Deep learning – computer vision

Object Detection: SSD

Learning Objectives

At the end of this module, you will understand:

- Working of SSD
- Architecture of SSD
- Implementation of SSD using tensorflow object detection API

Agenda

01 SSD

1. Introduction to SSD
2. Architecture of SSD

02 WORKING OF SSD

1. SSD 300 Architecture
2. SSD 300: Detector & Classifier

03 SSD ARCHITECTURE

1. Default box generation
2. Boundary correction, classification and filtering

04 TF OD API

1. SSD implementation using TF OD API

05 CONCLUSION

1. Quiz
2. Summary

01

SSD

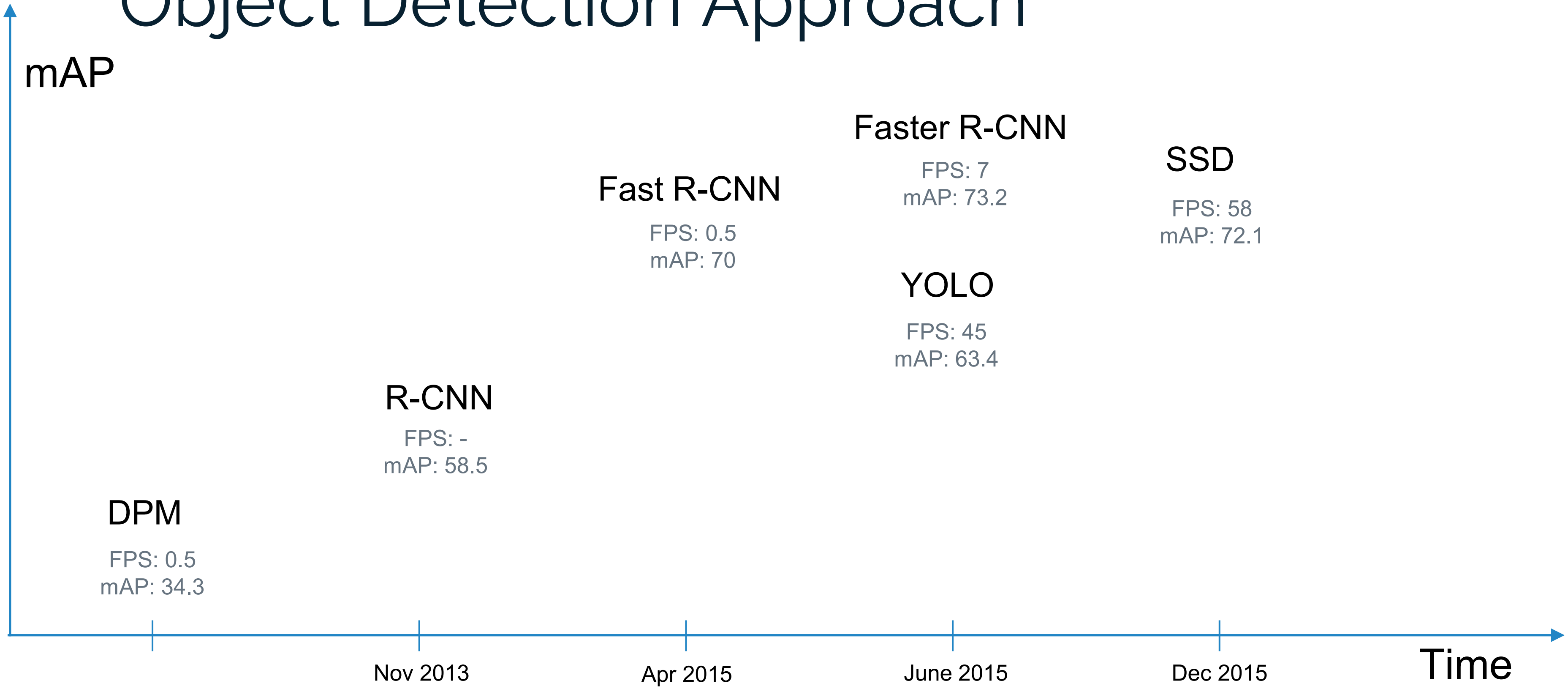
- Introduction to SSD
- Architecture of SSD

SSD

Single Shot MultiBox Detector

Single Shot Multibox Detector is a method for detecting objects in images using a single deep neural network.

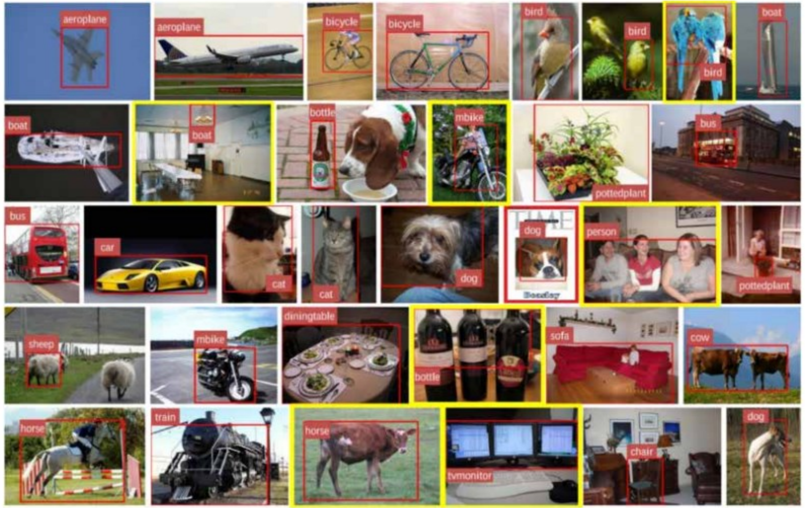
Object Detection Approach



Results on PASCAL VOC 2007 sample dataset

Ssd - single shot multibox detector

- SSD was released on November 2016.
- It reached new records in terms of performance a over 74% mAP (mean Average Precision) at 59 fr as PascalVOC and COCO.



Why ssd?

- SSD are designed for object detection in real-time.
- SSD speeds up the process by eliminating the need of the region proposal network.
- To recover the drop in accuracy, SSD applies a few improvements including multi-scale features and default boxes.

Architecture of ssd

- Single Shot: Tasks of object localization and classification are done in a single forward pass of the network
- MultiBox: name of a technique for bounding box regression
- Detector: The network is an object detector that also classifies those detected objects

02

WORKING OF SSD

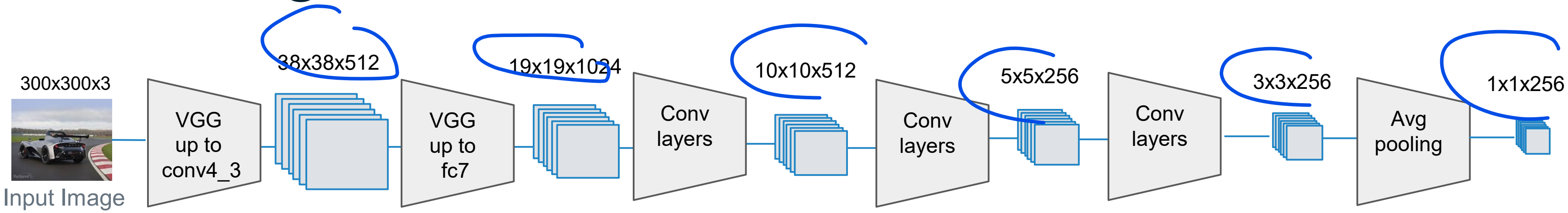
- SSD 300 Architecture
- SSD 300: Detector & Classifier

SSD 300 Architecture

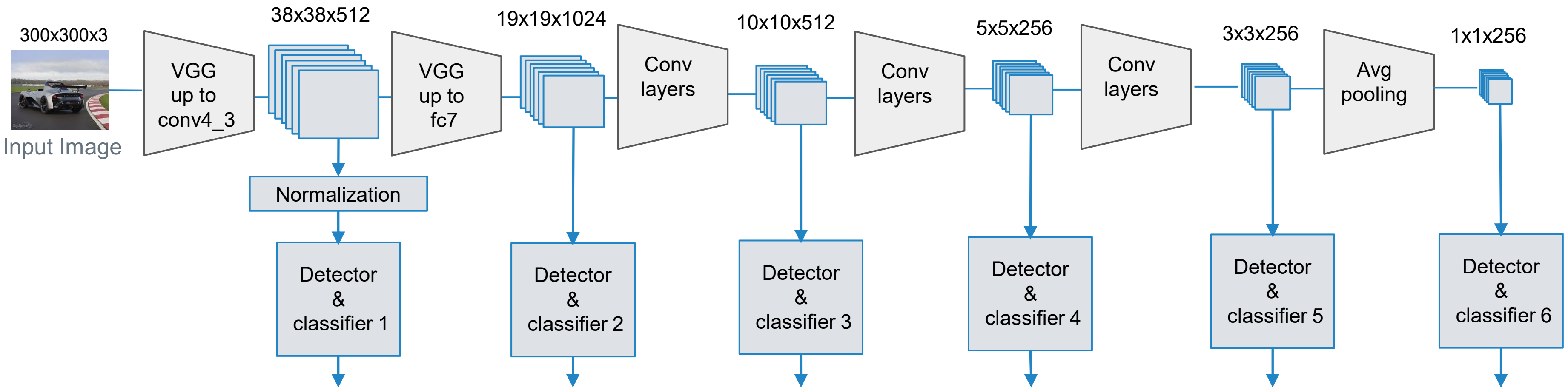
300x300x3

Input Image

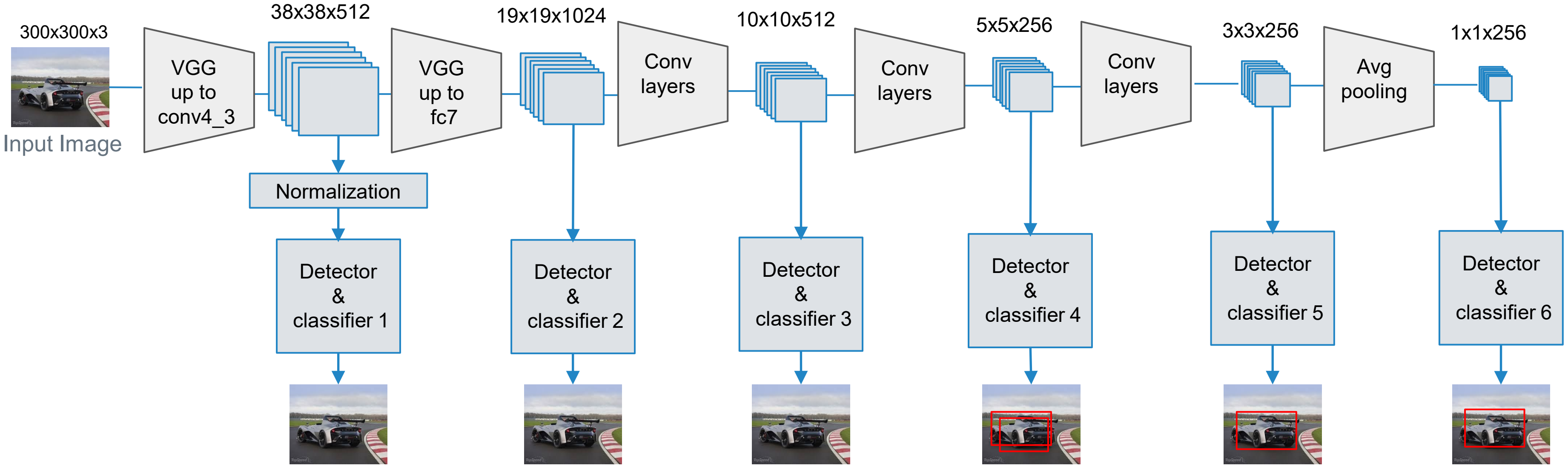
SSD 300 Architecture



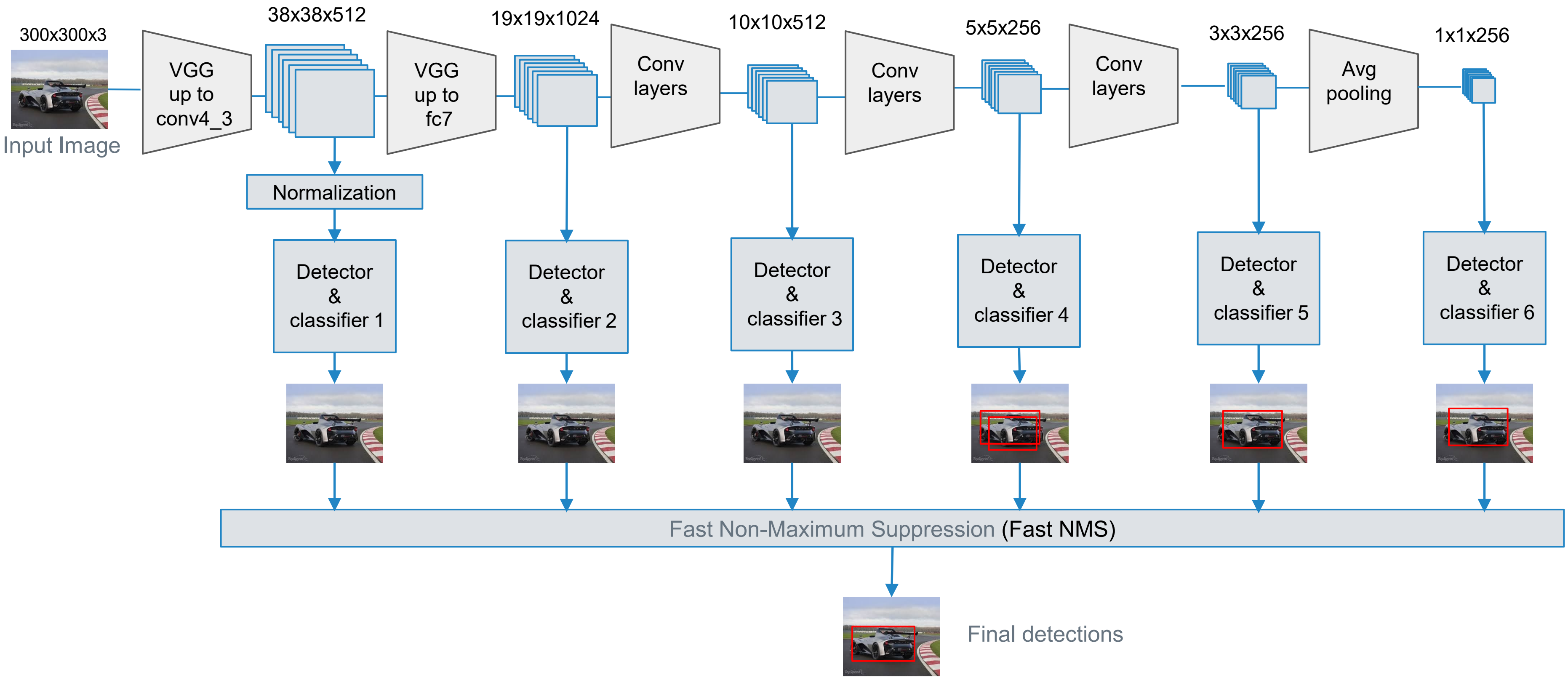
SSD 300 Architecture



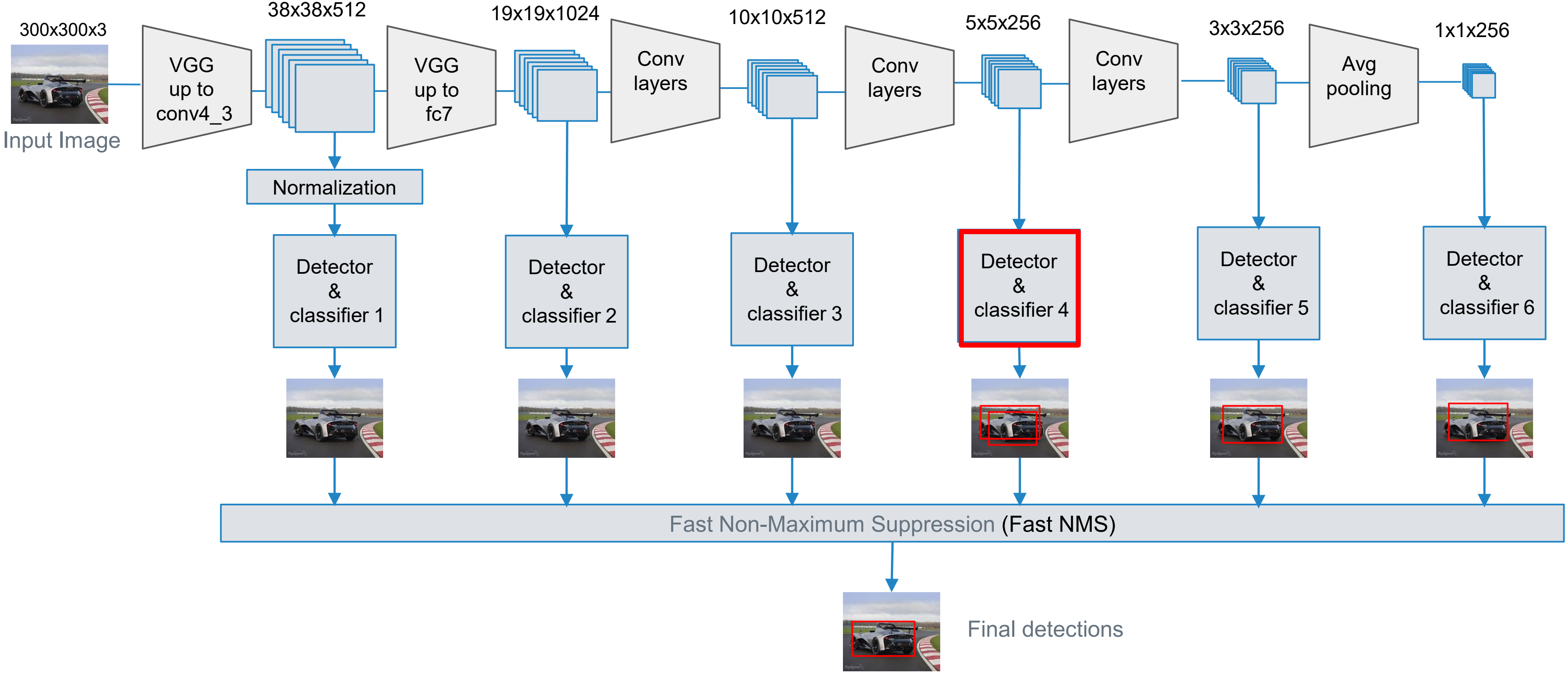
SSD 300 Architecture



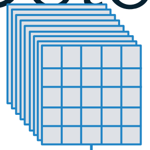
SSD 300 Architecture



SSD 300 Architecture



SSD 300: Detector & classifier

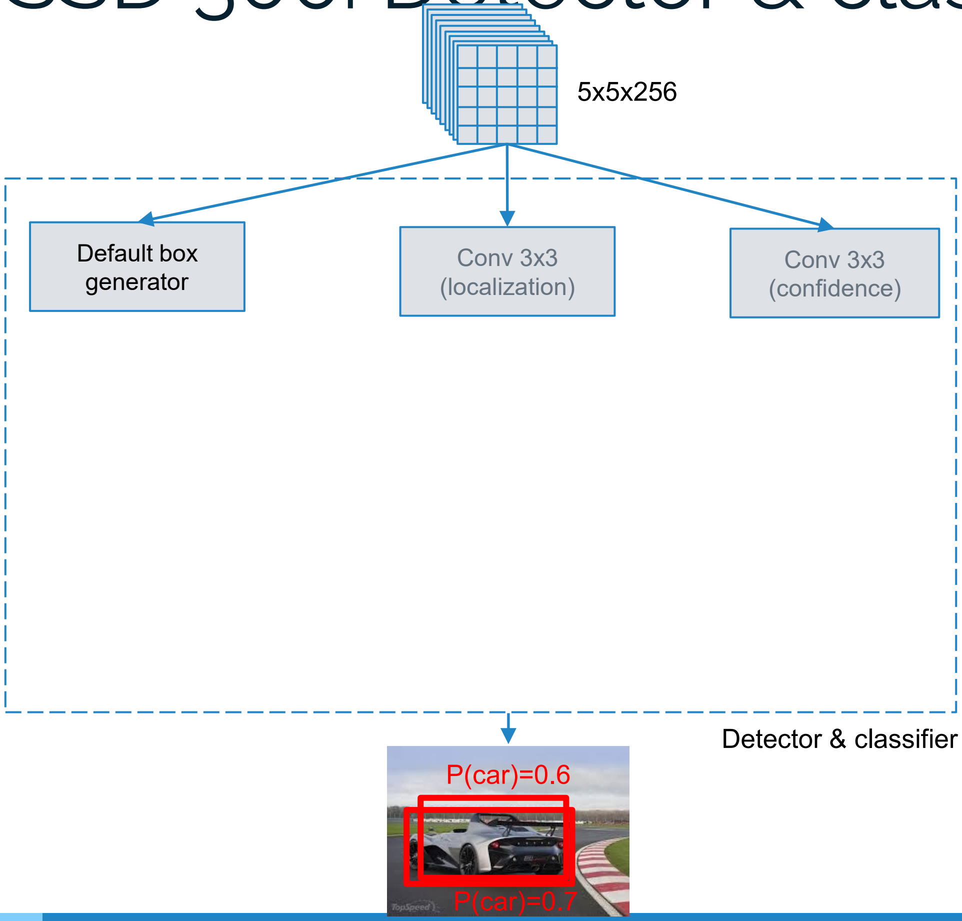


Considering the following parameters:

- Size of original image (300 x 300)
- Dimension of feature maps (5 x 5 x 256)
- #default boxes = 3

Detector & classifier

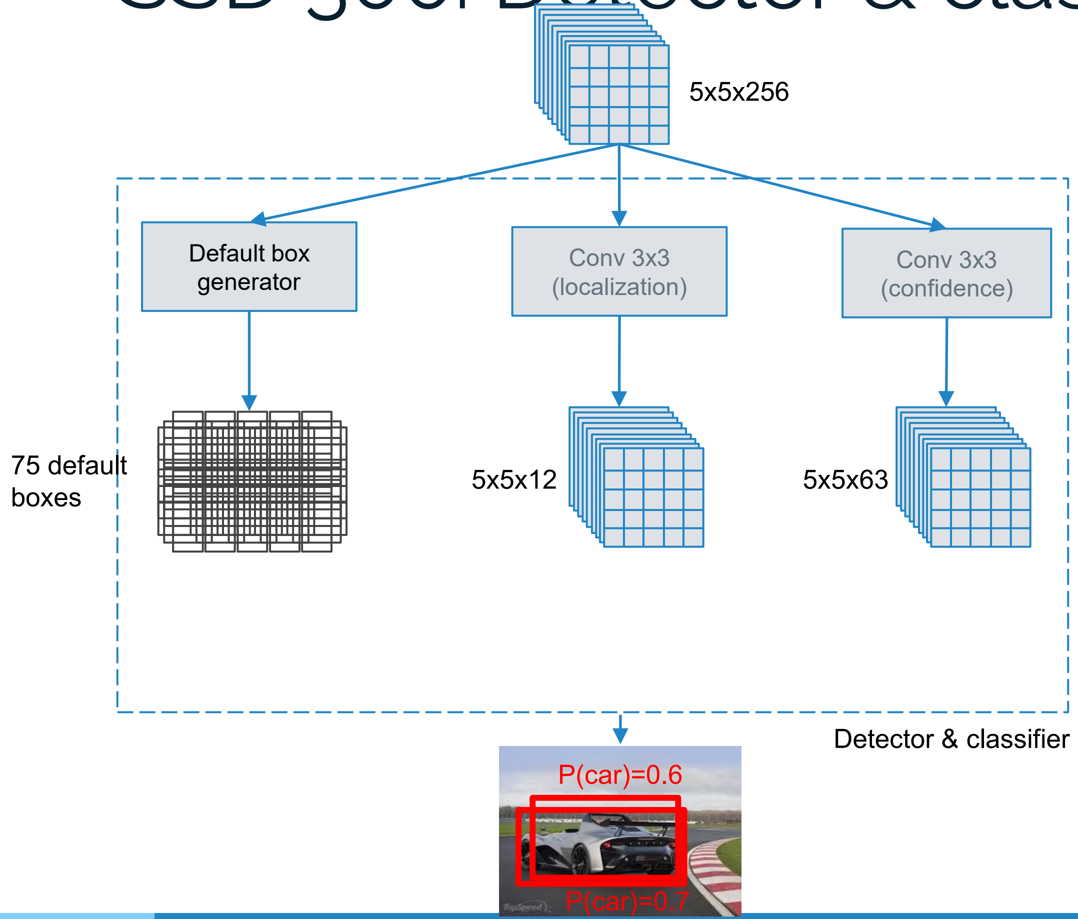
SSD 300: Detector & classifier



Considering the following parameters:

- Size of original image (300 x 300)
- Dimension of feature maps (5 x 5 x 256)
- #default boxes = 3

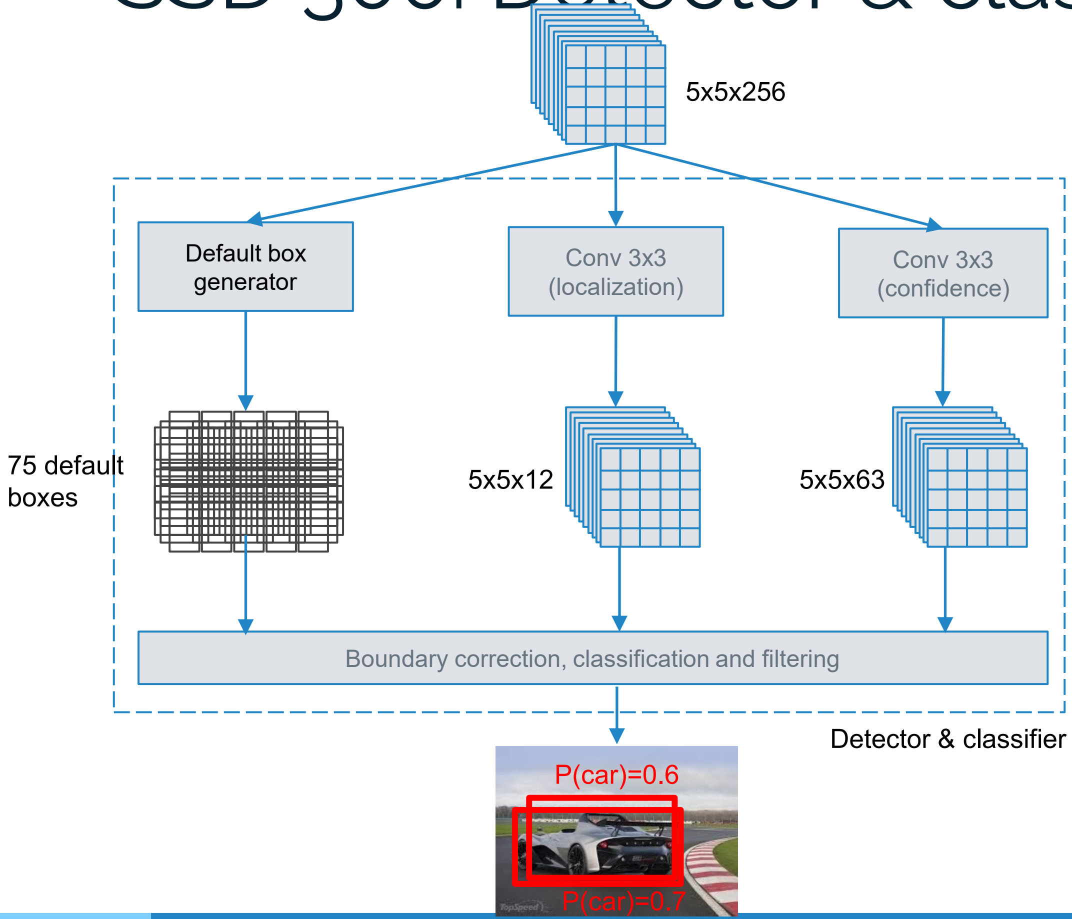
SSD 300: Detector & classifier



Considering the following parameters:

- Size of original image (300 x 300)
- Dimension of feature maps (5 x 5 x 256)
- #default boxes = 3

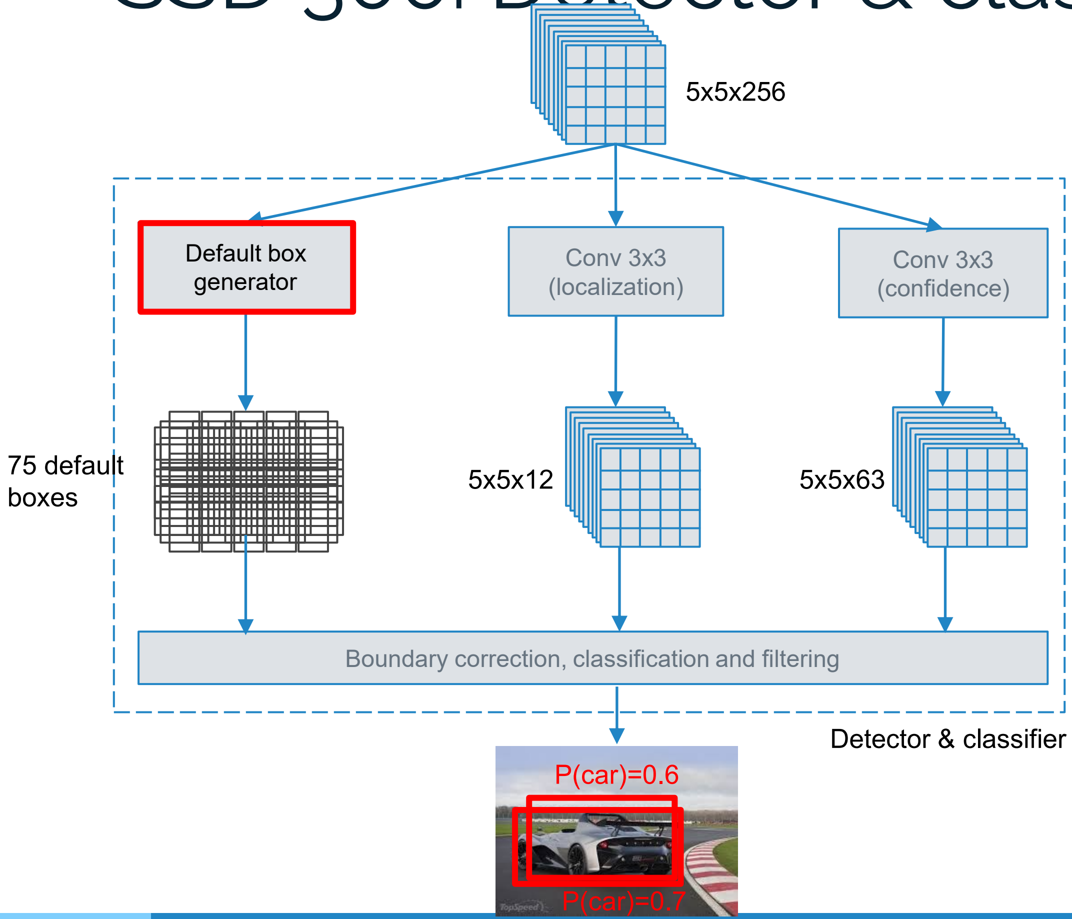
SSD 300: Detector & classifier



Considering the following parameters:

- Size of original image (300 x 300)
- Dimension of feature maps (5 x 5 x 256)
- #default boxes = 3

SSD 300: Detector & classifier



Considering the following parameters:

- Size of original image (300 x 300)
- Dimension of feature maps (5 x 5 x 256)
- #default boxes = 3

03

SSD ARCHITECTURE

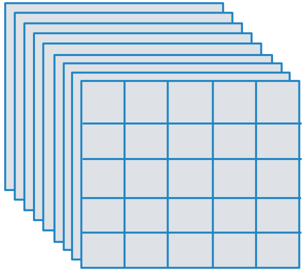
- Default box generation
- Boundary correction, classification and filtering

Default boxes generation

300

300

Input Image



Default boxes generation

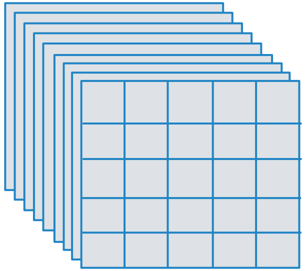
300

300

Input Image

Considering following parameters:

- Size of original image - (300 x 300)
- Spatial dimension of Feature maps (5 x 5)
- #default boxes = 3, (one point in feature map leads to 3 rectangles)
- min_size=168
- aspect_ratio=2

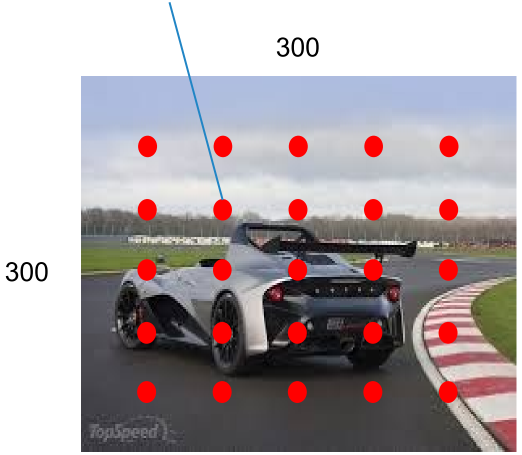


5

5

Default boxes generation

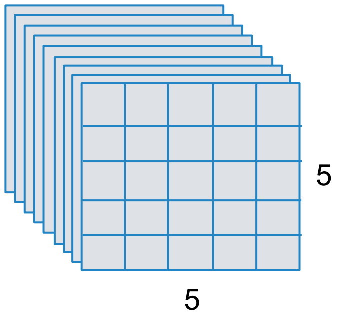
Centers of generated default boxes (xc, yc)



Input Image

Considering following parameters:

- Size of original image - (300 x 300)
- Spatial dimension of Feature maps (5 x 5)
- #default boxes = 3, (one point in feature map leads to 3 rectangles)
- min_size=168
- aspect_ratio=2

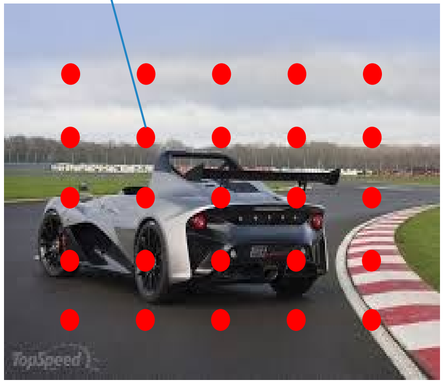


Default boxes generation

Centers of generated default boxes (xc, yc)

300

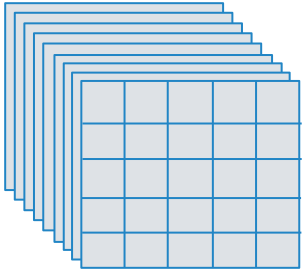
300



Input Image

Considering following parameters:

- Size of original image - (300 x 300)
- Spatial dimension of Feature maps (5 x 5)
- #default boxes = 3, (one point in feature map leads to 3 rectangles)
- min_size=168
- aspect_ratio=2



5

5

Default box is set by following values:

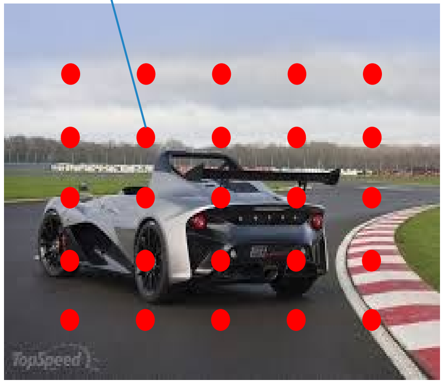
- xc, centre of the rectangle on x
- yc - centre of the rectangle on y
- w – width of the rectangle
- h – height of the rectangle

Default boxes generation

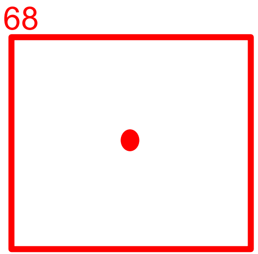
Centers of generated default boxes (xc, yc)

300

300

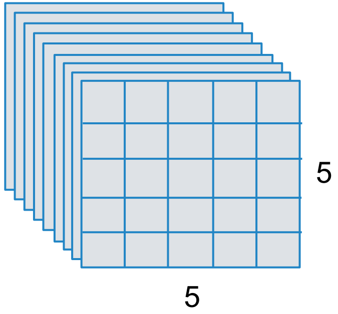


Input Image



Considering following parameters:

- Size of original image - (300 x 300)
- Spatial dimension of Feature maps (5 x 5)
- #default boxes = 3, (one point in feature map leads to 3 rectangles)
- min_size=168
- aspect_ratio=2



Default box is set by following values:

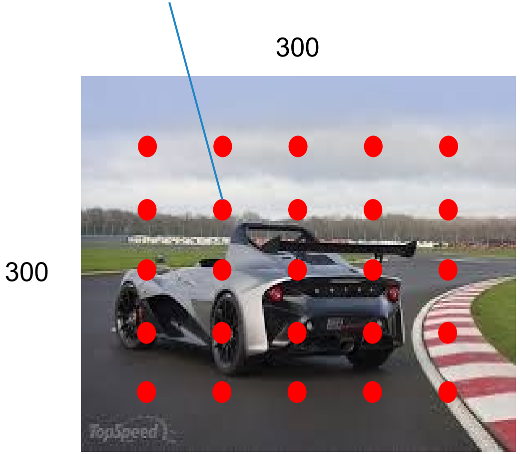
- xc, centre of the rectangle on x
- yc - centre of the rectangle on y
- w – width of the rectangle
- h – height of the rectangle

Default boxes generation

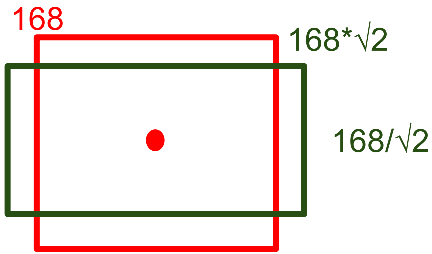
Considering following parameters:

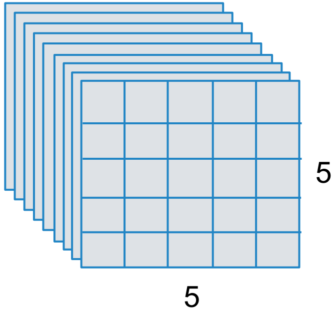
- Size of original image - (300 x 300)
- Spatial dimension of Feature maps (5 x 5)
- #default boxes = 3, (one point in feature map leads to 3 rectangles)
- min_size=168
- aspect_ratio=2

Centers of generated default boxes (xc, yc)



Input Image





Default box is set by following values:

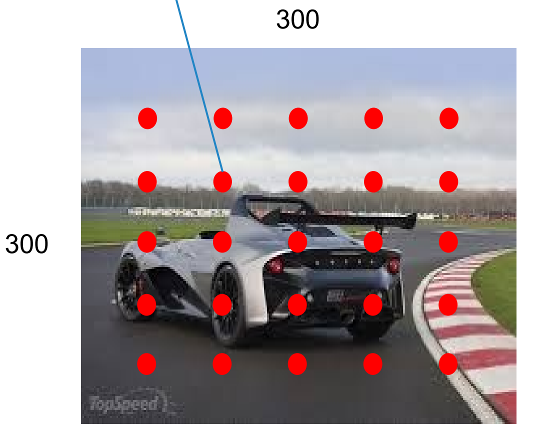
- xc, centre of the rectangle on x
- yc - centre of the rectangle on y
- w – width of the rectangle
- h – height of the rectangle

Default boxes generation

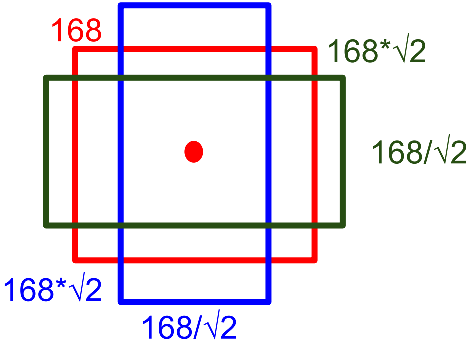
Considering following parameters:

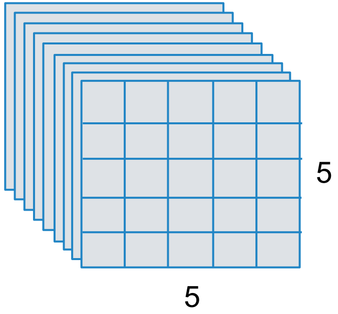
- Size of original image - (300 x 300)
- Spatial dimension of Feature maps (5 x 5)
- #default boxes = 3, (one point in feature map leads to 3 rectangles)
- min_size=168
- aspect_ratio=2

Centers of generated default boxes (xc, yc)



Input Image





Default box is set by following values:

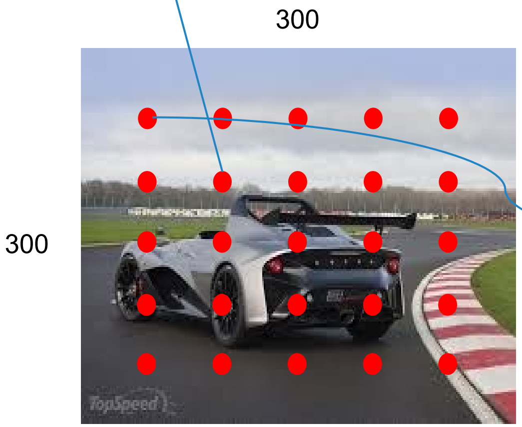
- xc, centre of the rectangle on x
- yc - centre of the rectangle on y
- w – width of the rectangle
- h – height of the rectangle

Default boxes generation

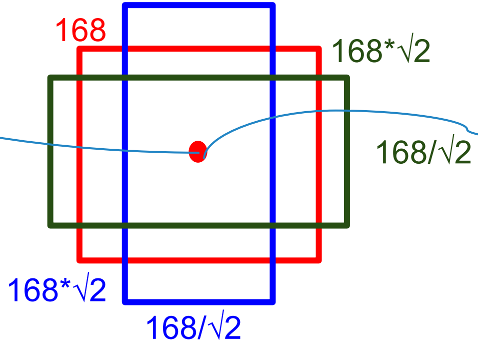
Considering following parameters:

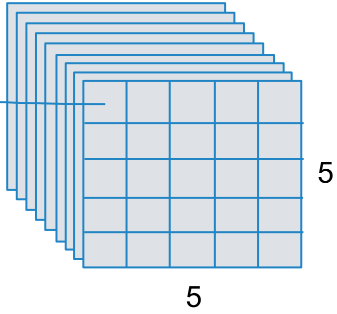
- Size of original image - (300 x 300)
- Spatial dimension of Feature maps (5 x 5)
- #default boxes = 3, (one point in feature map leads to 3 rectangles)
- min_size=168
- aspect_ratio=2

Centers of generated default boxes (xc, yc)



Input Image





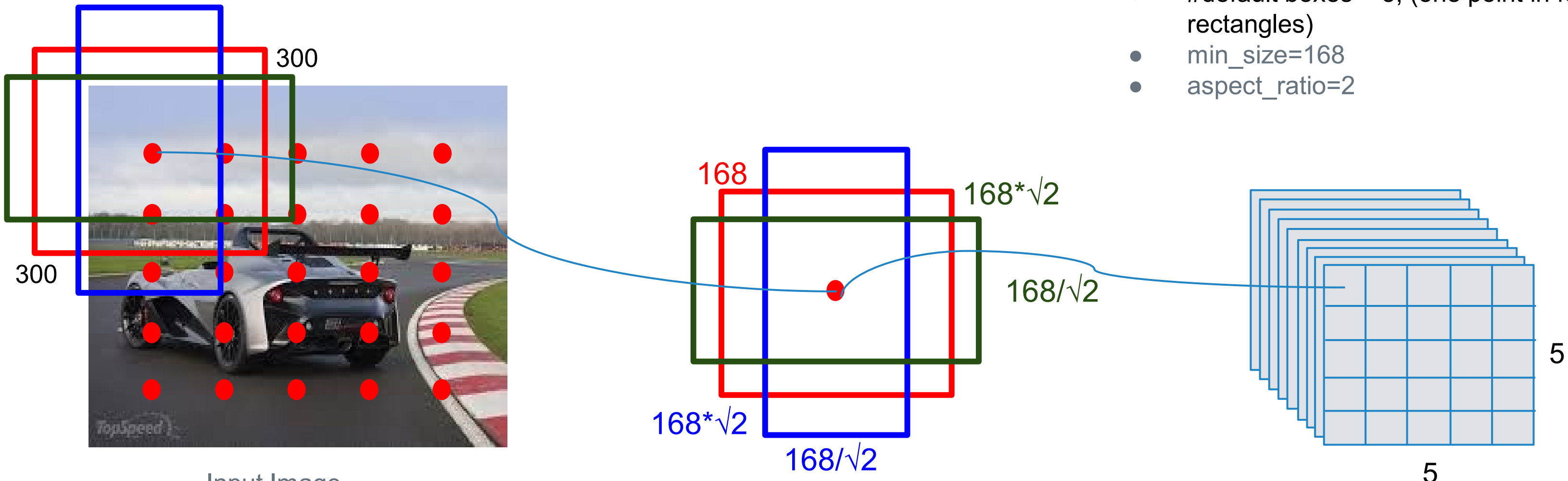
Default box is set by following values:

- xc, centre of the rectangle on x
- yc - centre of the rectangle on y
- w – width of the rectangle
- h – height of the rectangle

Default boxes generation

Considering following parameters:

- Size of original image - (300 x 300)
- Spatial dimension of Feature maps (5 x 5)
- #default boxes = 3, (one point in feature map leads to 3 rectangles)
- min_size=168
- aspect_ratio=2



Input Image

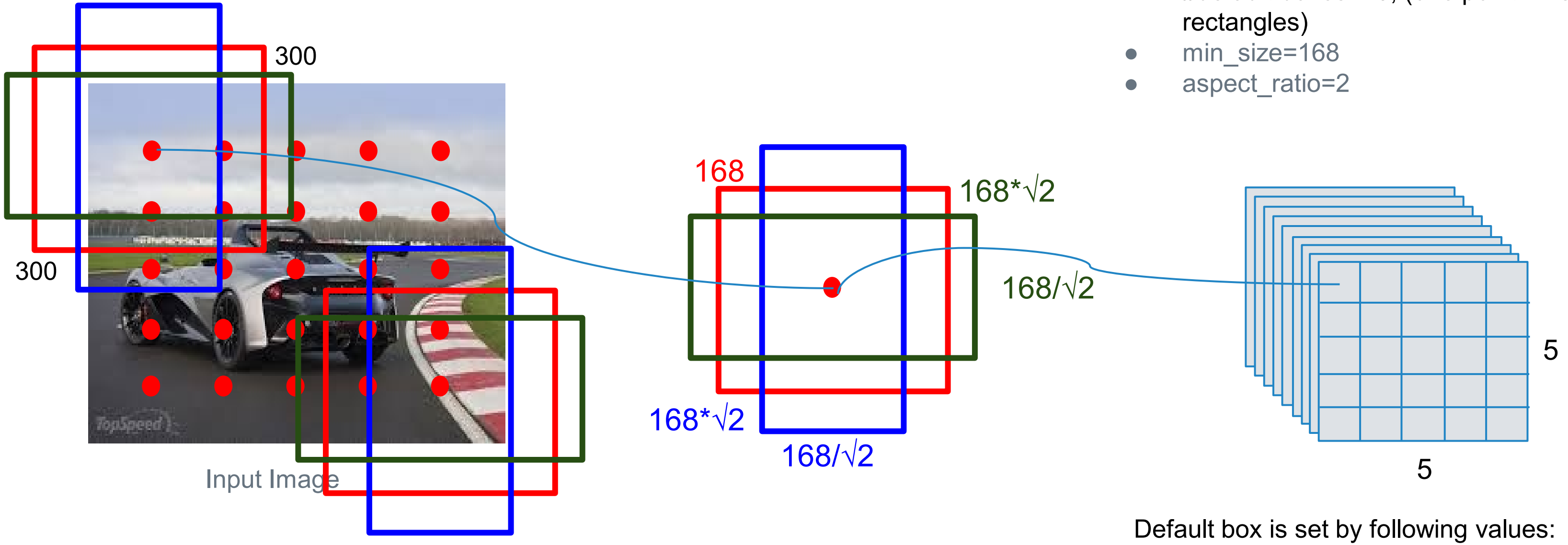
Default box is set by following values:

- xc, centre of the rectangle on x
- yc - centre of the rectangle on y
- w – width of the rectangle
- h – height of the rectangle

Default boxes generation

Considering following parameters:

- Size of original image - (300 x 300)
- Spatial dimension of Feature maps (5 x 5)
- #default boxes = 3, (one point in feature map leads to 3 rectangles)
- min_size=168
- aspect_ratio=2



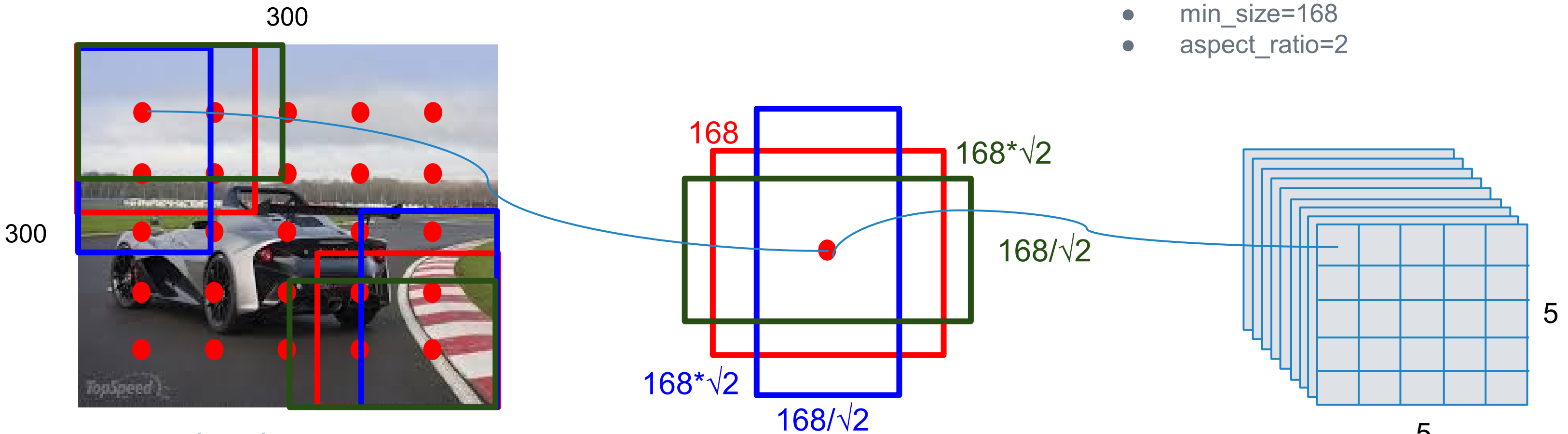
Default box is set by following values:

- xc, centre of the rectangle on x
- yc - centre of the rectangle on y
- w – width of the rectangle
- h – height of the rectangle

Default boxes generation

Considering following parameters:

- Size of original image - (300 x 300)
- Spatial dimension of Feature maps (5 x 5)
- #default boxes = 3, (one point in feature map leads to 3 rectangles)
- min_size=168
- aspect_ratio=2



Input Image

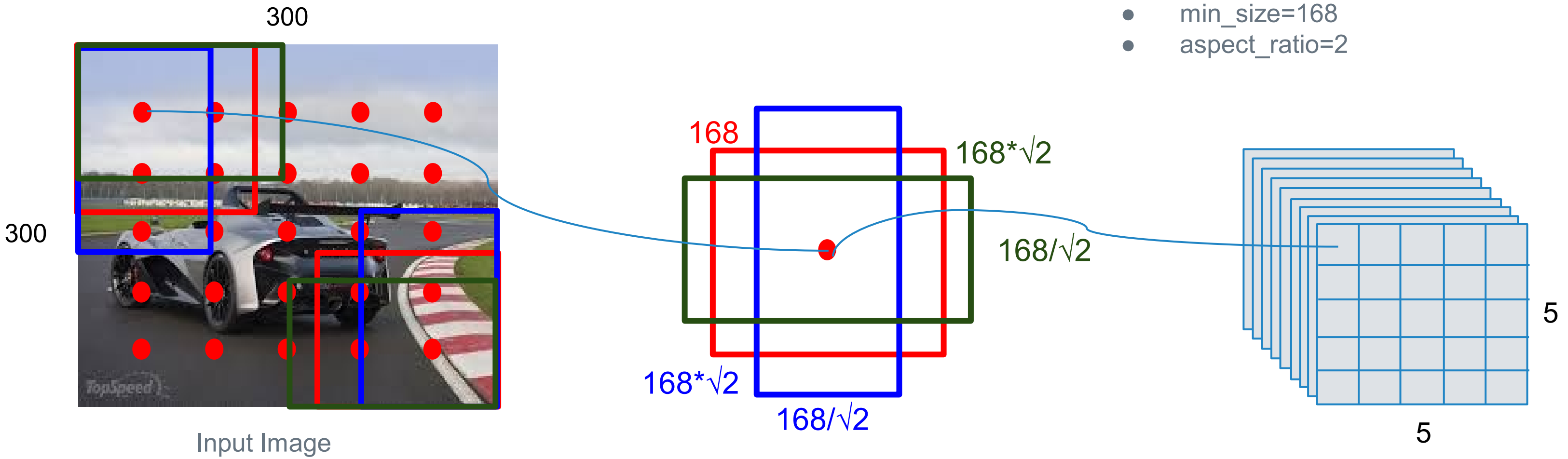
Default box is set by following values:

- xc, centre of the rectangle on x
- yc - centre of the rectangle on y
- w – width of the rectangle
- h – height of the rectangle

Default boxes generation

Considering following parameters:

- Size of original image - (300 x 300)
- Spatial dimension of Feature maps (5 x 5)
- #default boxes = 3, (one point in feature map leads to 3 rectangles)
- min_size=168
- aspect_ratio=2



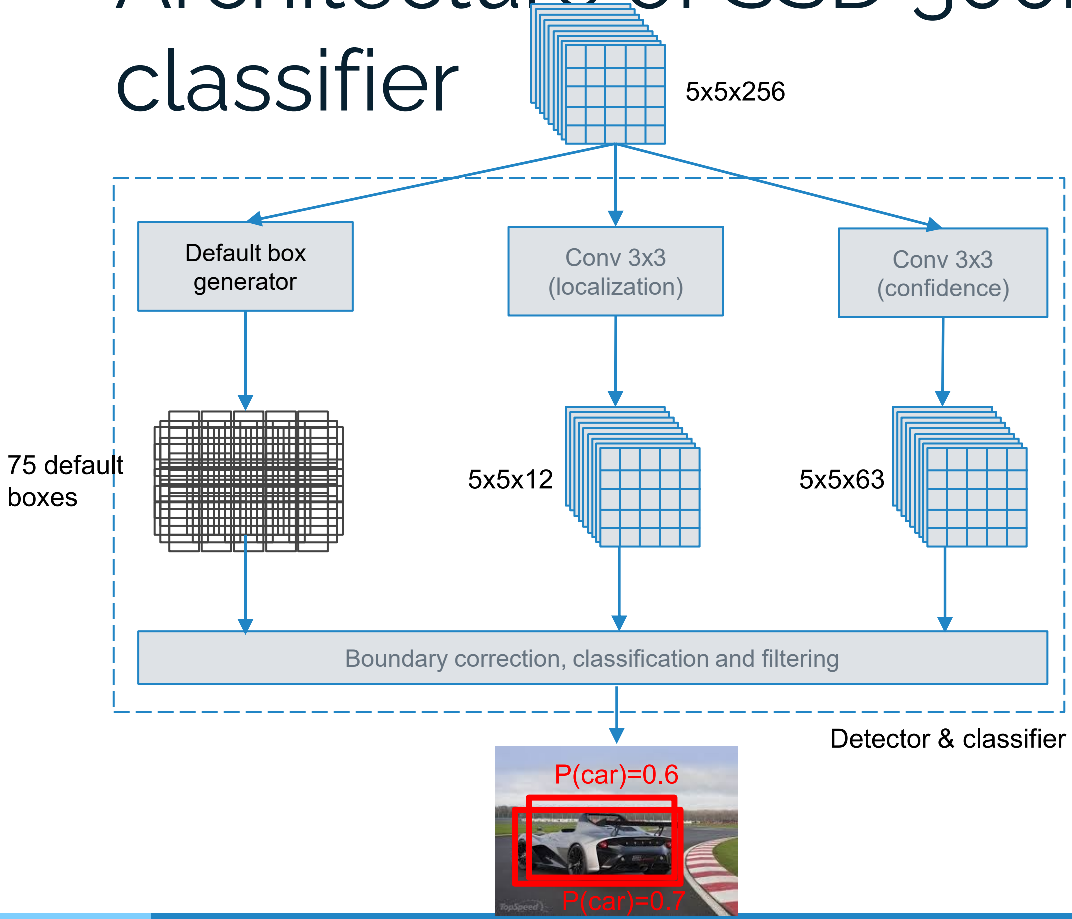
Input Image

Default box is set by following values:

- xc, centre of the rectangle on x
- yc - centre of the rectangle on y
- w – width of the rectangle
- h – height of the rectangle

A total of 5*5*3=75 rectangles will be generated

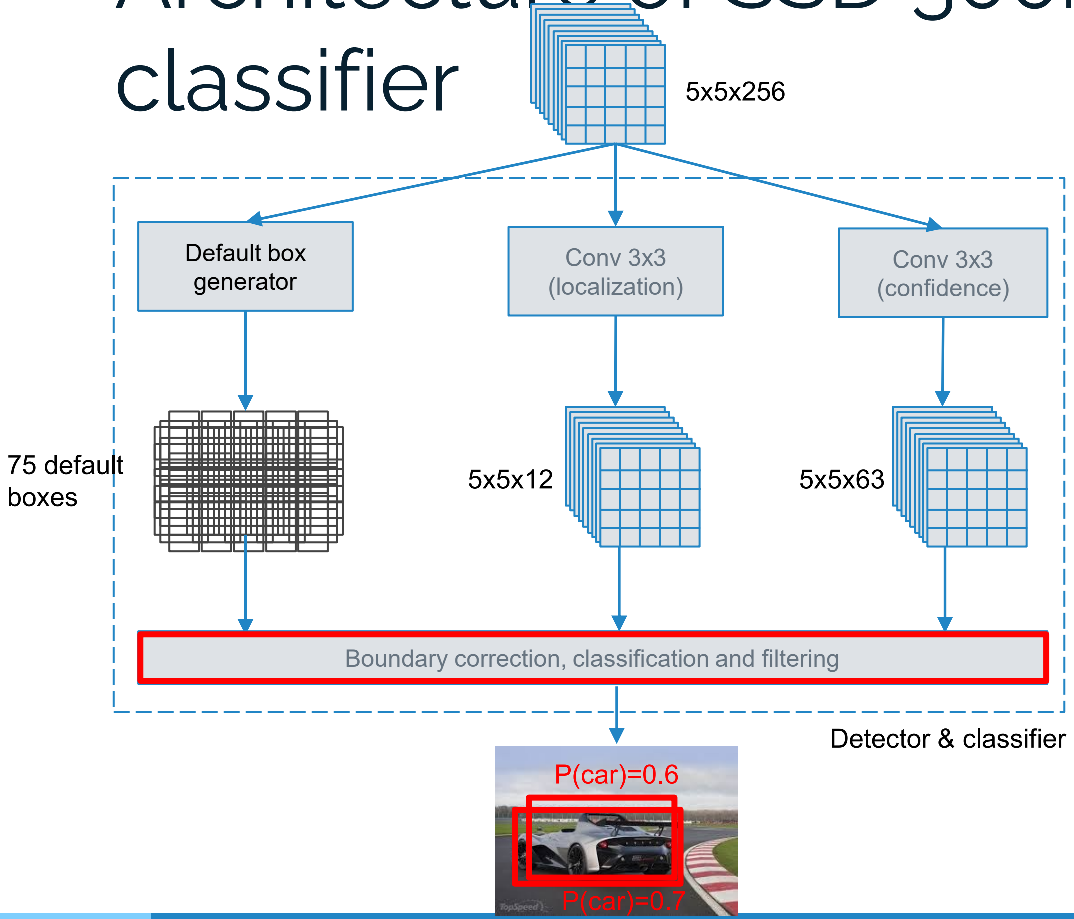
Architecture of SSD 300: Detector & classifier



Considering the following parameters:

- Size of original image (300 x 300)
- Dimension of feature maps (5 x 5 x 256)
- #default boxes = 3

Architecture of SSD 300: Detector & classifier

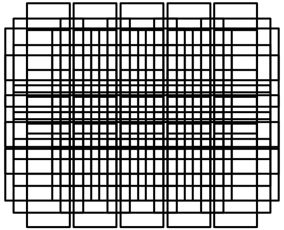


Considering the following parameters:

- Size of original image (300 x 300)
- Dimension of feature maps (5 x 5 x 256)
- #default boxes = 3

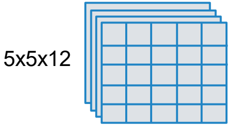
Boundary correction, classification and filtering (1)

Default boxes

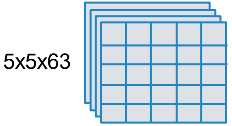


Only 5 * 5 * 3 = 75 rectangles

Localization

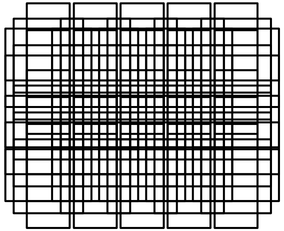


Confidence



Boundary correction, classification and filtering (1)

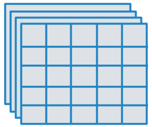
Default boxes



Only 5 * 5 * 3 = 75 rectangles

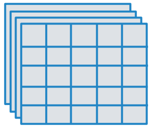
Localization

5x5x12



Confidence

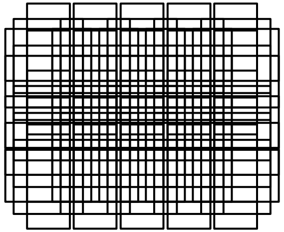
5x5x63



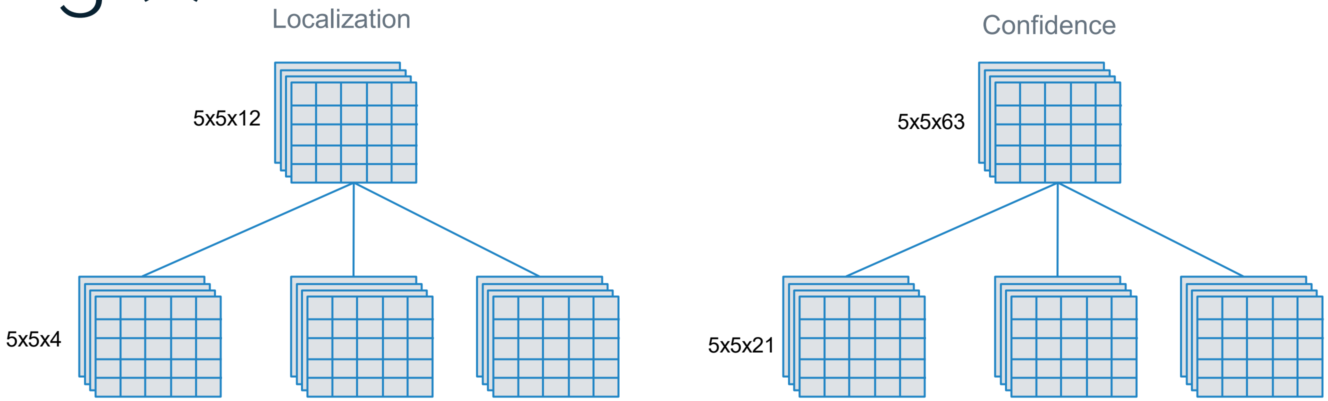
Input Image (300 x 300)

Boundary correction, classification and filtering (1)

Default boxes

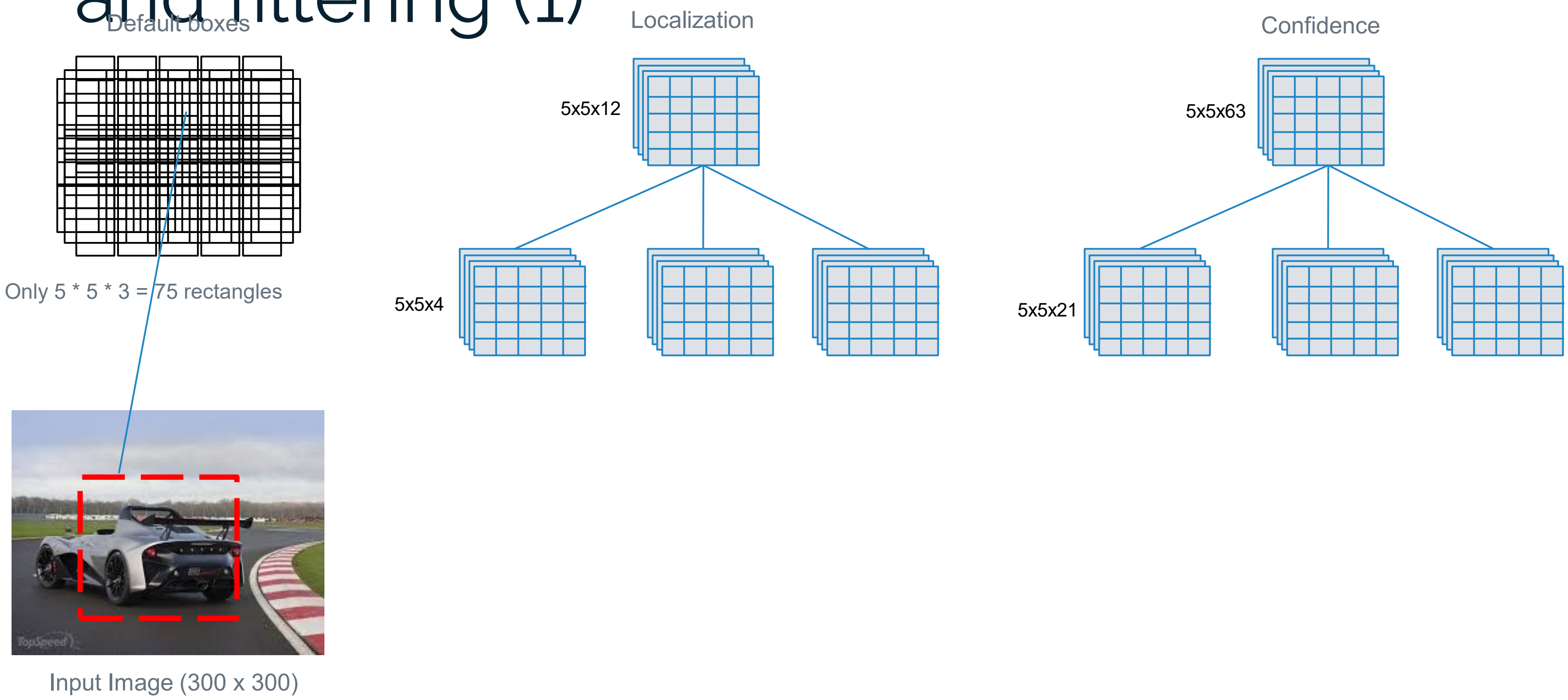


Only 5 * 5 * 3 = 75 rectangles

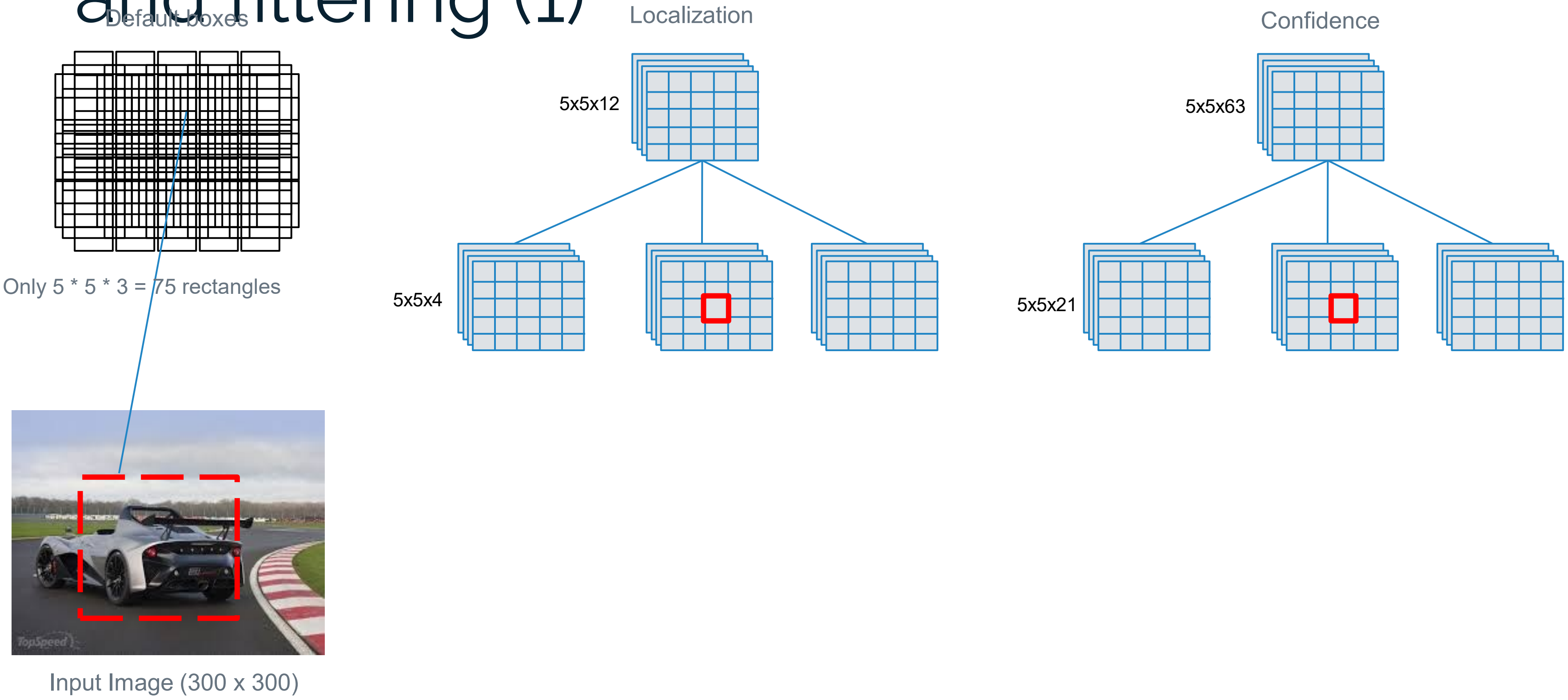


Input Image (300 x 300)

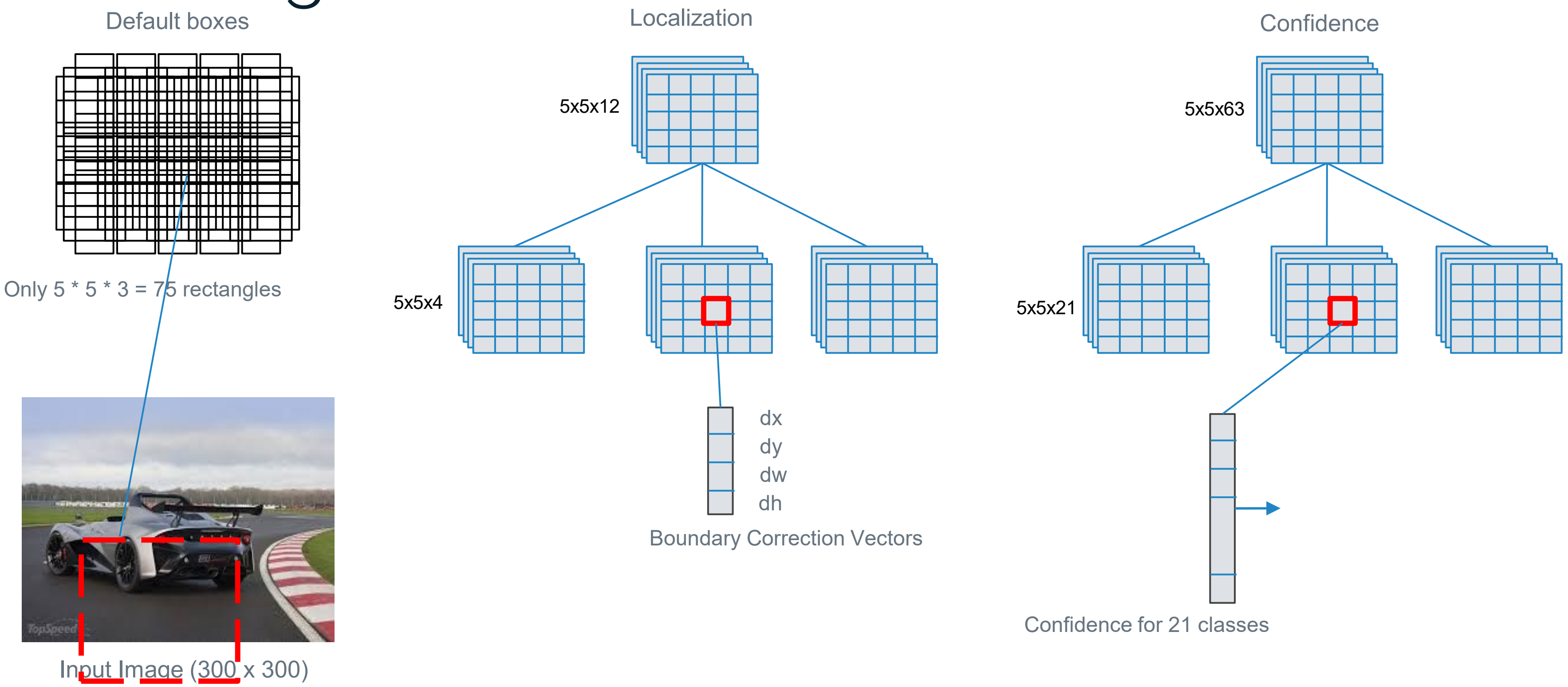
Boundary correction, classification and filtering (1)



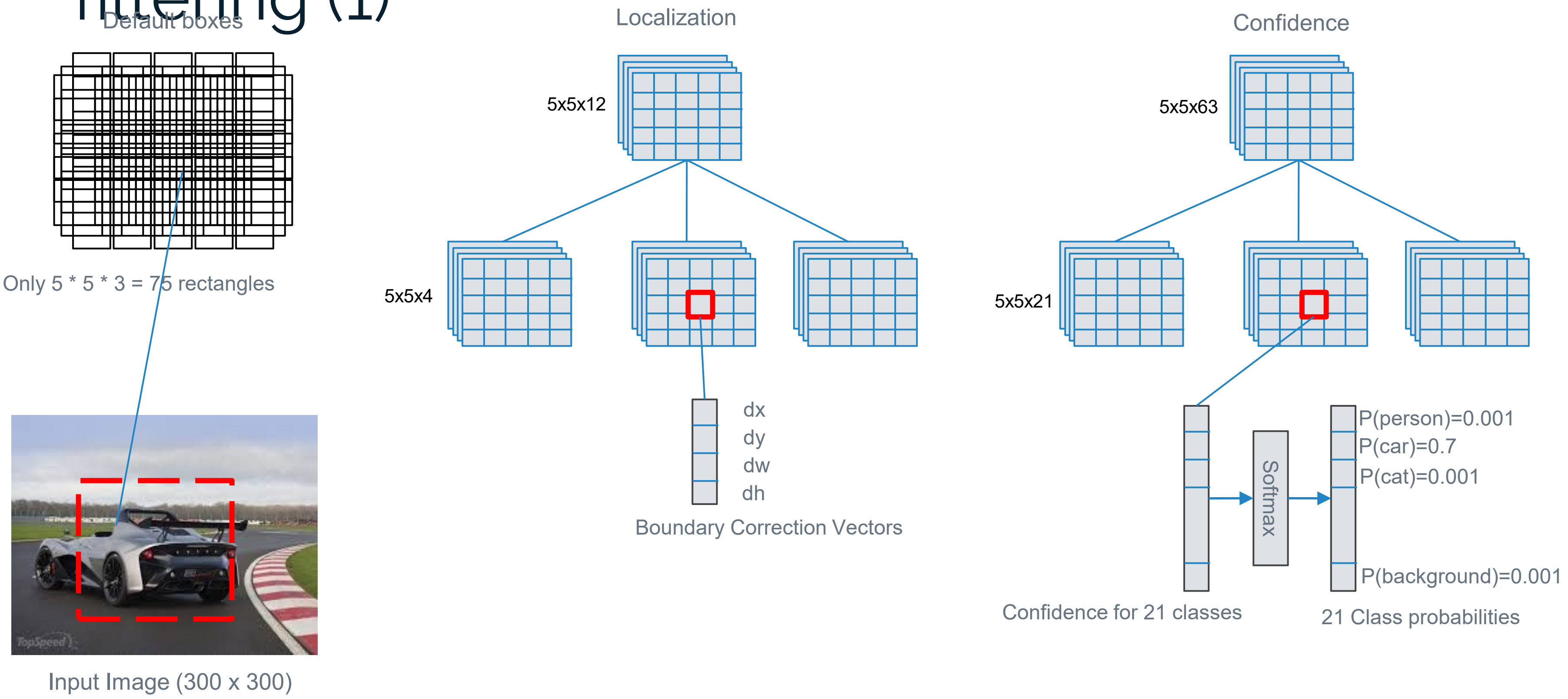
Boundary correction, classification and filtering (1)



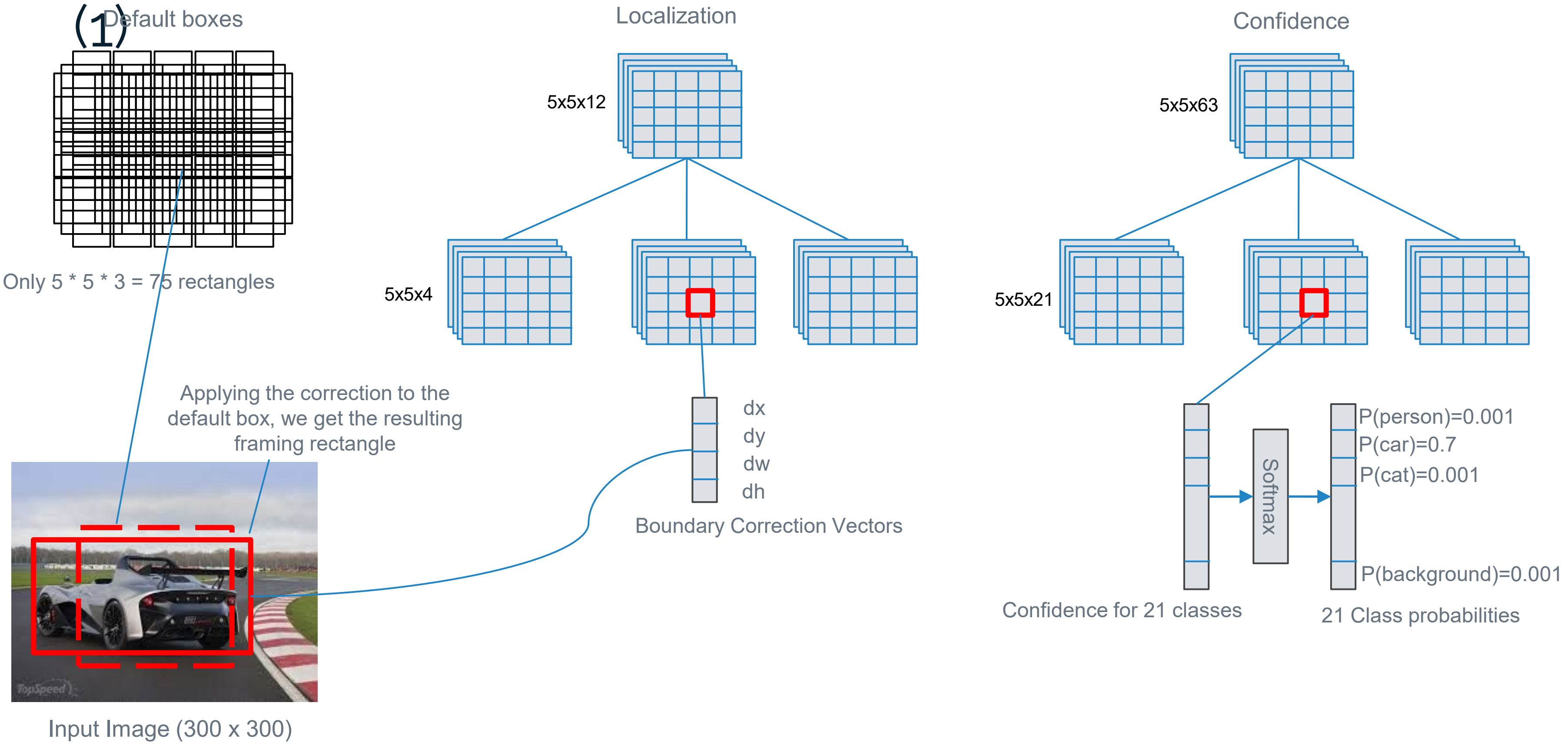
Boundary correction, classification and filtering (1)



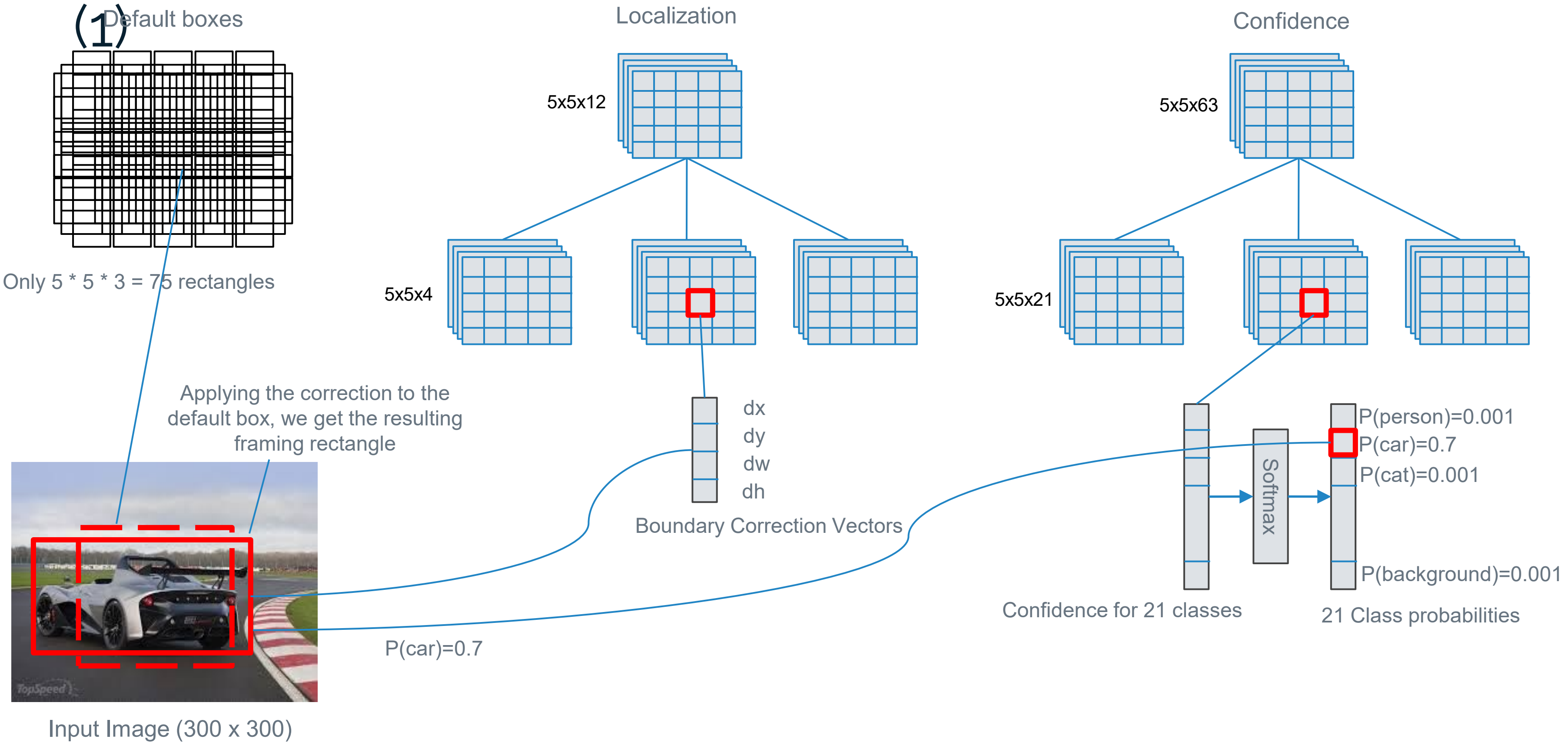
Boundary correction, classification and filtering (1)



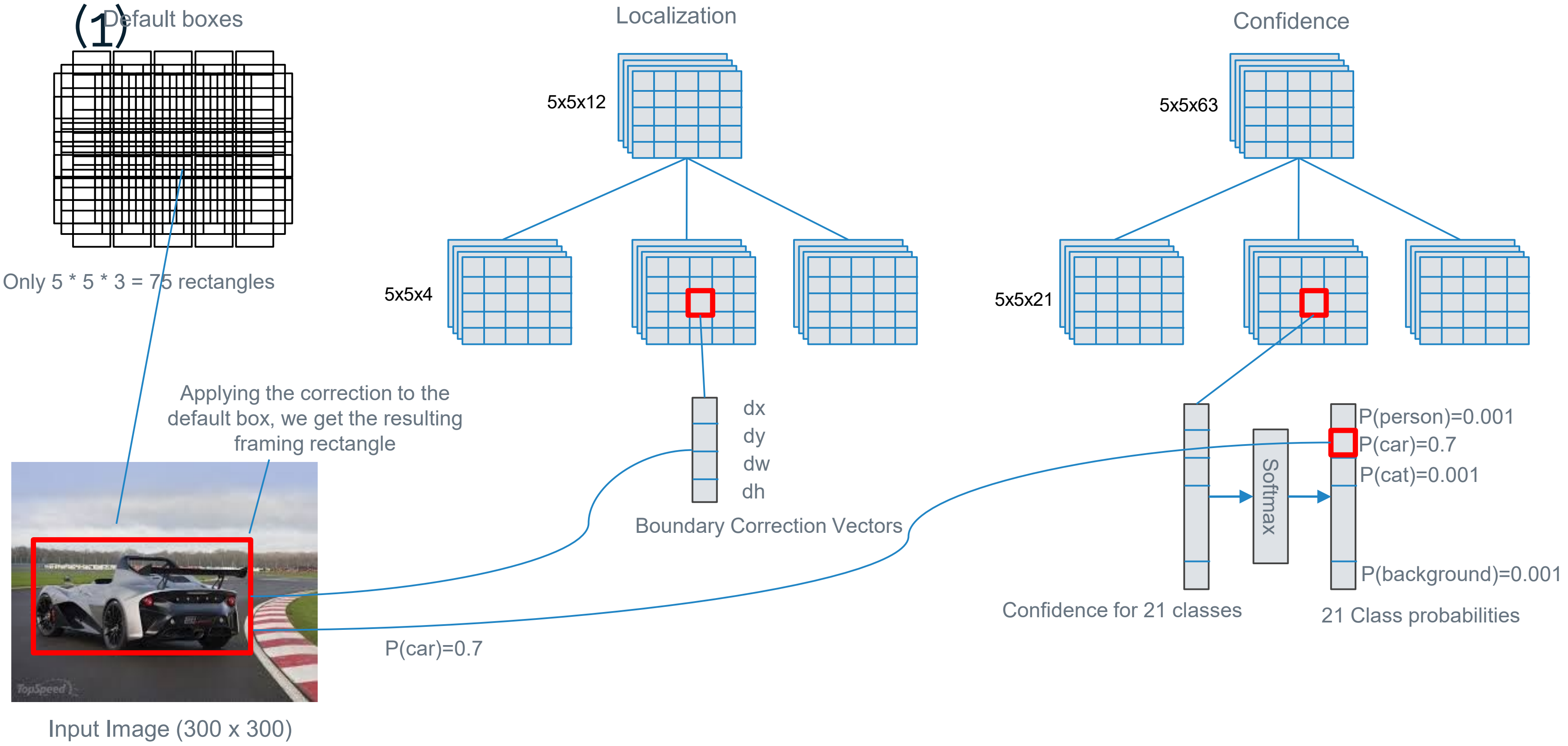
Boundary correction, classification and filtering (1)



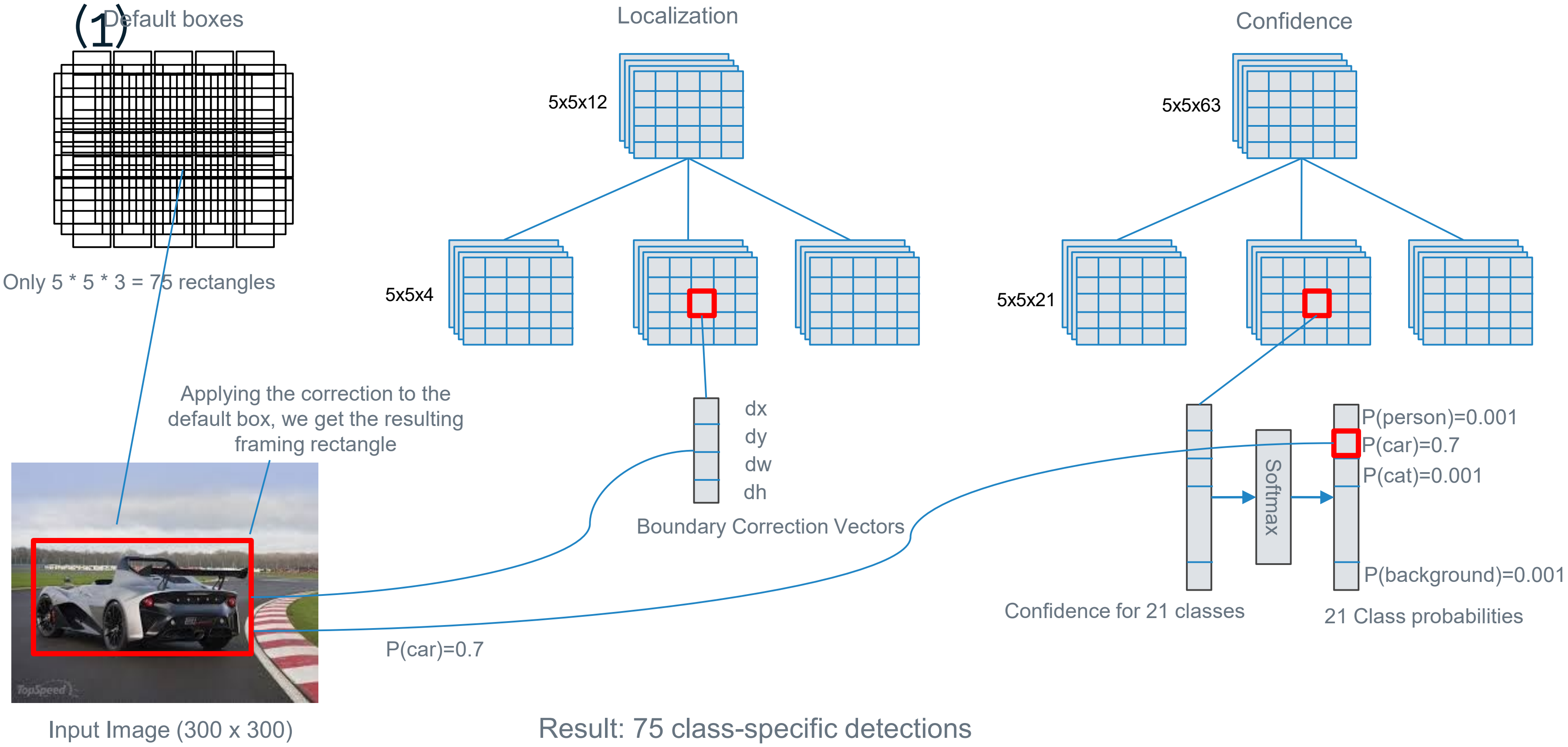
Boundary correction, classification and filtering (1)



Boundary correction, classification and filtering (1)



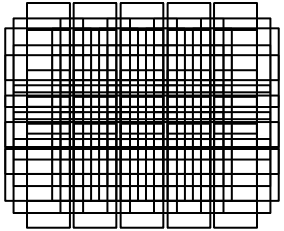
Boundary correction, classification and filtering (1)



Result: 75 class-specific detections

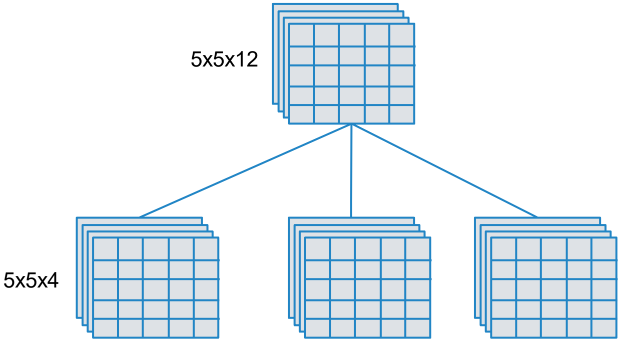
Boundary correction, classification and filtering (2)

Default boxes

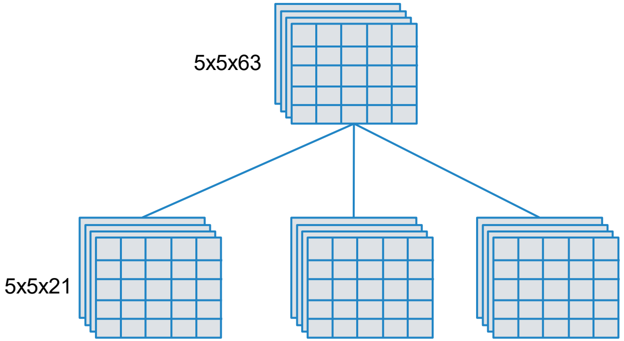


Only 5 * 5 * 3 = 75 rectangles

Localization

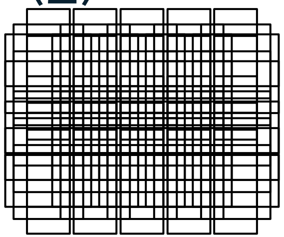


Confidence

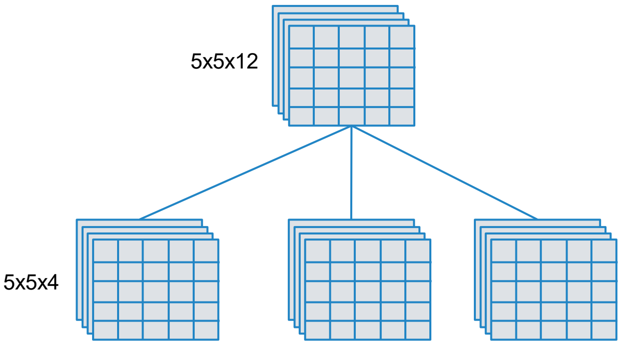


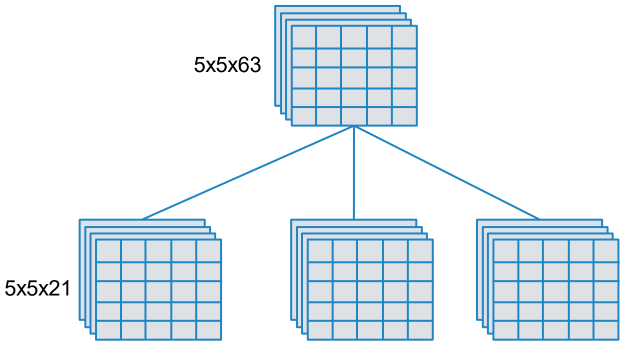
75 detectable objects

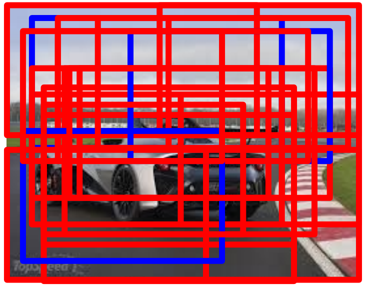
Boundary correction, classification and filtering (2)



Only 5 * 5 * 3 = 75 rectangles



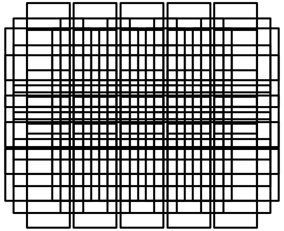




75 detectable objects

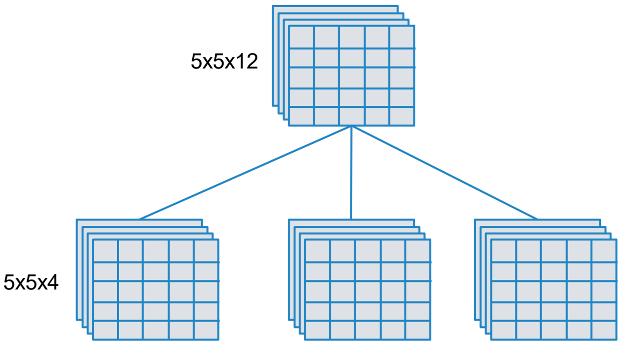
Boundary correction, classification and filtering (2)

Default boxes

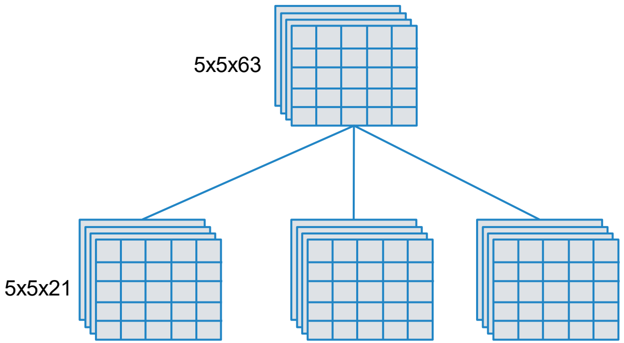


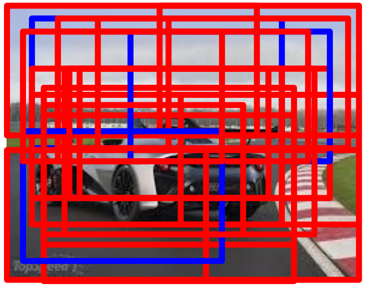
Only 5 * 5 * 3 = 75 rectangles

Localization



Confidence

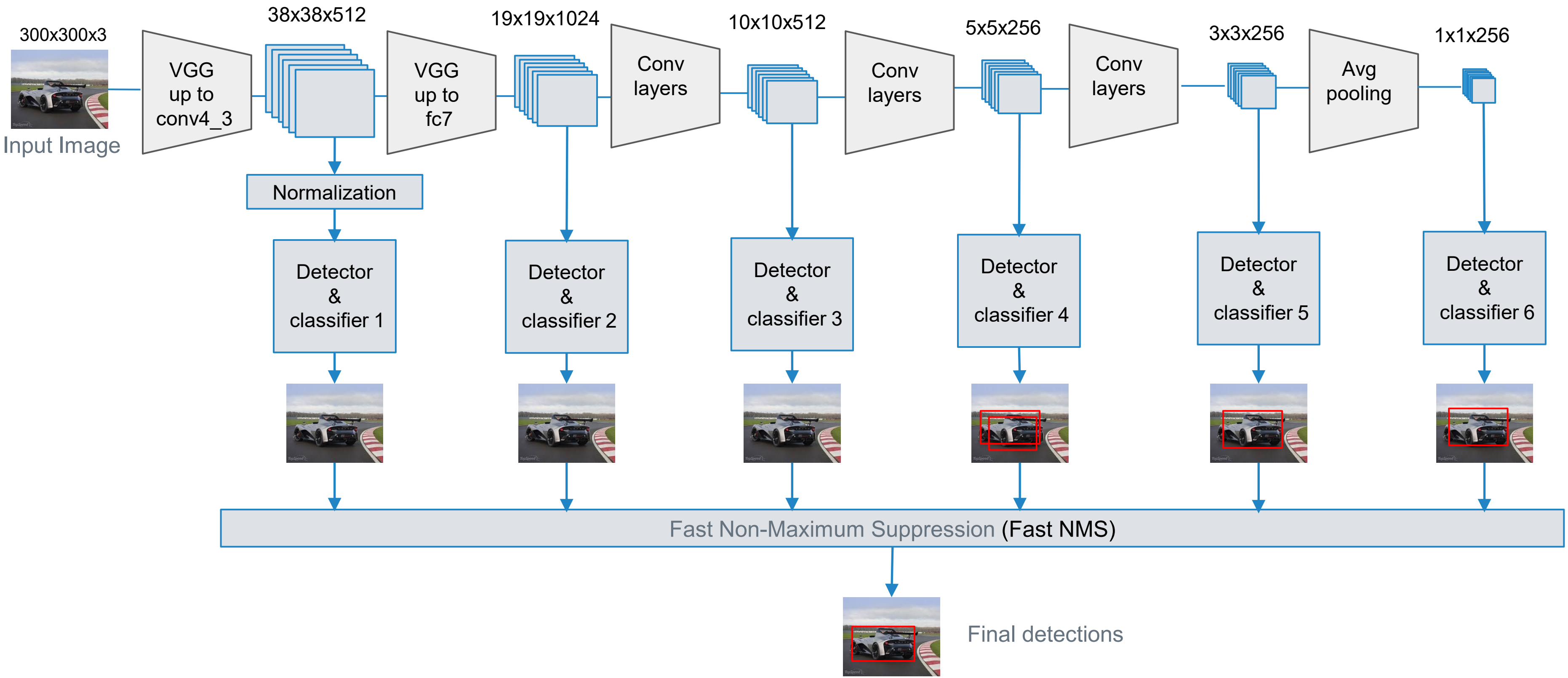




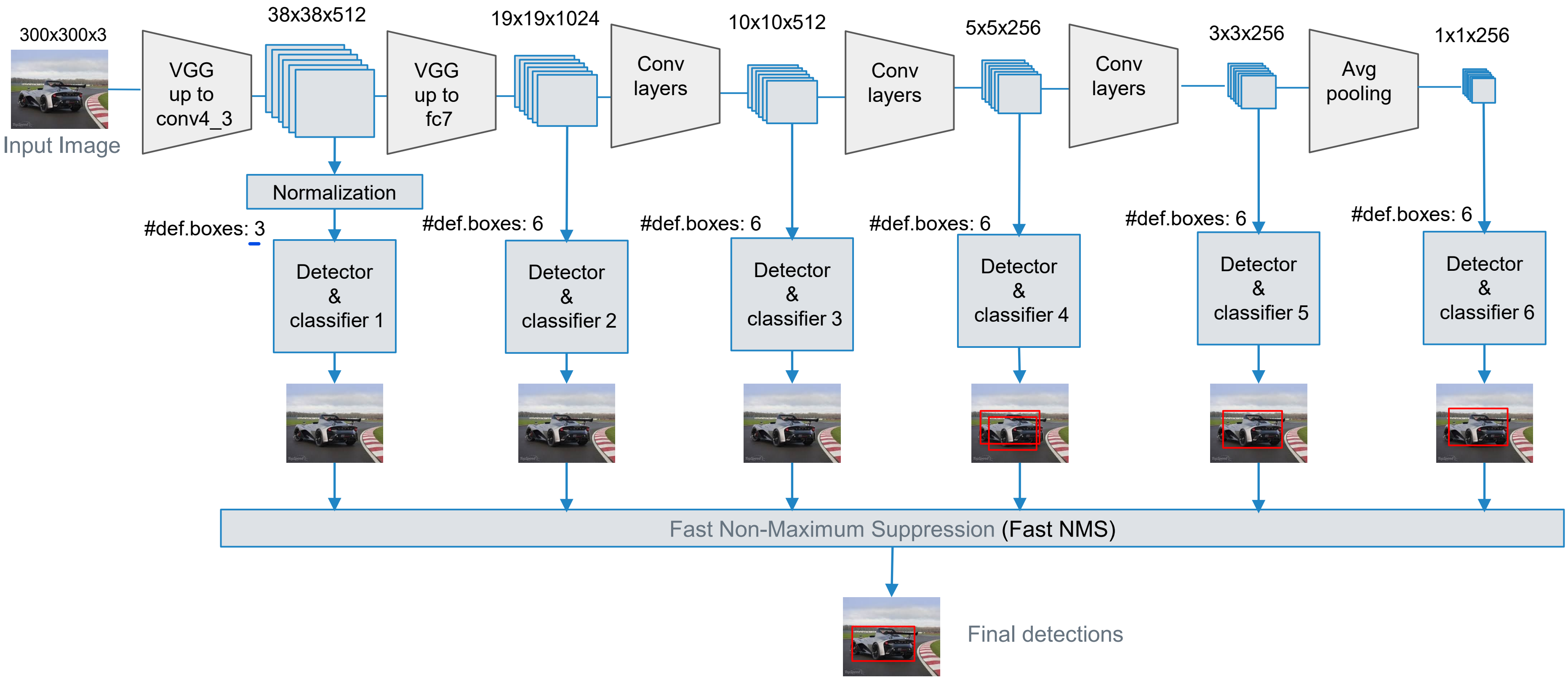
75 detectable objects

Confidence filtering

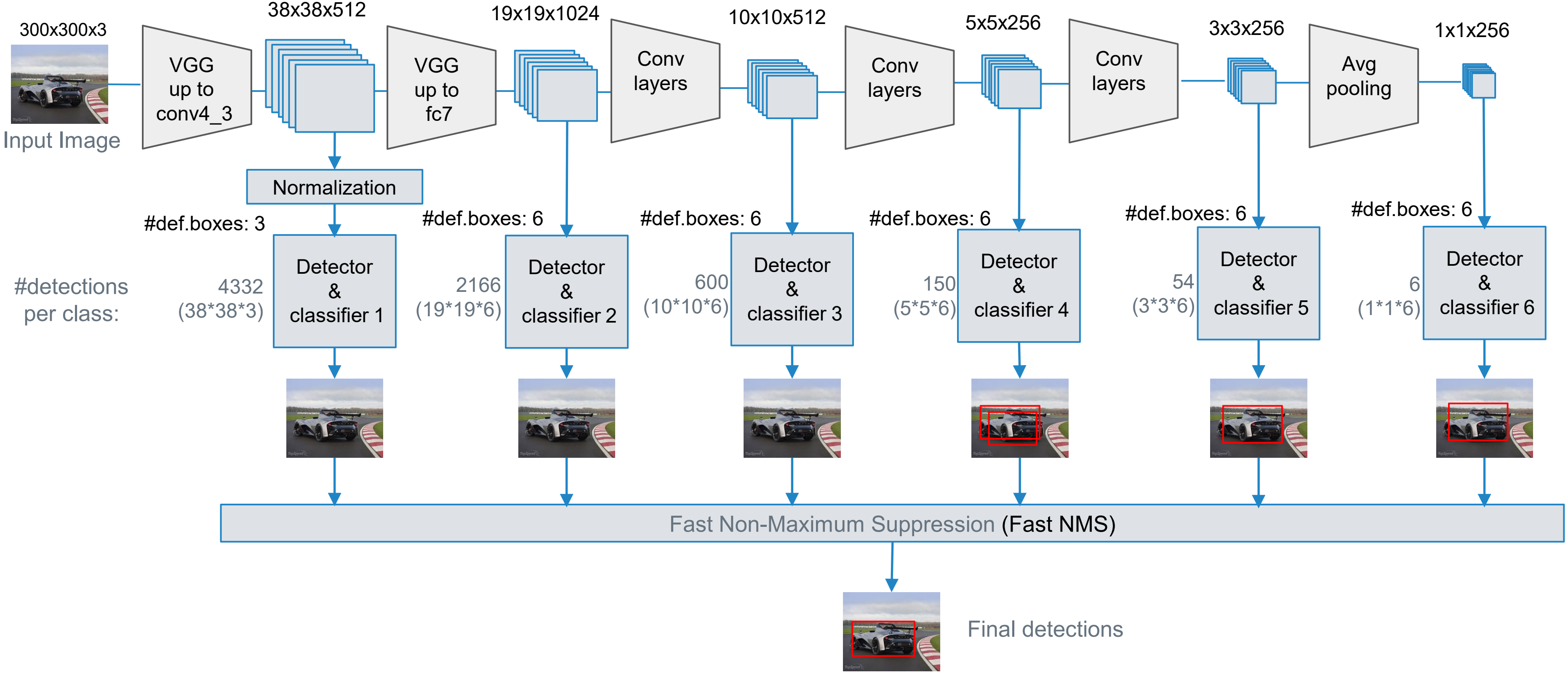
SSD 300 Architecture



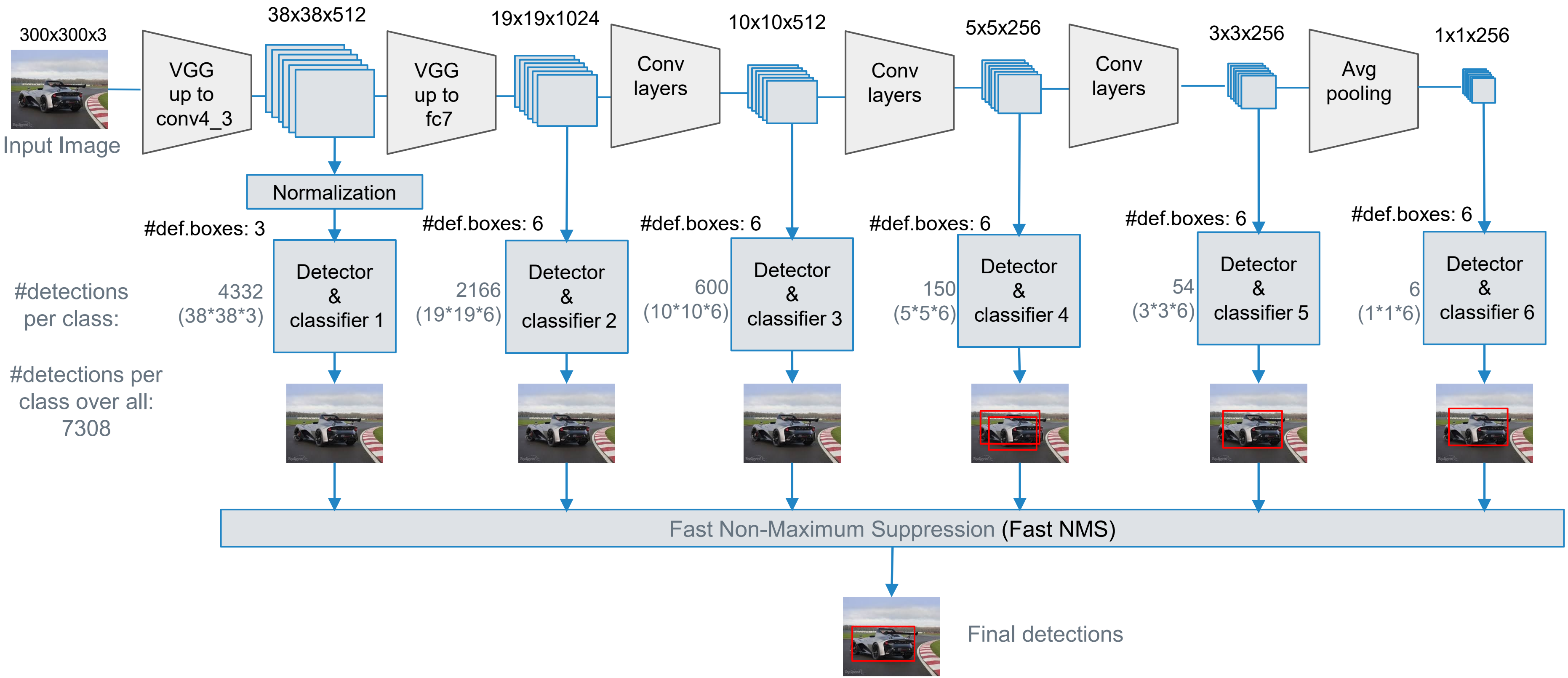
SSD 300 Architecture



SSD 300 Architecture



SSD 300 Architecture



Key Points

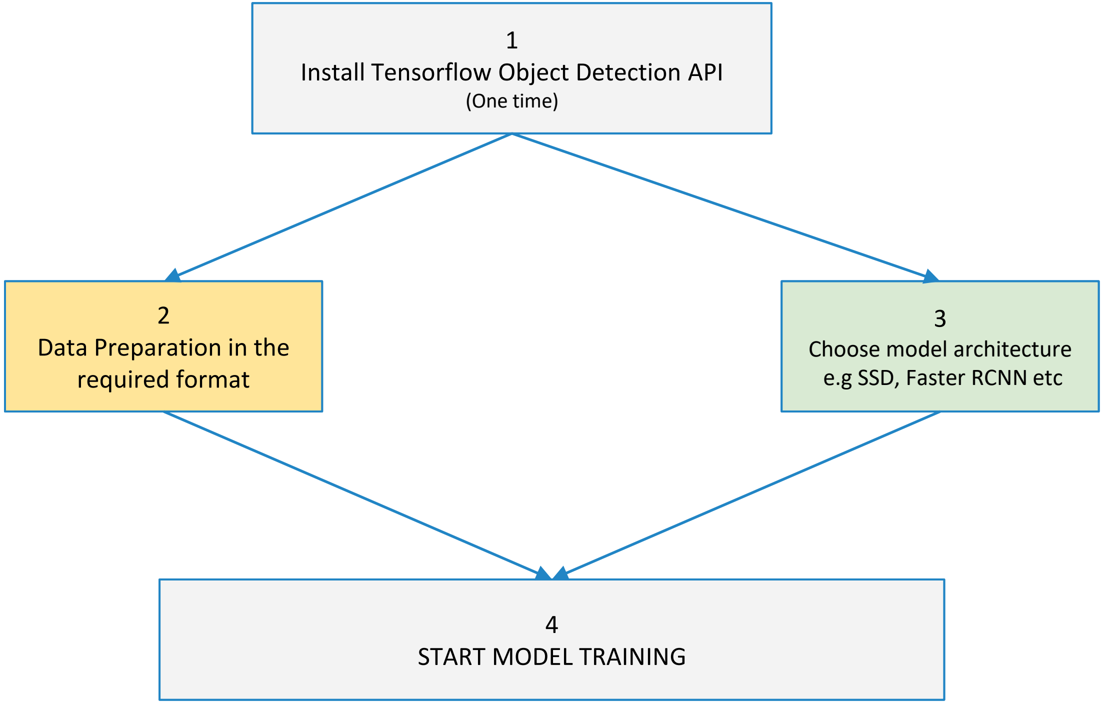
- SSD architecture allows real-time detection of objects
- Performance close to Faster R-CNN
- Detection takes place at different scales, which allows you to localize objects of different sizes
- A large number of default boxes are used, covering the input image at different scales.
- At the Inference stage, the SSD 300 architecture detects 7308 objects, most of which are subsequently filtered

04

TF OD API

- SSD implementation using TF OD API

SSD implementation using TF OD API



SSD implementation using TF OD API

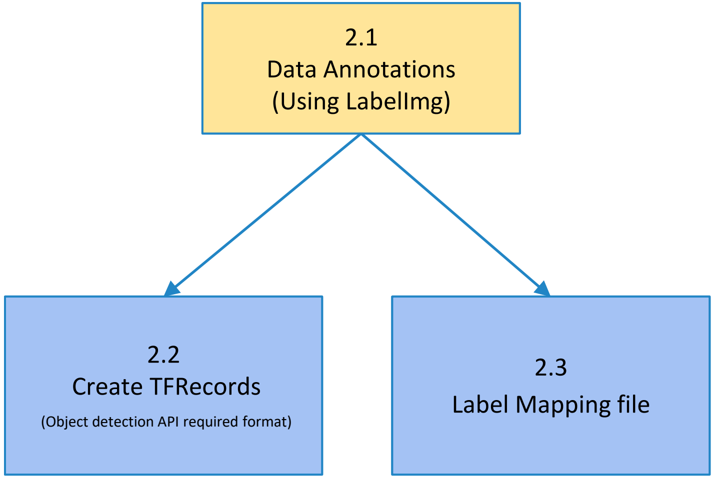
Installation Documentation

https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/installation.md

1. Installing TensorFlow Object Detection API

One time (on Google Colab every time)

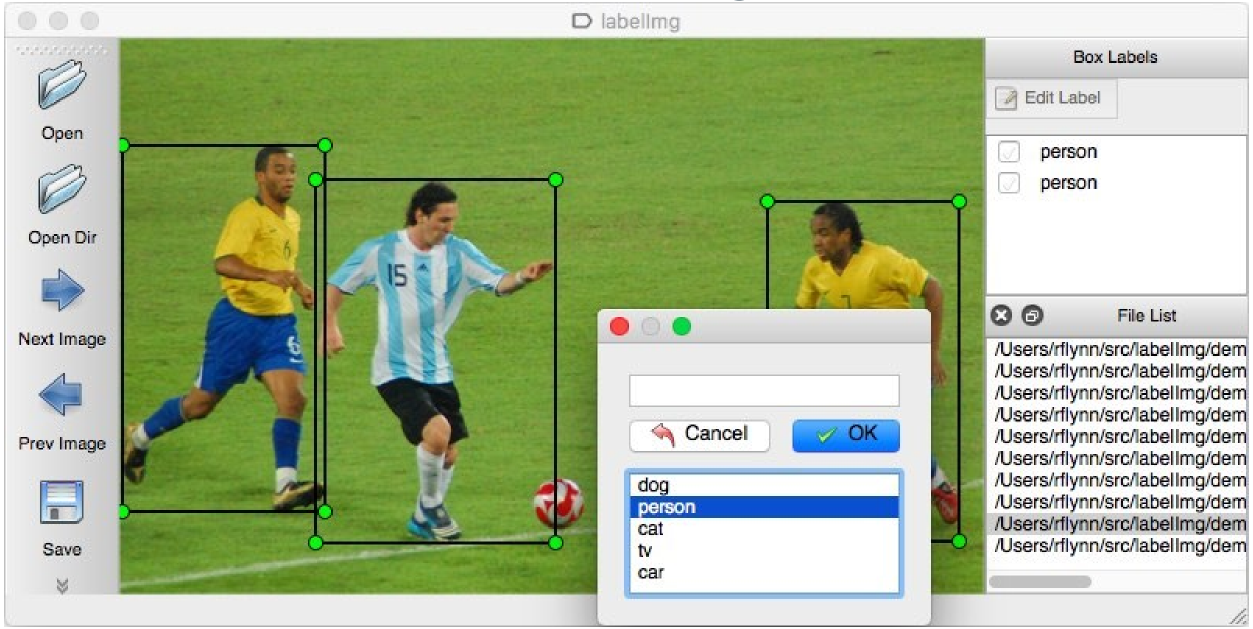
SSD implementation using TF OD API



2. Data Preparation

Involves 3 key steps

SSD implementation using TF OD API



2.1 Data Annotations

Use LabelImg tool (https://github.com/tzutalin/labelImg) for data annotation. It will create an XML file for each image. This will act as input to Object Detection API.

SSD implementation using TF OD API

Image Annotations in PASCAL VOC Format

xml_to_csv.py script

CSV file information from all XML files

(XML file from LabelImg)

	filename	width	height	class	xmin	ymin	xmax	ymax
0	003223.jpg	375	500	boat	136	230	230	299
1	006436.jpg	500	375	boat	170	36	376	282
2	005877.jpg	500	333	horse	106	40	219	333
3	005877.jpg	500	333	horse	292	38	484	333
4	009743.jpg	500	379	cat	150	165	342	220

Key observations about CSV file:

1. It has 7 columns.
2. It will have one record for each object
3. An image can have multiple records in csv file if it has multiple objects

2.2 Creating tfrecord files

Multi-step process but steps are repeatable for every dataset

SSD implementation using TF OD API

Image Annotations in PASCAL VOC Format → CSV file information from all XML files

(XML file from LabelImg)

	filename	width	height	class	xmin	ymin	xmax	ymax
0	003223.jpg	375	500	boat	136	230	230	299
1	006436.jpg	500	375	boat	170	36	376	282
2	005877.jpg	500	333	horse	106	40	219	333
3	005877.jpg	500	333	horse	292	38	484	333
4	009743.jpg	500	379	cat	150	165	342	220

	filename	width	height	class	xmin	ymin	xmax	ymax	label
0	003223.jpg	375	500	boat	136	230	230	299	4
1	006436.jpg	500	375	boat	170	36	376	282	4
2	005877.jpg	500	333	horse	106	40	219	333	13
3	005877.jpg	500	333	horse	292	38	484	333	13
4	009743.jpg	500	379	cat	150	165	342	220	8

Label encode 'class' with unique index (starting at 1)

2.2 Creating tfrecord files

Multi-step process but steps are repeatable for every dataset

SSD implementation using TF OD API

Image Annotations in PASCAL VOC Format

(XML file from LabelImg)

CSV file information from all XML files

	filename	width	height	class	xmin	ymin	xmax	ymax
0	003223.jpg	375	500	boat	136	230	230	299
1	006436.jpg	500	375	boat	170	36	376	282
2	005877.jpg	500	333	horse	106	40	219	333
3	005877.jpg	500	333	horse	292	38	484	333
4	009743.jpg	500	379	cat	150	165	342	220

Label encode ‘class’ with unique index (starting at 1)

	filename	width	height	class	xmin	ymin	xmax	ymax	label
0	003223.jpg	375	500	boat	136	230	230	299	4
1	006436.jpg	500	375	boat	170	36	376	282	4
2	005877.jpg	500	333	horse	106	40	219	333	13
3	005877.jpg	500	333	horse	292	38	484	333	13
4	009743.jpg	500	379	cat	150	165	342	220	8

Split data between training & test

Training & Test CSV files

generate_tfrecord.py script

Training & Test tfrecord files

2.2 Creating tfrecord files

Multi-step process but steps are repeatable for every dataset

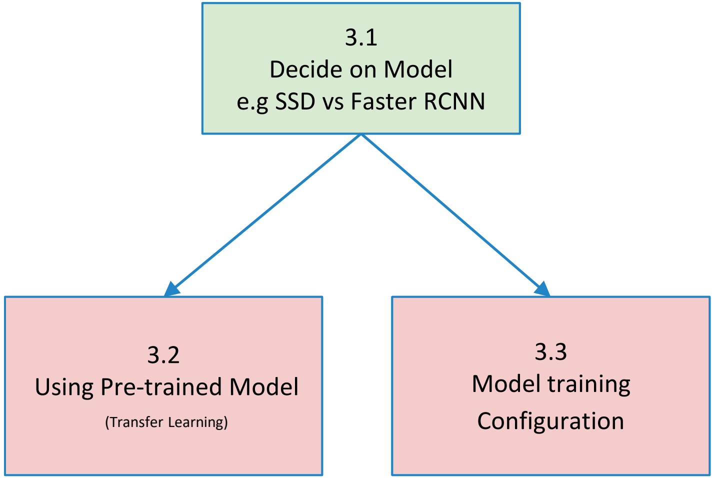
SSD implementation using TF OD API

```
item {
  id: 1
  name: 'aeroplane'
}

item {
  id: 2
  name: 'bicycle'
}

item {
  id: 3
  name: 'bird'
}

item {
  id: 4
  name: 'boat'
}

item {
  id: 5
  name: 'bottle'
}

item {
  id: 6
  name: 'bus'
}
```

2.3 Creating Label Mapping file

Single file with ‘item’ dict with unique class ‘id’ and ‘name’. The ‘id’ should start from ‘1’. Make sure class name has quotes around it.

SSD implementation using TF OD API



3. Choose Model Architecture

API supports both Faster RCNN and SSD

SSD implementation using TF OD API

TensorFlow Object Detection API

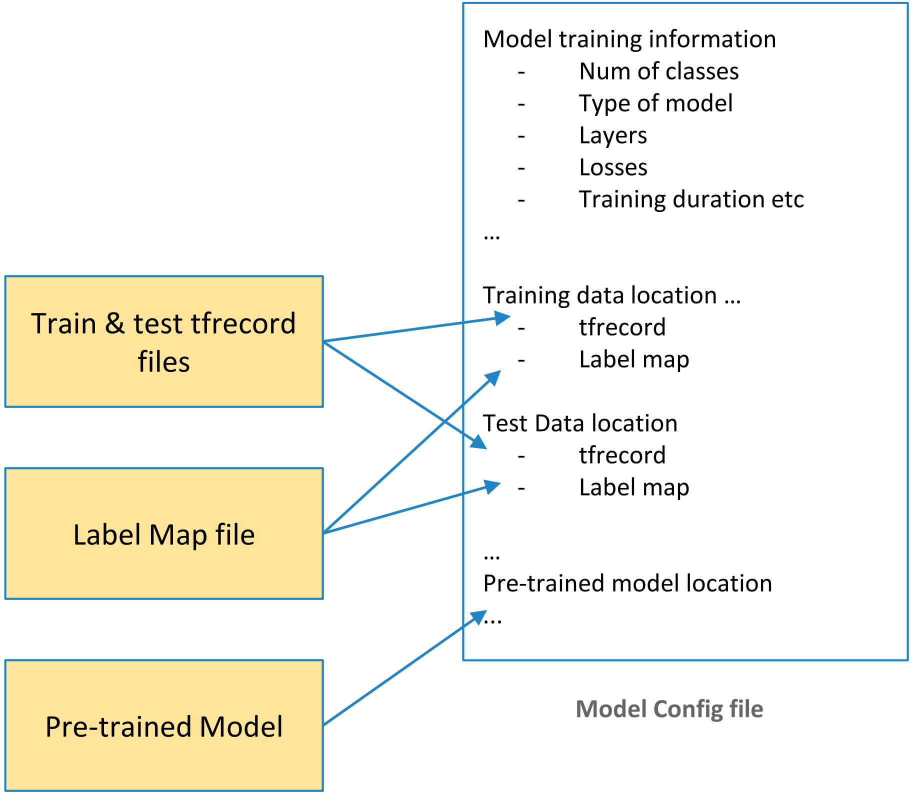
provides several pre-trained models for us to start training

https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md

3.2 Transfer Learning : Pre-trained Model

Model Training

SSD implementation using TF OD API



Model Config file

3.3 Model Training Configuration

What does Model config file contains?

SSD implementation using TF OD API

TensorFlow Object Detection API
provides model training configuration files which can be fine tuned for our model training

https://github.com/tensorflow/models/tree/master/research/object_detection/samples/configs

3.3 Model Training Configuration

Model Training

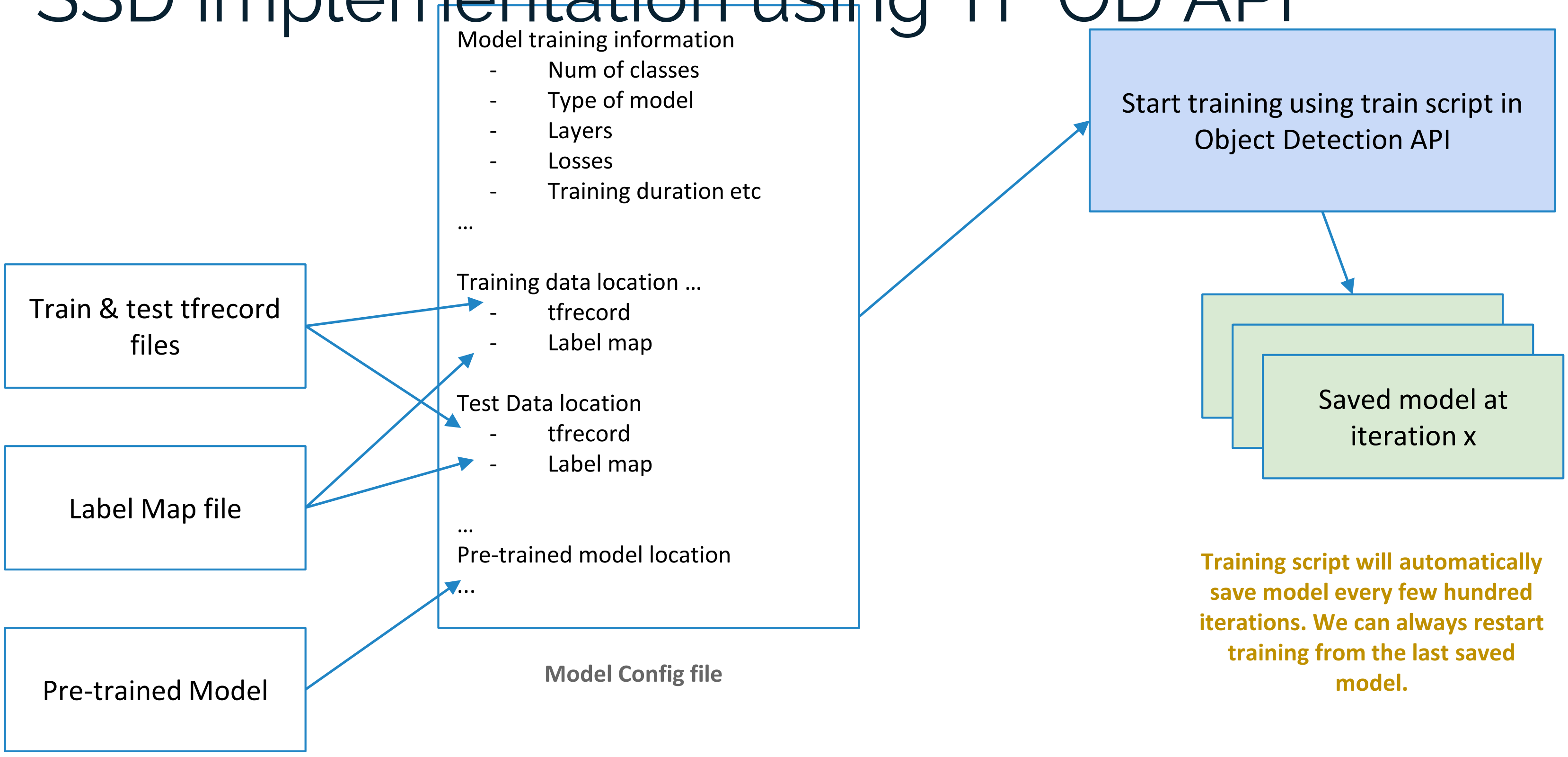
SSD implementation using TF OD API

1. Change **num_classes** parameter to number of classes in your dataset (e.g. 20 classes in pascal voc dataset)
2. For '**train_input_reader**' change '**input_path**' to filepath of train.record file.
3. For '**train_input_reader**' change '**label_map_path**' to filepath of pascal_voc.pbtxt file.
4. Repeat above two steps for '**eval_input_reader**'.
5. Change **fine_tune_checkpoint** to filepath where pre-trained model.ckpt file is available e.g ssd_mobilenet_v1_coco_2018_01_28/model.ckpt
6. Change '**batch_size**' accordingly to available memory.
7. Change '**num_steps**' to indicate how long the training will done e.g. 200000. Removing this parameter means that you can train indefinitely.

3.3 Model Training Configuration

Key things to change in Sample file

SSD implementation using TF OD API



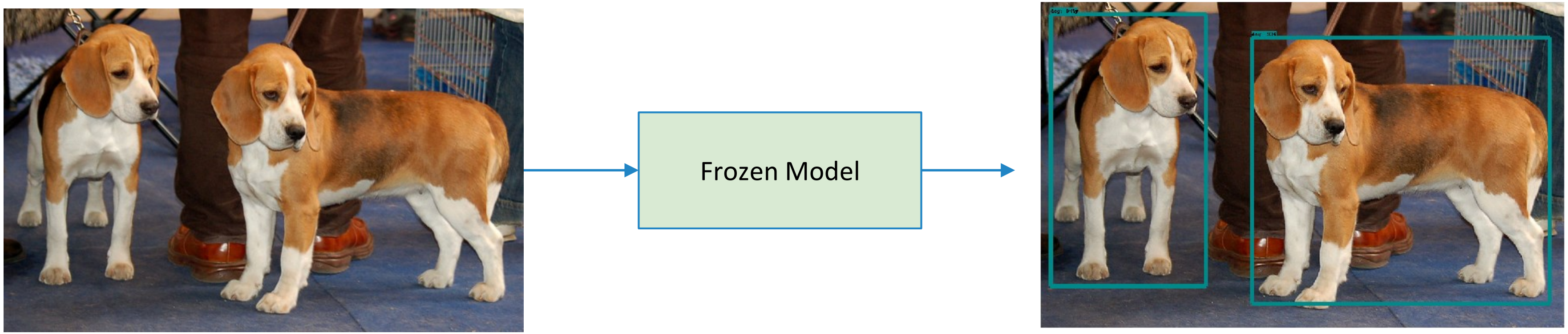
4. Model Training

SSD implementation using TF OD API

5. Freezing Trained Model

Remove nodes which are not used during prediction e.g Loss, Gradient Descent etc

SSD implementation using TF OD API



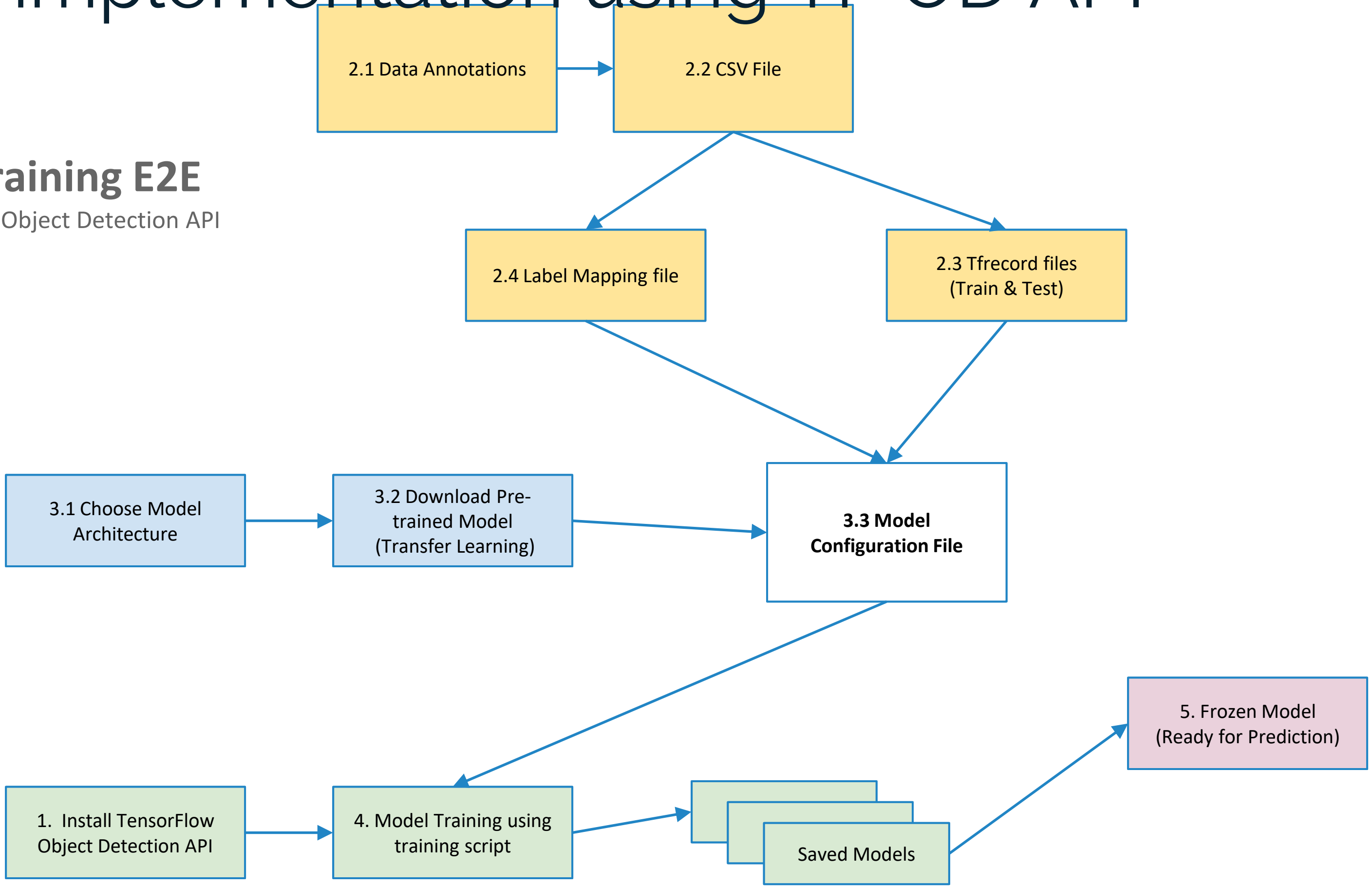
6. Model Prediction

Object Detection API makes a notebook available for model prediction which can be used with any model trained with TensorFlow object detection APIc

SSD implementation using TF OD API

Model Training E2E

Using TensorFlow Object Detection API



05

CONCLUSION

1. Quiz
2. Summary

Quiz

Question

Adding more default/anchor boxes in SSD can result in which of the following things?

A. improve accuracy
B. Decrease accuracy
C. Increase the speed
D. All of above

Quiz

Question

Adding more default/anchor boxes in SSD can result in which of the following things?

A. improve accuracy
B. Decrease accuracy
C. Increase the speed
D. All of above

Answer - A

Summary

In a nutshell you will understand

- Concept of SSD- single shot multibox detector
- Concept of object detection approach
- SSD 300 architecture
- Working of SSD which includes default boxes generation and boundary correction, classification and filtering (1)

Thank You

Happy Learning!

Blue Data Consulting
We make you trust data