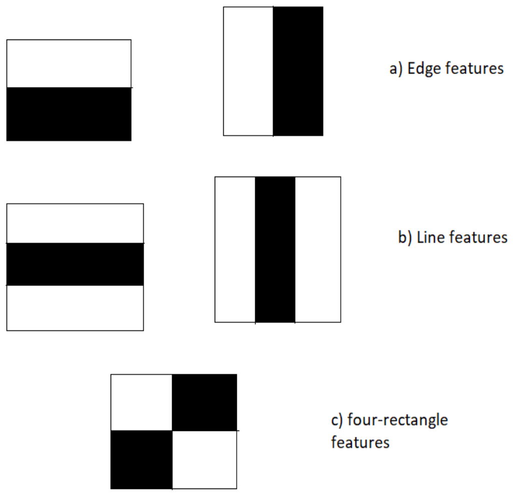
Haar cascades rely on a set of predefined features called **Haar features** to identify objects in images. These features are simple squares or rectangles that capture basic characteristics like edges and intensity variations. By training a classifier on these features, Haar cascades can effectively detect specific objects, such as faces, with impressive speed. This makes them well-suited for **real-time** applications where processing speed is crucial.



Picture 1: Step by step facial recognition of Haar Cascade

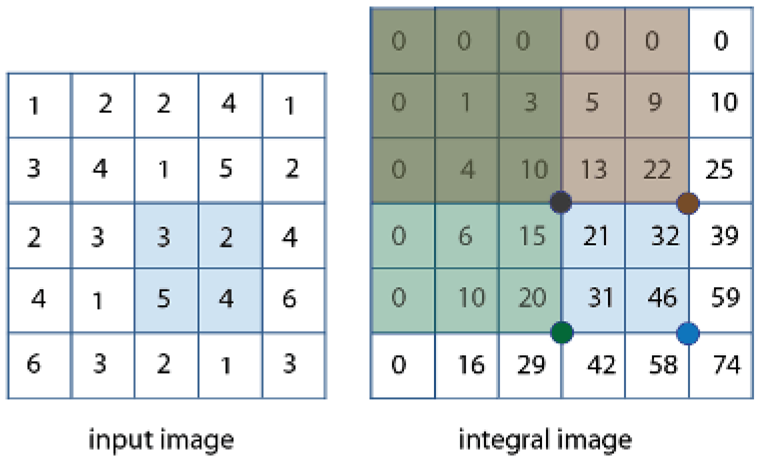
One Haar-like feature can be viewed as a **mask** consisting of three values:

- **1** for those pixels in the **white** region of the feature.
- **−1** for those **dark** region pixels in the feature.
- **0** for those pixels **outside** of the feature region.



Haar-like features can be **computed efficiently** using the **integral image** technique.

- The integral image at location x, y contains the sum of the pixels **above and to the left** of x, y.
- Using the integral image, any rectangular sum can be computed in **four array references**, leading to enormous savings in computation for features at varying locations and scales.
- Example: the summation of the shaded region from the left image can be computed as: 46 – 22 – 20 + 10 = 14.



1	2	2	4	1
3	4	1	5	2
2	3	3	2	4
4	1	5	4	6
6	3	2	1	3

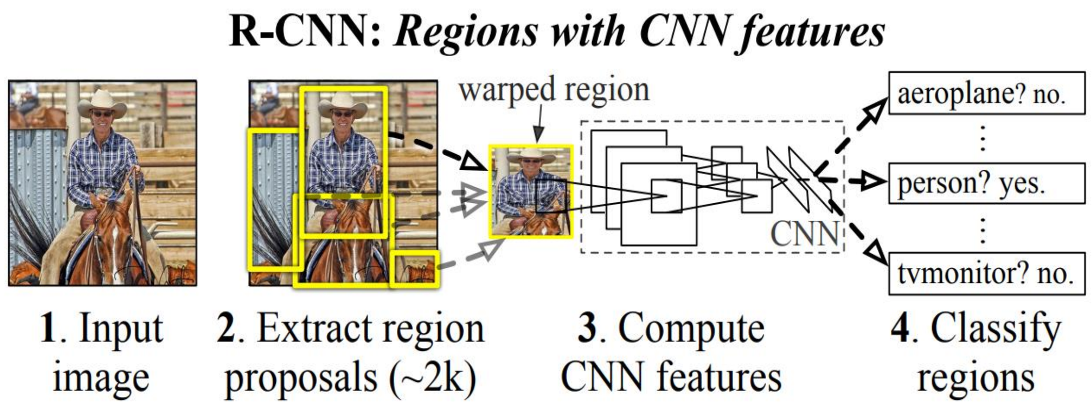
0	0	0	0	0	0
0	1	3	5	9	10
0	4	10	13	22	25
0	6	15	21	32	39
0	10	20	31	46	59
0	16	29	42	58	74

B. R-CNN Family

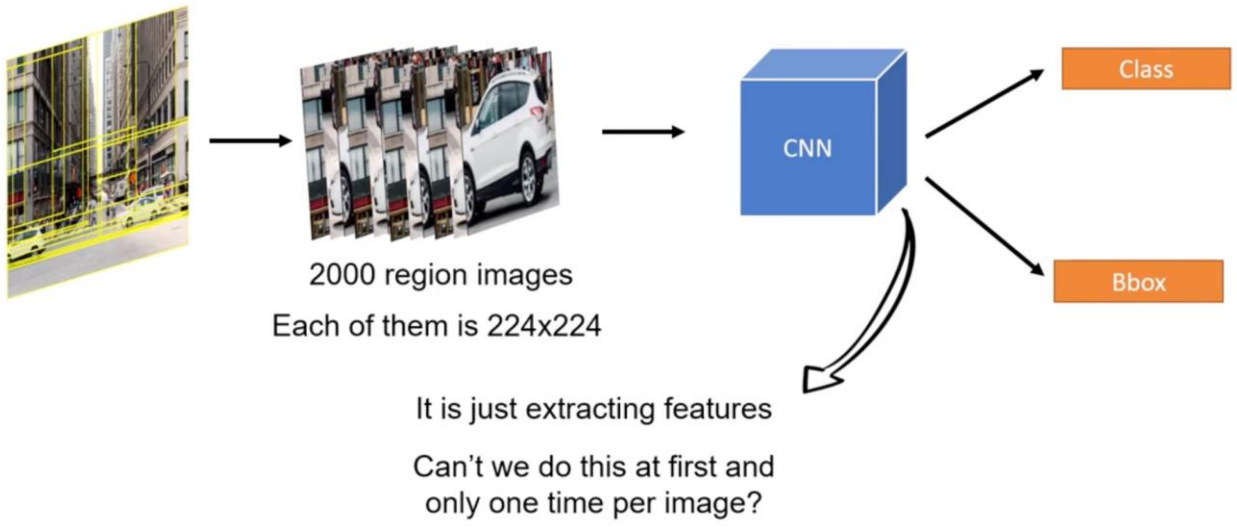
Deep learning has significantly impacted nearly every aspect of machine learning-based computer vision. For example, image recognition, image search engines (also known as content-based image retrieval or CBIR), Simultaneous Localization and Mapping (SLAM), and image segmentation, etc. has changed since the explosion of artificial neural networks and deep learning. Object detection is no exception.

One of the most popular deep learning-based object detection algorithms is the R-CNN family of algorithms, proposed by Girshick et al. (2013) introduced for the first time. Since then, the R-CNN algorithm has gone through many iterations, improving the algorithm with new versions and outperforming traditional object detection algorithms (e.g. Haar cascades (Viola and Jones, 2001); HOG + Linear SVM (Dalal and Triggs, 2005)) in all stages of development.

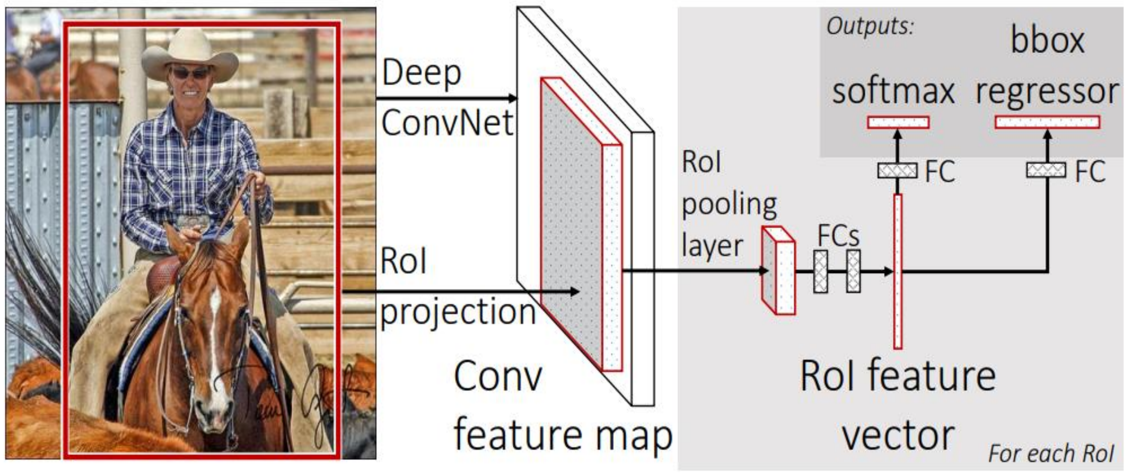
R-CNN (2-stage approach): This initial version relied on a two-stage process. First, an external algorithm called "**selective search**" identified potential object regions based on color and other image features. Then, these regions were fed into a deep learning network for **classification** and **bounding box refinement**. While effective, R-CNN was **slow** due to the separate proposal generation step.

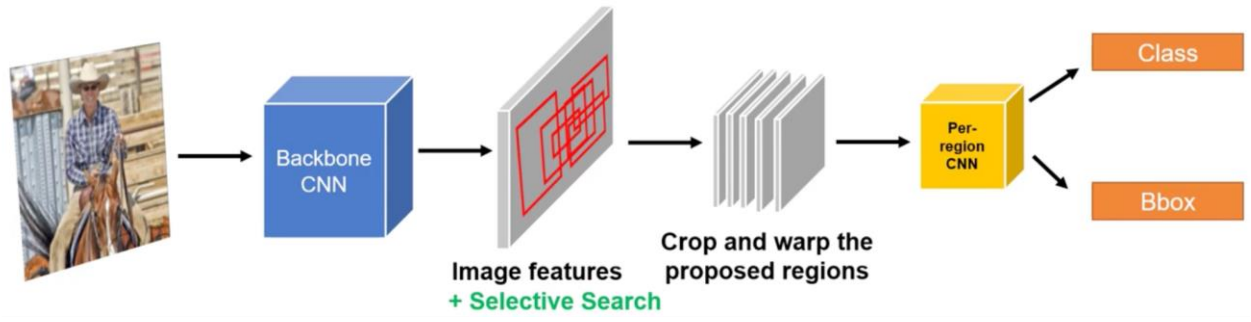


- **Speed Bottleneck**: training an R-CNN model is expensive and slow, as the following steps involve a lot of work:
    - selective search to propose 2000 region candidates for every image.
    - Generating the CNN feature vector for every image region (N_images * 2000).
    - The whole process involves three models separately without much shared computation: the convolutional neural network for image classification and feature extraction; the top SVM classifier for identifying target objects; and the regression model for tightening region bounding boxes.


2000 region images
Each of them is 224x224
CNN
Class
Bbox
It is just extracting features
Can't we do this at first and only one time per image?

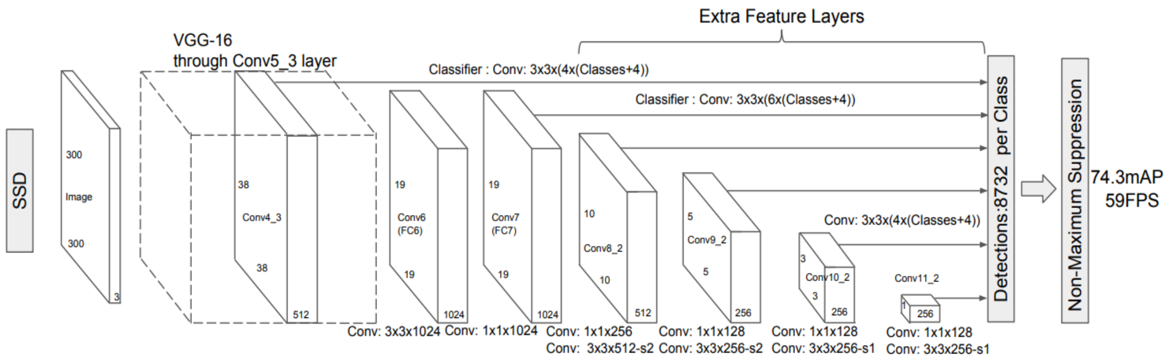
Fast R-CNN (faster 1st stage): Recognizing the bottleneck, Fast R-CNN aimed for speed improvement. It still used selective search but ran the deep learning network on the entire image first. Then, only the proposed regions were extracted and further processed for classification and bounding box refinement. This approach significantly improved processing speed compared to R-CNN.





Faster R-CNN (unified network): The key innovation of Faster R-CNN lies in its **unified** architecture. It eliminates the need for a separate proposal generation step like selective search. Instead, Faster R-CNN integrates a dedicated network called the **Region Proposal Network (RPN)** directly into the deep learning architecture. The RPN operates on the same feature map as the object classification and bounding box tasks, making the entire process faster and more efficient. This unified approach, along with the use of deep networks for proposal generation, is what sets Faster R-CNN apart from its predecessors and contributes to its superior accuracy.

C. The Single Shot Detector (SSD)



SSD is a real-time object detection model that achieves its speed through a unique approach. Unlike other models, it utilizes a single neural network on feature maps extracted from a convolutional neural network. This eliminates the need for a separate region proposal network step, significantly boosting processing speed. SSD further optimizes detection by employing multiple bounding boxes of various sizes, improving accuracy for objects of different dimensions. This one-pass design with a single neural network makes SSD considerably faster than other models while maintaining acceptable detection accuracy.

D. State-of-the-art Method

Proposed paper:

- Jian Li, Bin Zhang, Yabiao Wang, Ying Tai, Zhenyu Zhang, Chengjie Wang, Jilin Li, Xiaoming Huang, and Yili Xia. 2021. **ASFD: Automatic and Scalable Face Detector**. In Proceedings of the 29th ACM International Conference on Multimedia (MM '21). Association for Computing Machinery, New York, NY, USA, 2139–2147. https://doi.org/10.1145/3474085.3475372

Accuracy:

- 0.972 on WIDER Face(Easy) Dataset.
- 0.965 on WIDER(Medium) Dataset.
- 0.925 on WIDER(Hard) Dataset.

Key Components:

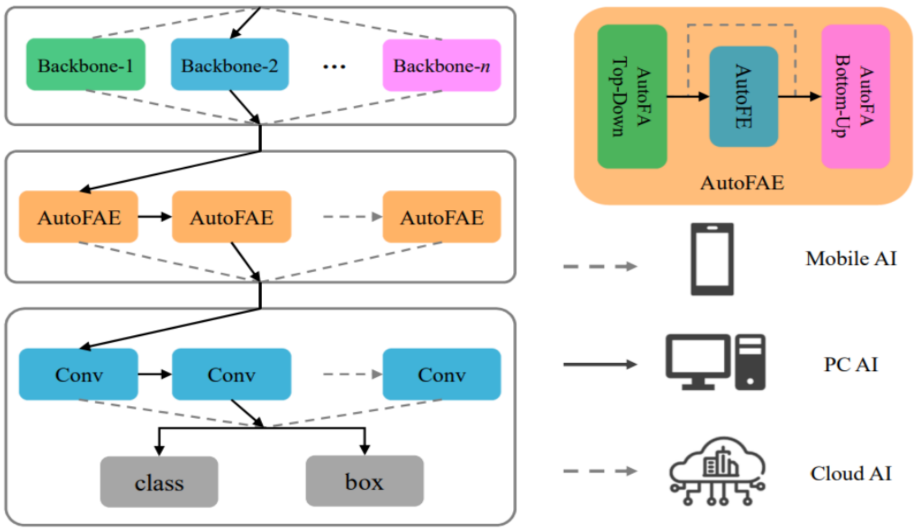
- **CNN Backbone for Feature Extraction**: The method employs a Convolutional Neural Network (CNN) backbone to extract feature maps from the input images. CNNs are well-suited for this task due to their ability to learn and identify spatial patterns in images, making them effective for feature extraction in various image processing tasks.
- **Neural Architecture Search (NAS) for Optimal Network Design**: The method utilizes NAS to automatically discover an optimal network architecture for the specific task of face detection. NAS algorithms can explore a vast space of possible network architectures and evaluate their performance based on a defined metric, such as face detection accuracy. This approach eliminates the need for manual network design and allows the method to potentially find a more effective network structure for the task at hand.
- **AutoFAE for Feature Aggregation and Enhancement**: The method incorporates an AutoFAE (Automatic Feature Aggregation and Enhancement) module to enhance the extracted features before feeding them into the main neural network. AutoFAE likely combines both Feature Aggregation (FA) and Feature Enhancement (FE) functionalities.
    - **FA** refers to the process of combining features extracted from different parts of an image. In face detection, FA can be crucial for combining

information from different facial regions like eyes, nose, and mouth to create a more robust representation of the face.

- **FE** focuses on improving the quality and discriminative power of the extracted features. This is essential for face detection, especially in challenging conditions like variations in lighting, pose, or occlusion.

Overall Flow:

1. The CNN backbone extracts feature maps from the input image.
2. The AutoFAE module performs FA and FE on the extracted feature maps, enhancing their quality and discriminative power.
3. The enhanced feature maps are fed into the main neural network for face detection.



Pros:

- By leveraging Neural Architecture Search (NAS) to find an optimal network architecture, ASFD could potentially achieve higher accuracy in face detection compared to existing methods.
- The name "Scalable" suggests the detector might be adaptable to different face sizes or resolutions.

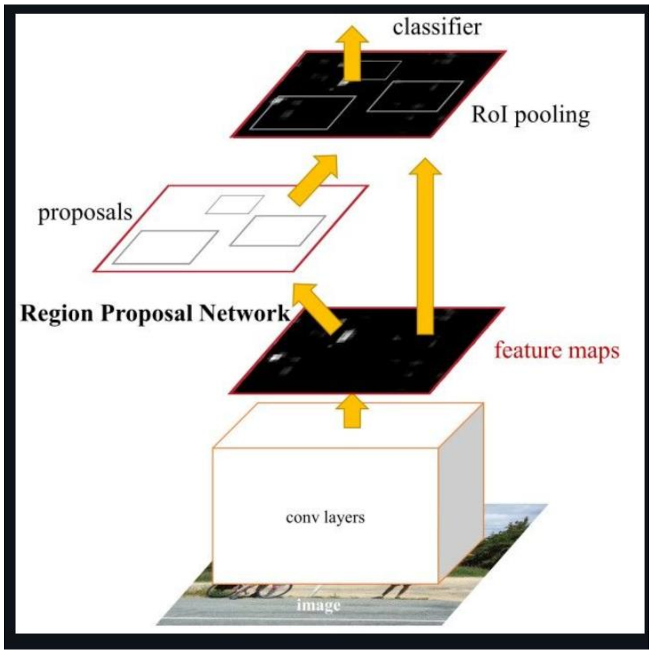
Cons:

- NAS algorithms can be computationally expensive as they explore a vast search space of network architectures.
- The effectiveness of the discovered architecture would depend on the quality and size of the training data used during NAS. A large and diverse dataset is crucial for training a robust face detector.

- Optimizing the network architecture for peak accuracy could lead to a computationally expensive model that might not be suitable for real-time applications.

III. Proposed Method

A. Architecture



1. Feature Extraction in Faster R-CNN

The feature extraction stage in Faster R-CNN uses a pre-trained convolution network (CNN) to convert raw image data into a high-dimensional representation. This representation, called a **feature map**, captures important information about the image content, such as edges, shapes, and textures.

- **Pre-trained CNN**: Popular choices include ResNet or VGG. These models are trained on large datasets like ImageNet and learn synthetic image features applicable to various object detection tasks.
- **Convolutional Layers**: These layers perform core feature extraction. They apply image scanning filters and triggers based on specific patterns.
- **Feature Map**: Output of Convolutional Layers. This is a tensor (multidimensional array), in which each element represents a specific feature at a specific location in the image.

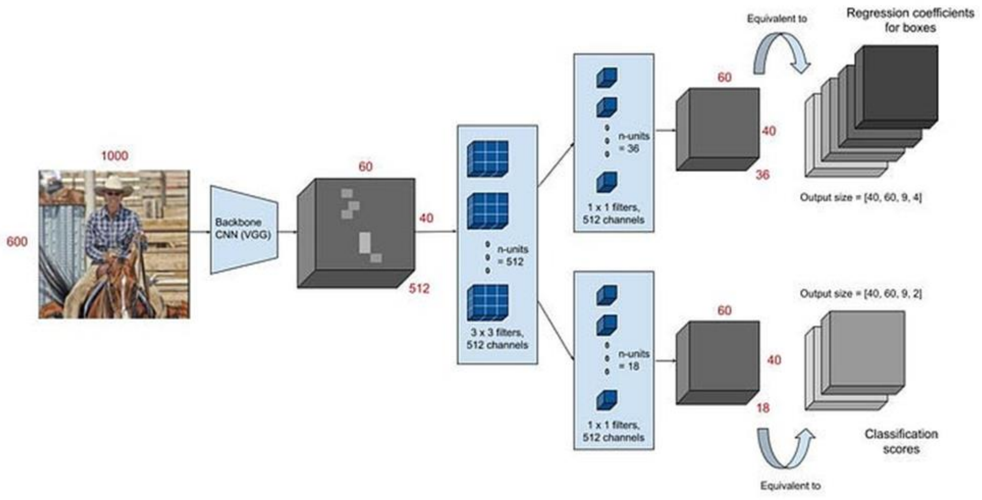
2. Region Proposal Network (RPN)

Instead of using selective search like Fast RCNN to generate proposal regions, RPN uses a small neural network to extract these proposal regions. Before understanding the structure of RPN and how it extracts proposal regions, let's understand the concept of anchor boxes.

Anchor Boxes:

- Every ‘pixel’ of the feature image is considered an **anchor**. For each pixel in the input of RPN, we will check **k** different cases of the anchor box placed at this point.
    - In every image, we are only interested in two main criteria: **scale** and **aspect ratio**.
    - Usually, we fix 3 scale cases and 3 aspect ratio cases for each anchor box. So k usually be 3*3=9.
- For each anchor, we will consider it similar to a sliding window in Fast RCNN, but instead of searching on every combination of sliding window at each pixel of the image (which makes Fast RCNN "extremely slow") like Fast RCNN, we will use 2 separate neural networks to identify whether this anchor meets the standard of a “proposal region”.

RPN structure:



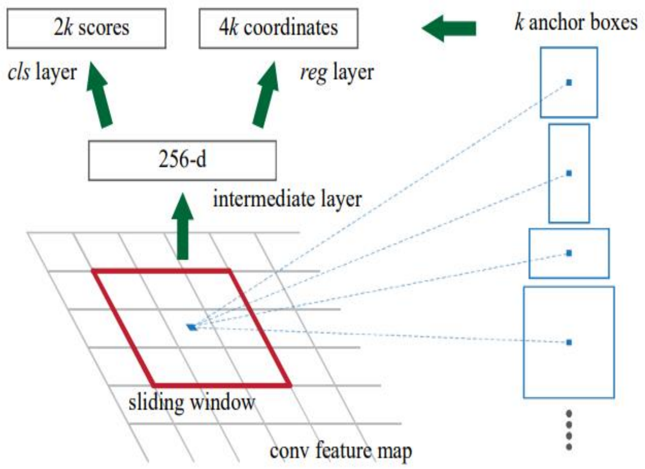
1000
600
Backbone CNN (VGG)
60
40
512
n-units = 512
3 x 3 filters, 512 channels
n-units = 36
1 x 1 filters, 512 channels
60
40
36
Equivalent to
Regression coefficients for boxes
Output size = [40, 60, 9, 4]
n-units = 18
1 x 1 filters, 512 channels
60
40
18
Output size = [40, 60, 9, 2]
Classification scores
Equivalent to

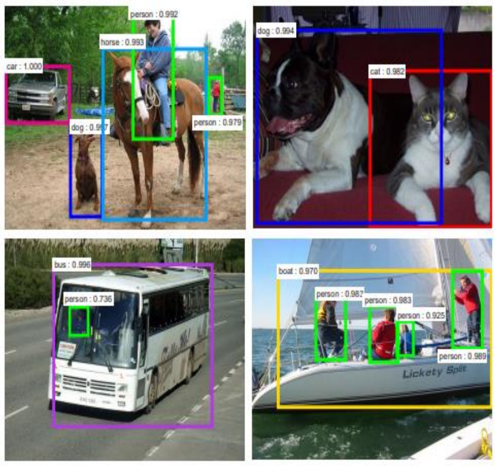
We will use a **Regression layer** to regress the "**position**" of the anchor box and combine it with a **Classification layer** to predict the "**probability**" of containing an object (in this case, a face) in the anchor box.

- Specifically, the regression layer will have an output of a **4-dimensional** vector representing the **coordinate** values of the anchor box.
- Usually, the model will label positive anchor boxes with a classification greater than 0.7 and negative anchor boxes with a classification less than 0.3, and the rest will be dropped.

In summary, for each pixel, we will:

- **Analyzing each k anchor box that has this pixel as an anchor.**
- **Perform regression with the 4 coordinates of the anchor box to find the minimum size to detect the object**
- **Then finally perform classification with these anchor boxes.**





Loss Function:

Since this is an algorithm that requires training, we need to define the loss function.

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{box}}$$
$$\mathcal{L}(\{p_i\}, \{t_i\}) = \frac{1}{N_{\text{cls}}} \sum_i \mathcal{L}_{\text{cls}}(p_i, p_i^*) + \frac{\lambda}{N_{\text{box}}} \sum_i p_i^* \cdot L_1^{\text{smooth}}(t_i - t_i^*)$$

- p_i is the **predicted** probability of anchor i being an object.
- p_i^* is the **ground truth** label whether anchor i is the object.
- t_i is a 4-dimensional vector representing the **predicted** bounding box coordinate values.
- t_i^* is a 4-dimensional vector representing the **ground-truth** box coordinate values corresponding to the **positive** anchor.

The first term is the classification loss over 2 classes (There is object or not).

Lcls is the log loss of the 2 classes (object and non-object)

$$\mathcal{L}_{\text{cls}}(p_i, p_i^*) = -p_i^* \log p_i - (1 - p_i^*) \log(1 - p_i)$$

The second term is the regression loss of bounding boxes only when there is object (i.e. $p_i^* = 1$).

Lbox uses SmoothL1Loss

$$L_1^{\text{smooth}} = \begin{cases} |x| & \text{if } |x| > \alpha; \\ \frac{1}{|\alpha|}x^2 & \text{if } |x| \leq \alpha \end{cases}$$

(The default value of α is 1)

- The **Ncls** is a **normalization term** set to the mini-batch size which is **256.**
- The **Nbox** is also a **normalization term** set to **the number of anchor boxes** (~2500).
- The **λ** is set to **10**, which is a **balancing parameter** such that **Lcls** and **Lbox** are weighted equally.
- The RPN is trained via **end-to-end backpropagation** and standard **Stochastic Gradient Descent** with a learning rate of **0.001.**

3. ROI Pooling

Definition of this part:

- ROI Pooling is a crucial component in Faster R-CNN (Regions with CNN features) object detection architectures. It addresses the challenge of processing Regions of Interest (ROIs) with **varying sizes** within a fixed-size framework for classification and bounding box regression.
- RoI pooling layer, a Spatial pyramid Pooling (SPP) technique, is the main idea behind Fast R-CNN and the reason that it outperforms R-CNN in accuracy and speed respectively

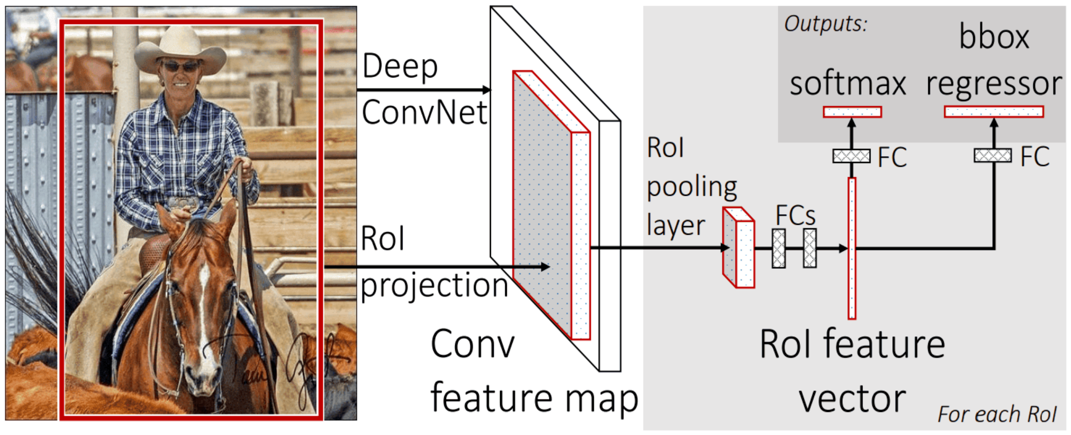
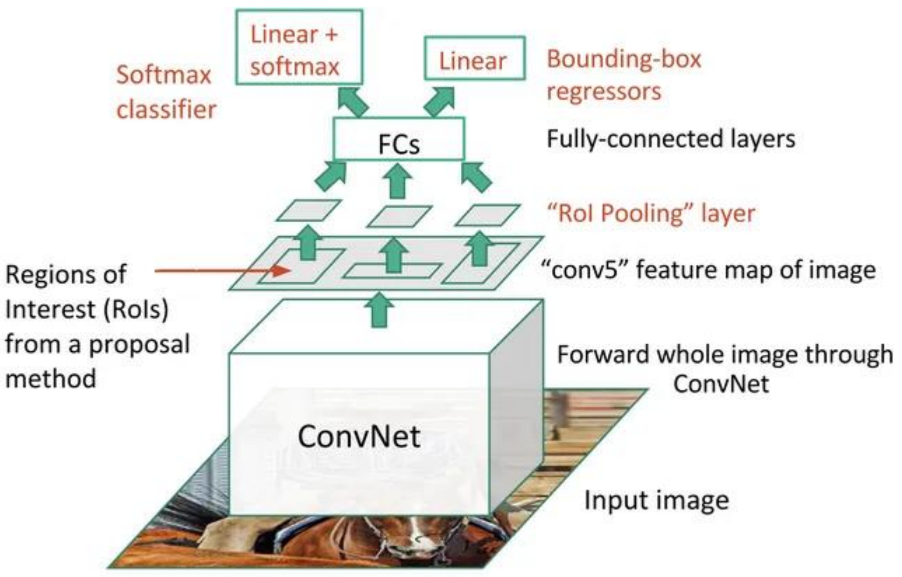


Image source: Faster R-CNNs - PyImageSearch

Core Function:

- Takes a convolutional feature map generated from the entire input image by a pre-trained convolutional neural network (CNN).
- Extracts fixed-size feature vectors from specific ROIs proposed by a region proposal network (RPN).

How it work:



Source: [Region of Interest Pooling. A Technique which allowed a new… | by Sambasivarao. K | Towards Data Science (towards data science-com.translate.google)]()

- ROI Selection: The RPN generates a set of rectangular ROIs that potentially contain objects.

- ROI Pooling Layer:
  - Divides each ROI into a grid of smaller sub-regions (e.g., 7x7).
  - Performs max pooling (or another pooling operation) within each sub-region. This captures the most significant feature within each sub-region.
  - Outputs a fixed-size feature map for each ROI
- ROI Pooling in Action (Simplified Example):
  - **Input image**: 400x400 pixels.
  - **Convolutional feature map**: 100x100 (after processing by the CNN)
  - **ROI 1**: 50x80 pixels (proposed object region).
  - **ROI Pooling** (7x7 grid): Divides ROI 1 into 49 sub-regions (7x7). Performs max pooling within each sub-region, resulting in a fixed-size 7x7 feature map capturing the most prominent features from ROI 1.
  - Similar processing for other ROIs proposed by the RPN.

B. Experiments

Toolbox: Detectron2

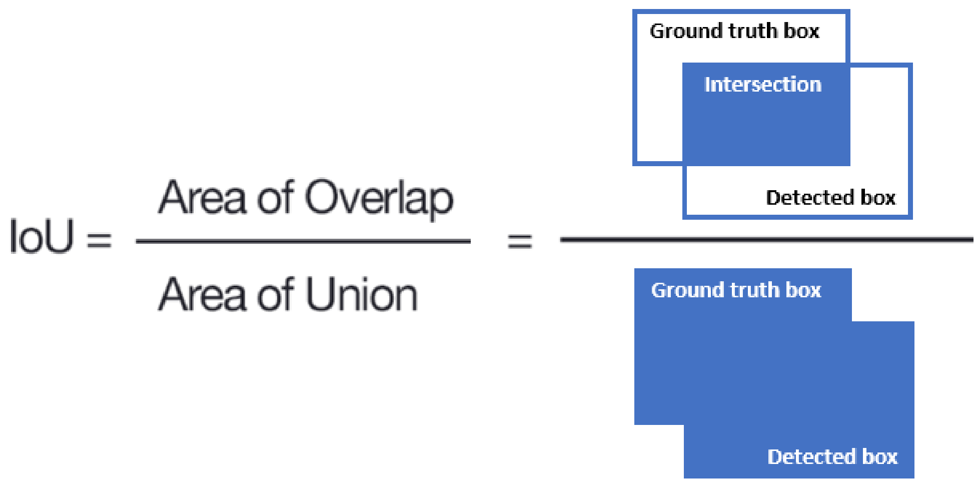
- Detectron2 is a **platform** for object detection, segmentation and other visual recognition tasks. Detectron2 offers a wide range of **pre-trained models** for various computer vision tasks, including instance segmentation, panoptic segmentation, and object detection. This extensive **model zoo** provides users with ready-to-use models that can be fine-tuned or used directly for inference, saving time and effort in model development.
- Detectron2 provides a wide range of models in its model zoo, each tailored for specific computer vision tasks. It includes implementations for the following **object detection** algorithms:
    - Faster R-CNN.
    - RetinaNet.
    - RPN & Fast R-CNN.

Dataset: Human Faces (Object Detection)

- **Source**: Kaggle.
- **Scope**: 2204 unique images.
- **Description**: A diverse compilation of human facial images encompassing various races, age groups, and profiles, with the aim of creating an unbiased dataset that includes coordinates of facial regions suitable for training object detection models.
- **Structure**:
    - Image folder.
    - Annotation file (.csv).
- **Attributes**:
    - **image_name**: Image file name.
    - **width**: Image width.
    - **height**: Image height.
    - **x0**: x-coordinate of the top-left point.
    - **y0**: y-coordinate of the top-left point.
    - **x1**: x-coordinate of the bottom-right point.
    - **y1**: y-coordinate of the bottom-right point.

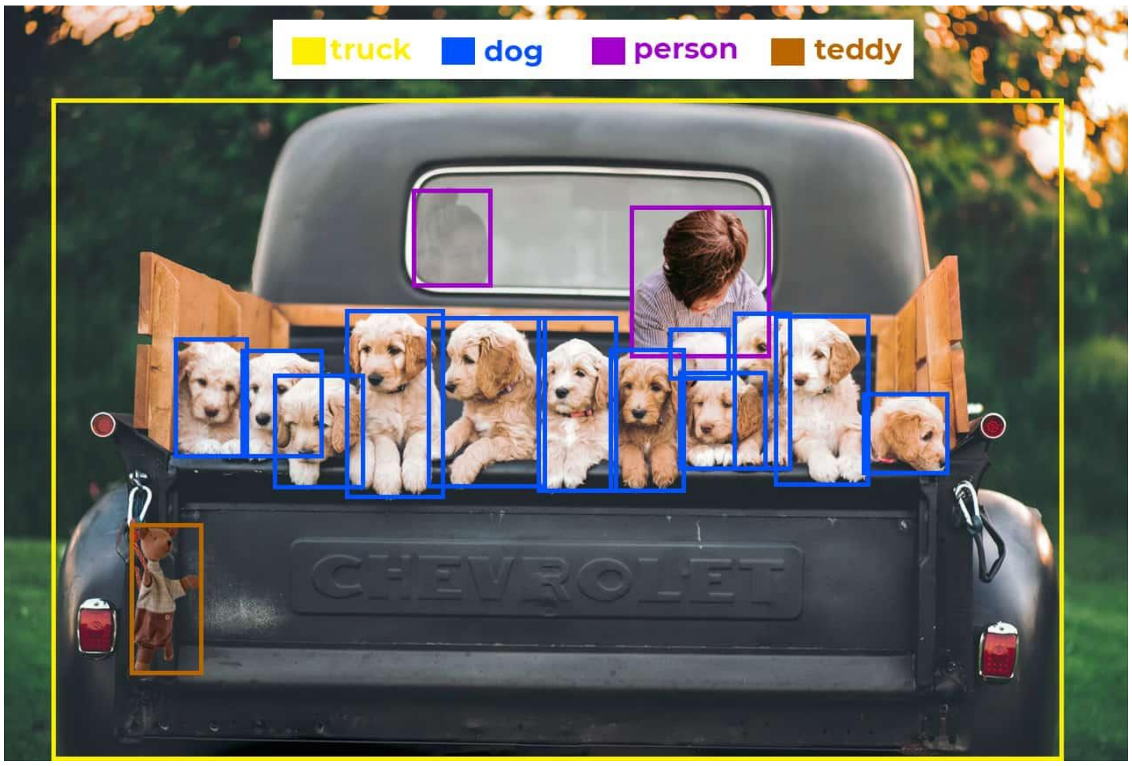
Metric: COCO Evaluation Metrics

- **Intersection over Union**
    - IoU measures the **overlap** between the prediction and the annotation by taking the ratio of the intersection area to the union area.

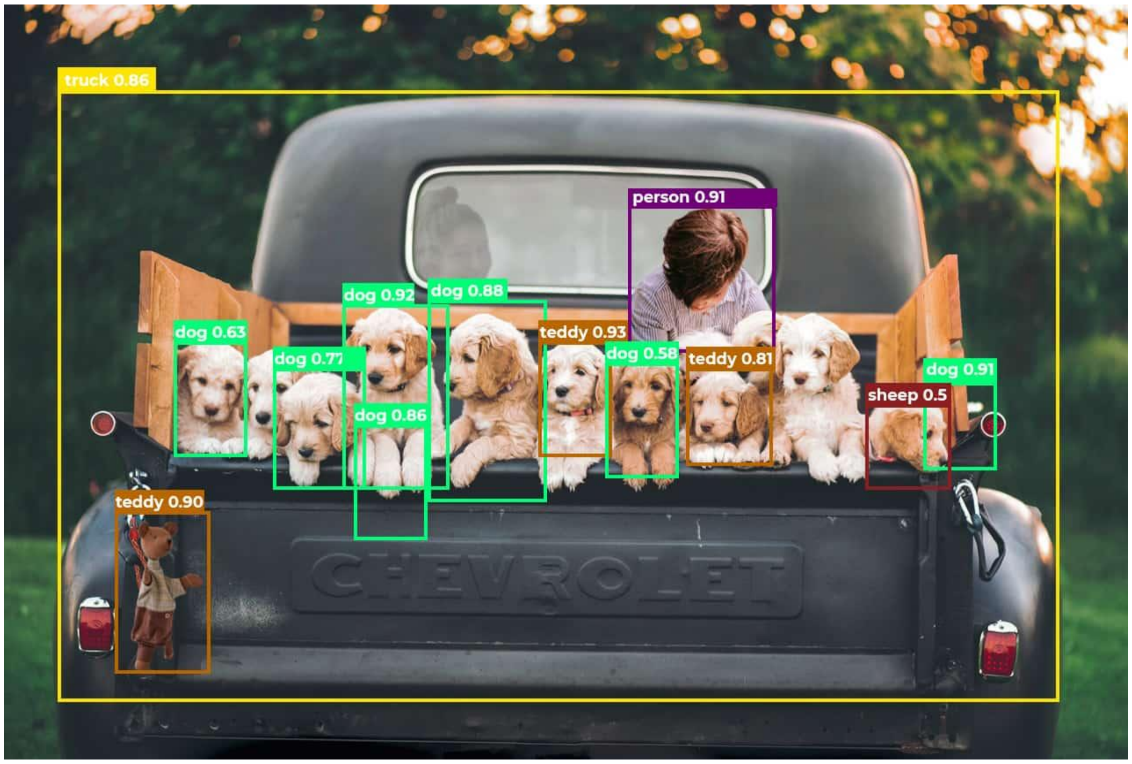


- ○ **IoU Threshold Selection**: On COCO, the average precision is evaluated over several IoU values: the threshold of 0.50, the range 0.50-0.95, and the threshold of 0.75.

- **AP (Average precision)**
    - Average Precision (AP) is not the average of Precision (P). It is the area under the precision-recall curve.
        - A precision-recall curve is simply a graph with Precision values on the y-axis and Recall values on the x-axis.
    - Example:
        - Ground truth: 2 people, 12 dogs, 1 teddy, 1 truck.



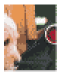
- Predictions:



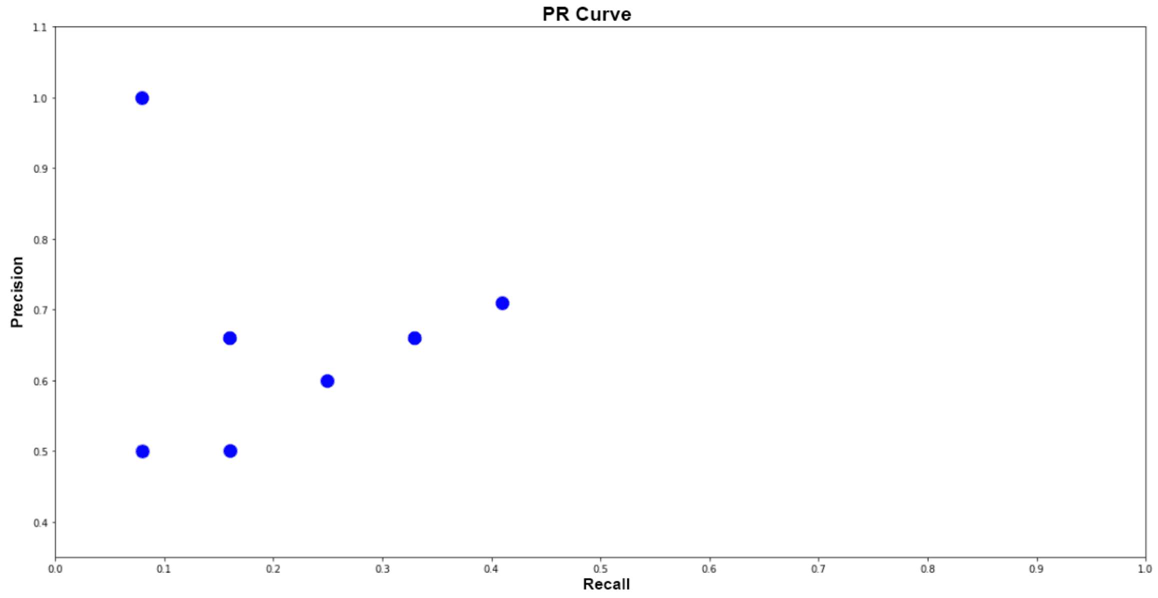
- Record every Dog detection along with the Confidence score

Detections							
Conf.	0.63	0.77	0.92	0.86	0.88	0.58	0.91
Matches GT by IoU?	TP	TP	TP	FP	TP	TP	FP

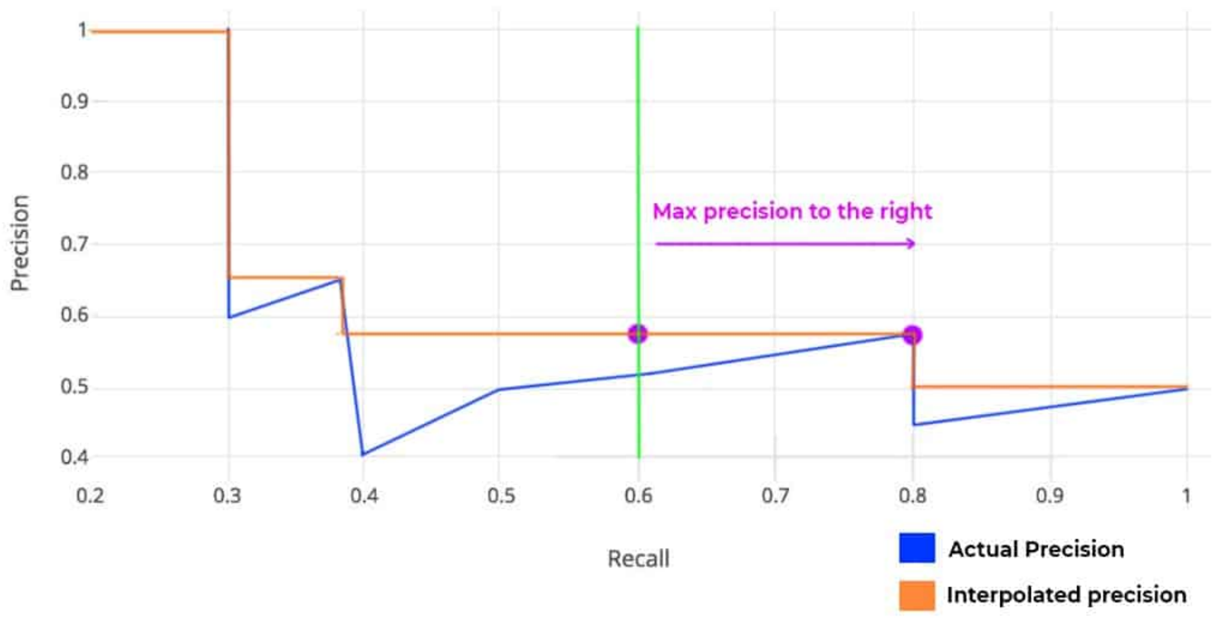
- Calculate Precision and Recall

Preds.	Conf.	Matches	Cumulative TP	Cumulative FP	Precision	Recall
	0.92	TP	1	0	1/(1+0) = 1	1/16 = 0.08
	0.91	FP	1	1	1/(1+1) = 0.5	1/16 = 0.08
	0.88	TP	2	1	2/(2+3) = 0.66	2/16 = 0.16
	0.86	FP	2	2	0.5	0.16
	0.77	TP	3	2	0.6	0.25
	0.63	TP	4	2	0.66	0.33
	0.58	TP	5	2	0.71	0.41

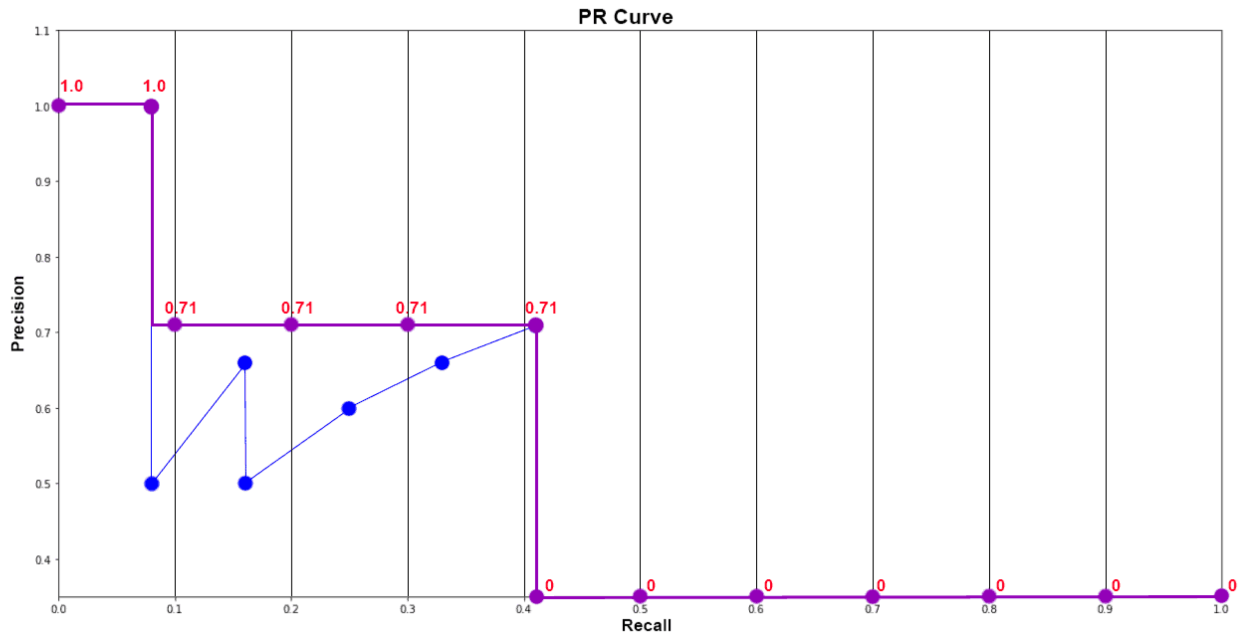
- Plot Precision-Recall graph



- Calculate Average Precision (AP) using PASCAL VOC 11 Point Interpolation Method



- Plot Final Interpolated graph and calculate Average Precision for Dog Class



- $\mathbf{AP}_{\mathbf{dog}}$ = 1/11 * (1 + 4*0.71 + 6*0) = 0.349

- **Mean Average Precision (mAP)**
  - Mean Average Precision or mAP is the average of AP over all detected classes.

Training parameters:

- Base learning rate: 0.001
- Number of unique images: 400
- Number of images per training batch: 4
- Number of iterations for the warm-up phase: 600
- Number of iterations for training: 900
- Number of region of interest (ROI) proposals per image: 64
- Number of classes: 2
- Steps at which the learning rate is reduced: (600,)
- Factor by which the learning rate is reduced at each step (gamma): 0.05

Hardware Specification:

- Runtime type: Google Colab T4 GPU
- System RAM: 12.7 GB
- GPU RAM: 15.0 GB
- Disk: 78.2 GB

Results:

	Faster R-CNN	**Mask R-CNN**
AP	31.173	34.541
AP50	84.154	84.404
AP75	9.602	28.779
APm	28.531	18.057
APl	32.942	46.336

Comparisons:

- Faster R-CNN has lower results compared to Mask R-CNN. However, Faster R-CNN still exhibits strengths in certain areas:
  - The AP50 score of 84.15% indicates that the Faster R-CNN model achieves a high precision rate when considering detections with an IoU threshold of 0.5 or higher as correct.
  - The APm and APl scores of 28.53% and 32.94% respectively indicate that the model performs reasonably well in detecting and localizing medium and large objects.

IV. Conclusion

Faster R-CNN is a two-stage object detection model that employs a pre-trained CNN for feature extraction and a Region Proposal Network (RPN) to efficiently identify candidate object regions. ROI pooling then extracts fixed-size features from these regions for classification and bounding box refinement.

While achieving a lower overall average precision compared to Mask R-CNN on a human faces dataset, Faster R-CNN demonstrates strengths in high precision for detections with a 0.5 IoU threshold and reasonable performance in detecting medium and large objects. This balance between accuracy and speed makes Faster R-CNN a valuable tool for various object detection tasks.

V. References

[1] Integral Image

[2] Mean Average Precision (mAP) in Object Detection

[3] A deeper look at how Faster-RCNN works

[4] ASFD: Automatic and Scalable Face Detector

[5] Fast R-CNN: Everything you need to know from the paper

[6] R-CNN: Clearly EXPLAINED!

[7] Faster R-CNN: Faster than Fast R-CNN!

[8] Deep Residual Learning for Image Recognition

[9] Visualizing and Comparing AlexNet and VGG using Deconvolutional Layers