

Deep Learning AI 2100



Object detection with voice feedback





Overview





Overview

- Implemented YOLOv3.
- Used pretrained weights as training on coco takes days to provide good results.
- Provided a small extension to it using gTTs.

*YOLOv3 was implemented taking help from [here](#).



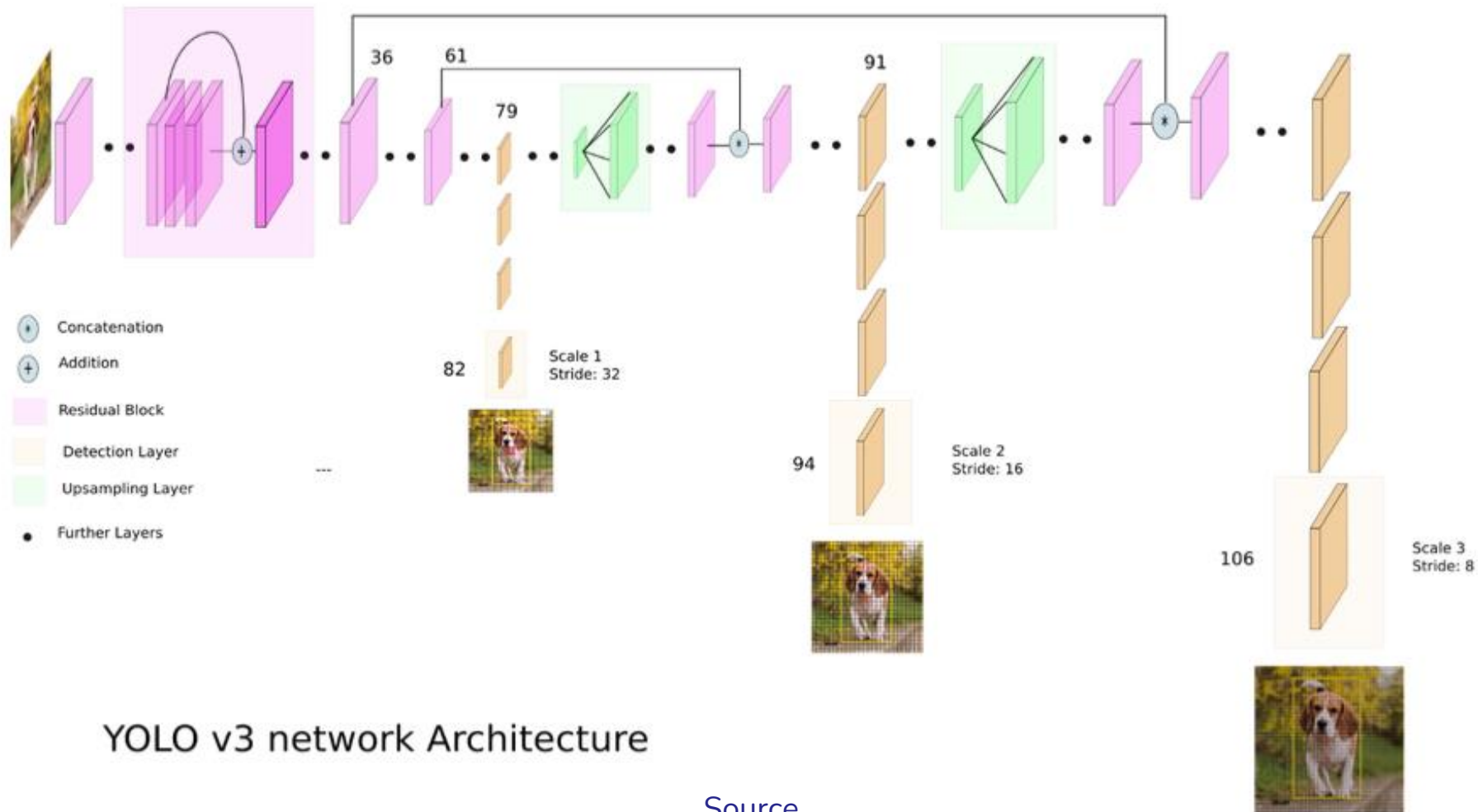
How does the YOLO Framework Function?


- Input image is divided into grids.
- Image classification and localization are applied on each grid.
- Output for each bounding box is a 5+C dimensional vector.
- The location and class of the object are predicted.
- The following are used to enhance the algorithm's performance even further
 - **Intersection over Union and Non-Max Suppression**
 - **Anchor boxes**




Architecture







Components of the architecture

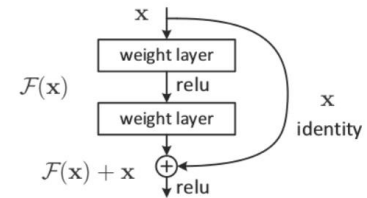


Component-1: Feature Extractor



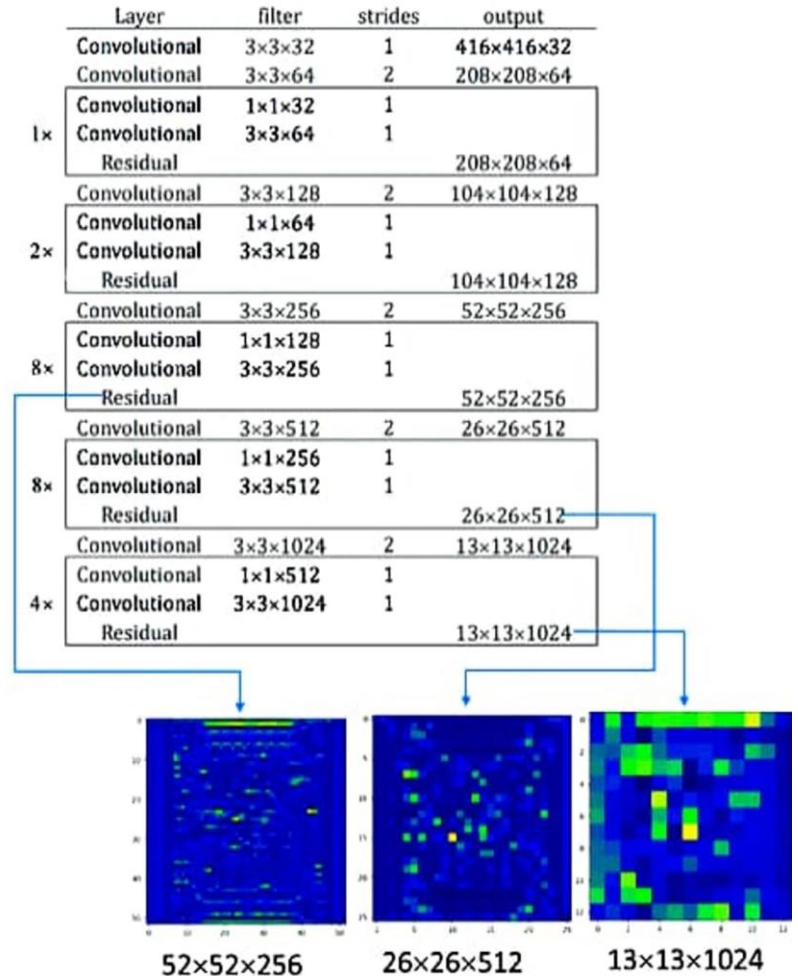
- YoloV3 uses an extractor that is the hybrid of YoloV2 layers and Darknet 53.
- The network uses 53 convolution layers.
- The network is built with **consecutive 3x3 and 1x1 convolution layers** followed by a **skip connection**.
- 53 layers of the darknet are further stacked with 53 more layers for the detection head, making YOLO v3 a total of a **106 layer fully convolutional underlying architecture**.
- For an example image of size 416x416, our three scales of detection will be 52x52, 26x26 and 13x13.

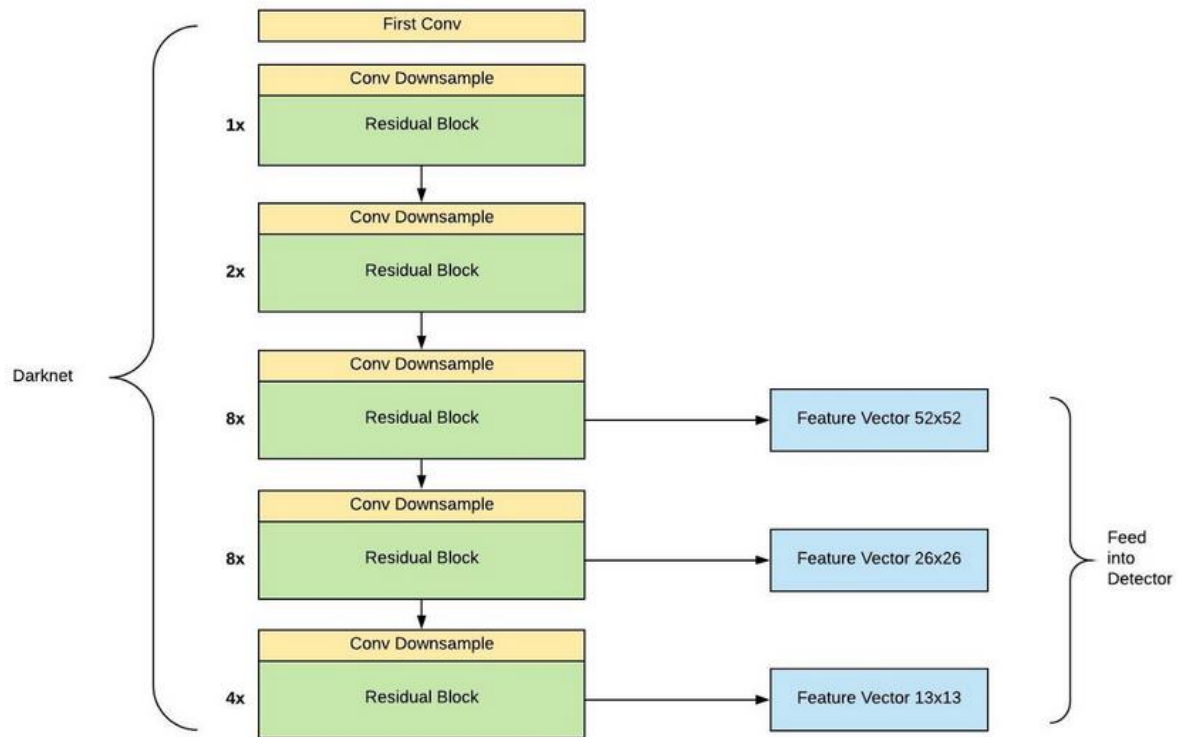
	Layer	filter	strides	output
	Convolutional	3×3×32	1	416×416×32
	Convolutional	3×3×64	2	208×208×64
1×	Convolutional	1×1×32	1	
	Convolutional	3×3×64	1	
	Residual			208×208×64
2×	Convolutional	3×3×128	2	104×104×128
	Convolutional	1×1×64	1	
	Convolutional	3×3×128	1	
	Residual			104×104×128
8×	Convolutional	3×3×256	2	52×52×256
	Convolutional	1×1×128	1	
	Convolutional	3×3×256	1	
	Residual			52×52×256
	Convolutional	3×3×512	2	26×26×512
	Convolutional	1×1×256	1	
	Convolutional	3×3×512	1	
	Residual			26×26×512
4×	Convolutional	3×3×1024	2	13×13×1024
	Convolutional	1×1×512	1	
	Convolutional	3×3×1024	1	
	Residual			13×13×1024



Skip connections

- The darknet framework loads 106 layers.
- The object detections are made at layer 82 , 94 , 106.
- Yolo v3 makes detection at 3 different scales . Feature vector of shape (13 ,13) for large objects , feature vector of shape (26 ,26) for medium objects and feature vector of shape (52 ,52) for small objects is extracted from darknet.







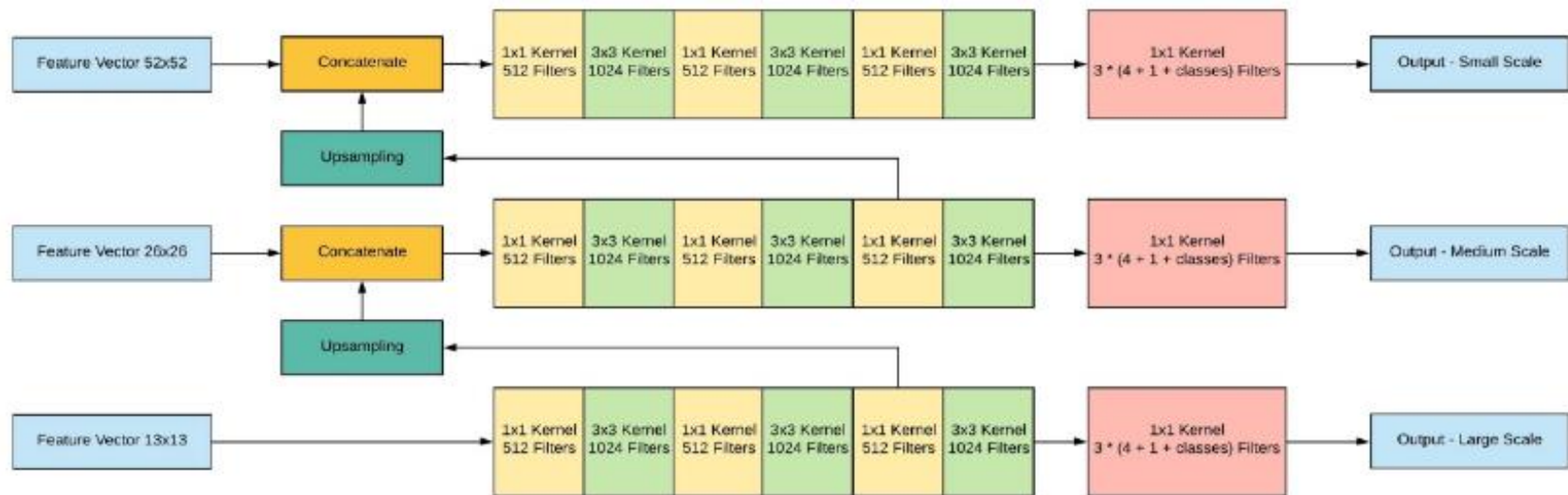
- In the prediction part, the network uses convolution layer, upsample layer, skip connection and convolution layer with stride 2 as downsample layer.
- We don't use any kind of pooling layer to downsample the input image as the pooling layer losses more low-level features from the input image.
- This network predicts bounding box on downsampled images for fast real-time prediction.
- We downsample image by a factor, called stride. So stride of the network is equal to the factor by which the output layer is smaller than the input image. YOLOv3 uses three different strides (8, 16, 32).
- So if the size of the input image is (416 , 416) , then the output tensors would be of size (52 , 52), (26 , 26), (13 , 13).

Component-2: Multi-Scale Detector



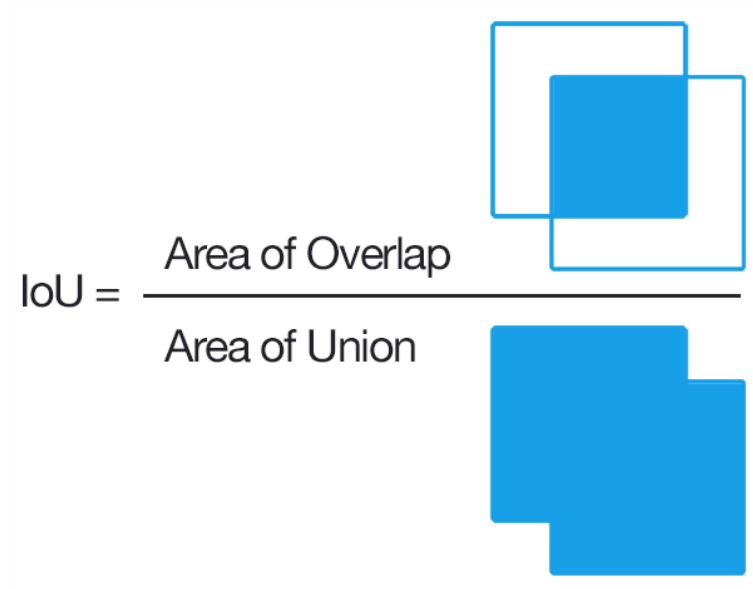


- An important feature of the YOLO v3 model is its multi-scale detector
- The detection for an eventual output of a fully convolutional network is done by applying 1x1 detection kernels on feature maps of three different sizes at three different places.
- The shape of the kernel is $1 \times 1 \times (B * (5 + C))$.
- For the COCO dataset, $C=80$ (number of classes.)
- B stands for the number of bounding boxes which is 3 in this case.
- '5' refers to the five bounding box attributes (eg: center coordinates(bx, by), height(bh), width(bw), and confidence score).
- YOLOv3 predicts offsets to pre-defined default bounding boxes, called anchor boxes. YOLOv3 uses different anchors on different scales. YOLOv3 model predicts bounding boxes on three scales and in every scale, three anchors are assigned. So in total, this network has **nine** anchor boxes. These anchors are taken by running K-means clustering on dataset.



$\text{IoU} = \text{Area of the intersection} / \text{Area of the union}$

IoU is used to understand whether our prediction is good or it isn't. We take a certain threshold say 0.5. If IoU is greater than 0.5, we can say that the prediction is good enough.





Non Max Suppression

NMS

A problem that can occur here is multiple bounding boxes detecting the same object.

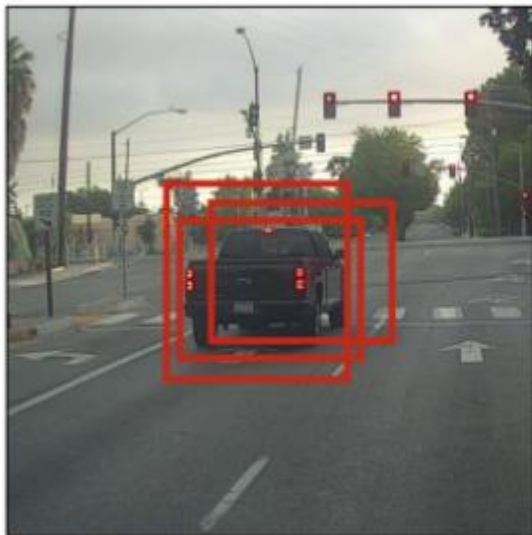
The Non-Max Suppression technique cleans up this up so that we get only a single detection per object.

We look at the probabilities associated with each detection and take the largest one.

The boxes which have high IoU with the current box are suppressed.

After the boxes have been suppressed, it selects the next box from all the boxes with the highest probability. WE repeat this until all the boxes have been considered.

Before non-max suppression



**Non-Max
Suppression**



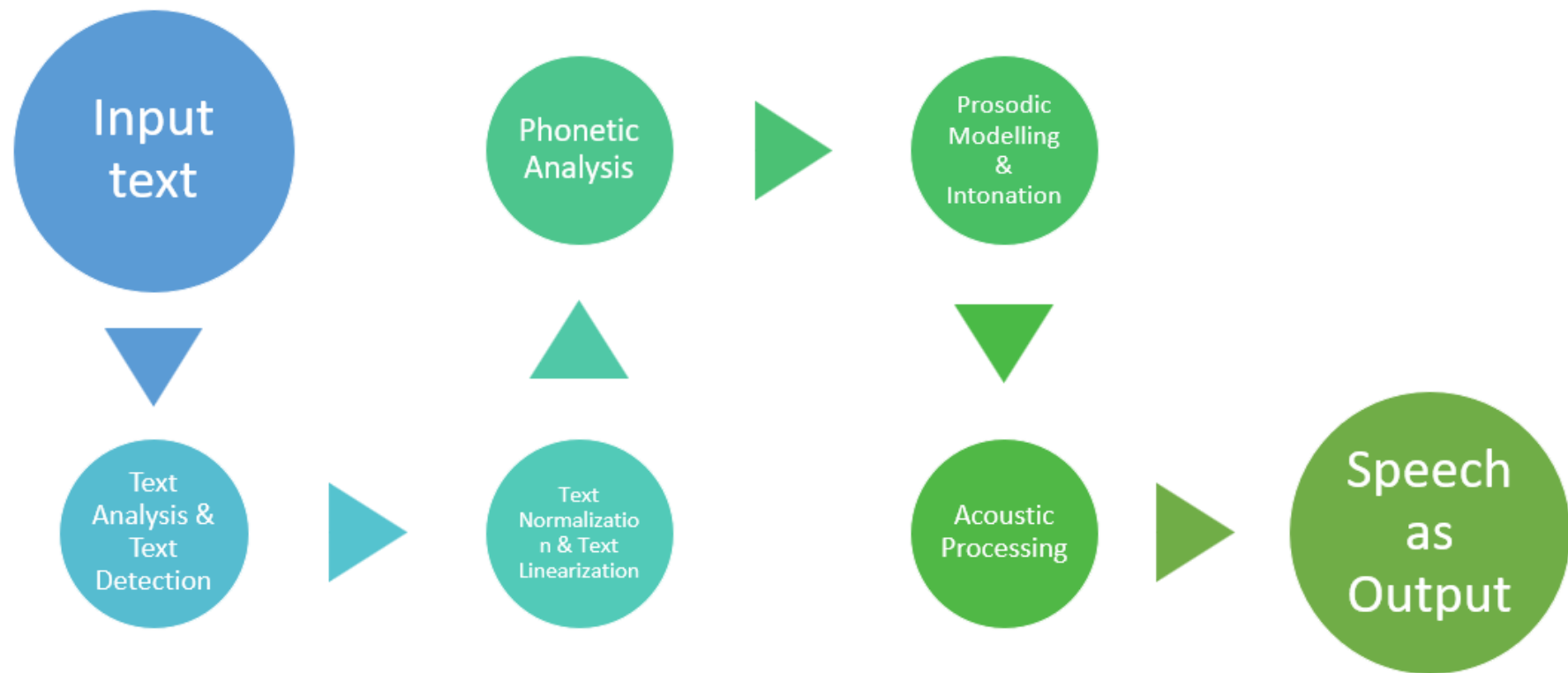
After non-max suppression





Text to Speech Conversion





gTTS