

Crowd Management System



Table of content :

- Abstract
- Introduction
- Background
- Related Work
- Data collection
- Data augmentation
- Training
- Discussion
- Conclusion
-

Abstract

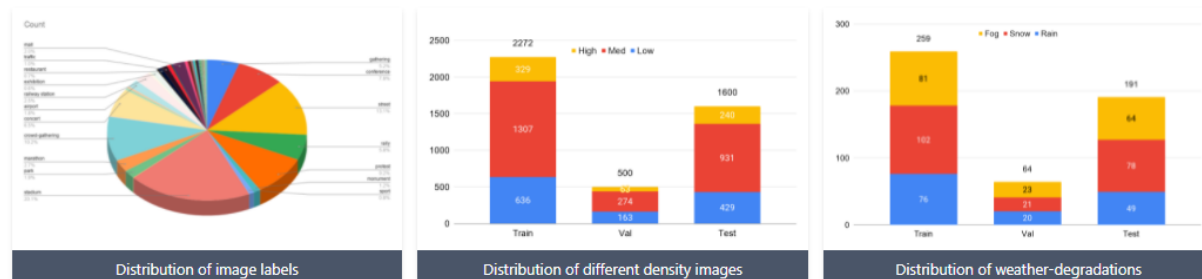
In this report, we will explain the mechanism for identifying and training the work environment, to identify the heads in a picture that contains a lot of people.

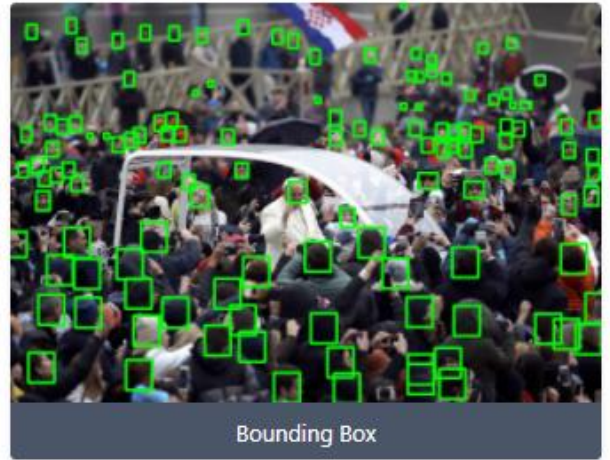
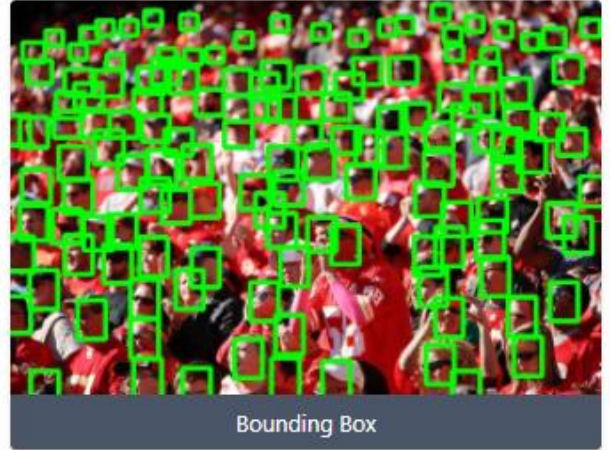
Introduction

Many cities and towns in various countries contain different activities that include large numbers of people. These events may be religious, sporting or social. For example, in our country, the Kingdom of Saudi Arabia, during the Hajj and Umrah seasons, a large number of pilgrims and Umrah performers from various Arab and foreign countries fall. Managing such crowds is a difficult and complex process for humans

CMS are systems concerned with studying the number, nature, behavior, safety and security...etc. Places crowded with people in large and huge events such as , World Cup events, or a concert by a famous artist, etc. These systems are useful in maintaining security and safety, such as determining the behavior of One of the crowd members with a different behavior from the rest so that the focus is on it. Also, moving corridors can be set up to manage the crowd if it is mobile by carrying out statistical operations after determining the number of people and their distribution and finding the best solution in their distribution. Monitors are faster and more accurate. These systems are very useful in improving the quality of services provided to crowds through data collected from the crowd over time. The main objective of these systems is, as we mentioned, improving services and helping to maintain safety and security for crowds. It is possible to know the number of people in a particular crowd by counting the heads so that each head expresses a specific person, it comes to your mind why you chose only the head and not the whole body, and the reason is that in large crowds the most You can see it as a user and a machine that is the head, so we took advantage of this feature in the crowds to count people and analyze their movements

Through these systems, we are able to obtain some statistics such as the number of people in a particular place and a certain hour, where this task is difficult for a person and easy for a machine, or the average number of daily visitors to a particular place, or the percentage of people staying in a particular place so that we know the important places for the largest number of people and the investment of





1. UCSD
2. MAL
3. UCF_CROWD
4. WorldExpo10
5. ShanghaiTech
6. UCF_QNRF
7. NWPU-CROWD

Dataset	Num of Images	Num of Annotations	Avg Count	Max Count	Avg H×W	Weather degradations	Distractors	Type of annotations
UCSD [8]	2,000	49,885	25	46	158×238	×	×	P
Mall [49]	2,000	62,325	-	53	320×240	×	×	P
UCF_CROWD_50 [3]	50	63,974	1,279	4,543	2101×2888	×	×	P
WorldExpo '10 [4]	3,980	199,923	50	253	576×720	×	×	P
ShanghaiTech [5]	1,198	330,165	275	3,139	598×868	×	×	P
UCF-QNRF [21]	1,535	1,251,642	815	12,865	2,013×2,902	×	×	P
NWPU-CROWD [29]	5,109	2,133,238	418	20,033	2,311×3,383	×	✓	P
JHU-CROWD (ours)	4,250	1,114,785	262	7,286	900×1,450	✓	✓	P, O, B, S, I
JHU-CROWD++ (ours)	4,372	1,515,005	346	25,791	910×1,430	✓	✓	P, O, B, S [†] , I

A. AllGoVision Video Analytics

1. System Description:

It is a real-time enterprise grade advanced video analytics software product developed by AllGo Systems, a leader in

Multimedia technology. Being a specialist in Video Analytics, it focuses on Accuracy and Reliable performance [8].

It is a desktop application, and it analyses the video contents in real-time and provides the crowd count with reasonably

high accuracy. Based on a pre-defined threshold value [8]. The crowding threshold can be set either based on the

number of people occupying the region of interest or the percentage of area occupied by crowd in the user specified

region, also supports crowd flow analysis by highlighting sections of the crowd by their direction of movement.

Counter flow of crowd, which happens in a direction opposite to the desired path, can also be intelligently detected and

alerted against [8].

2. System Services:

Video Analytics provided crowd management solution which includes Crowd Counting, Crowding Detection,

Overcrowding Alert, Crowd Flow Analysis, Crowd Counter Flow Detection [8].

B. iOmniscient and vedioIQ

1. System Description:

Omniscient offers the most comprehensive range of video analytics in the industry. From the most complex behaviour

analysis to the recognition of vehicles and humans, from simple video management to automated response systems, if

any analysis can be performed using Video, iOmniscient provides it. The iQ-120 is the Crowd Management, which is a

very smart system [9].

It gives a live count of the number of people within one or more regions of interest that have been drawn on the screen.

It is a desktop application and it combine crowd estimates from multiple cameras to gauge the total crowd size [9].

2. System Services:

It is will Counting in a Crowd, Crowd Density Analysis, Overcrowding (people), Traffic Congestion (vehicles) [9]. It is

can be used to monitor an area and alert staff if it becomes overcrowded and it will raise an alarm if the figure reaches a

pre-defined occupancy limit [9].

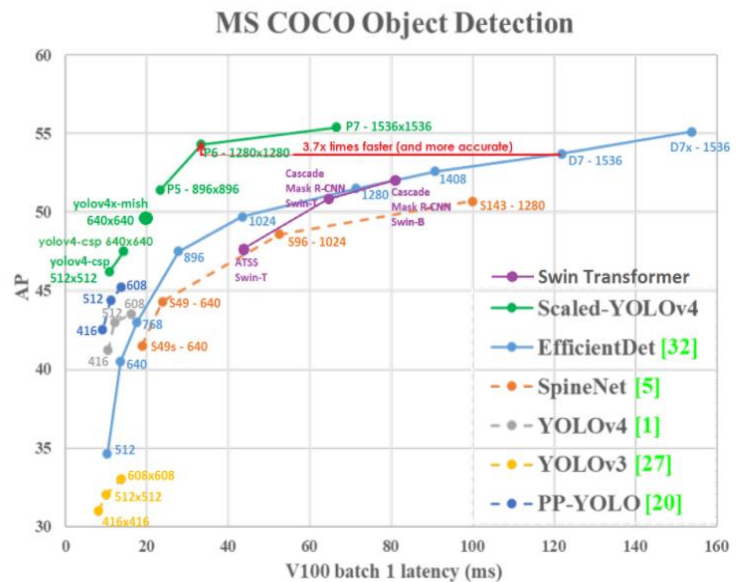
Application Name/ Feature	AllGo Vision	Iomiscient and vedioIQ	IPSOTEK	Qognify	VISIONGE NI
Desktop Application	✓	✓	✓	✓	✓
Crowd dictation /counting	✓	✓	✓	✓	✓
Crowd management	✓		✓	✓	✓
Overcrowding (People) alert	✓	✓	✓	✓	
Crowd density analysis		✓			
Traffic congestion		✓			
Crowd flow analysist	✓		✓		
Crowd counter flow dictation	✓			✓	

Background

Darknet :

Darknet: Open Source Neural Networks in C

Darknet is an open source neural network framework written in C and CUDA. It is fast, easy to install, and supports CPU and GPU computation. You can find the source on GitHub



CVAT :

Computer Vision Annotation Tool (CVAT) is a free, open source, web-based image and video annotation tool which is used for labeling data for computer vision algorithms. Originally developed by Intel, CVAT is designed for use by a professional data annotation team, with a user interface optimized for computer vision annotation tasks.

CVAT supports the primary tasks of supervised machine learning: object detection, image classification, and image segmentation. CVAT allows users to annotate data for each of these cases.

CVAT has many powerful features, including interpolation of shapes between key frames, semi-automatic annotation using deep learning models, shortcuts for most critical actions, a dashboard with a list of annotation projects and tasks, LDAP and basic access authentication, etc

CVAT is written mainly in TypeScript, React, Ant Design, CSS, Python, and Django. It is distributed under the MIT License, and its source code is available on [GitHub](https://github.com).

LabelMe

LabelMe is a project created by the MIT Computer Science and Artificial Intelligence Laboratory (CSAIL) which provides a dataset of digital images with annotations. The dataset is dynamic, free to use, and open to public contribution. The most applicable use of LabelMe is in computer vision research. As of October 31, 2010, LabelMe has 187,240 images, 62,197 annotated images, and 658,992 labeled objects.

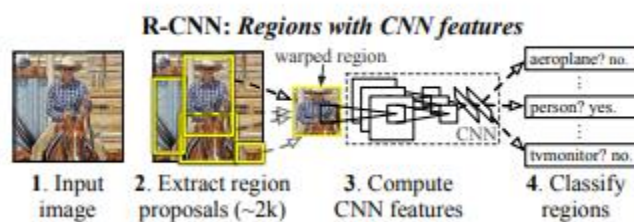


Related works:

RCNN:

The RCNN(1) takes an input

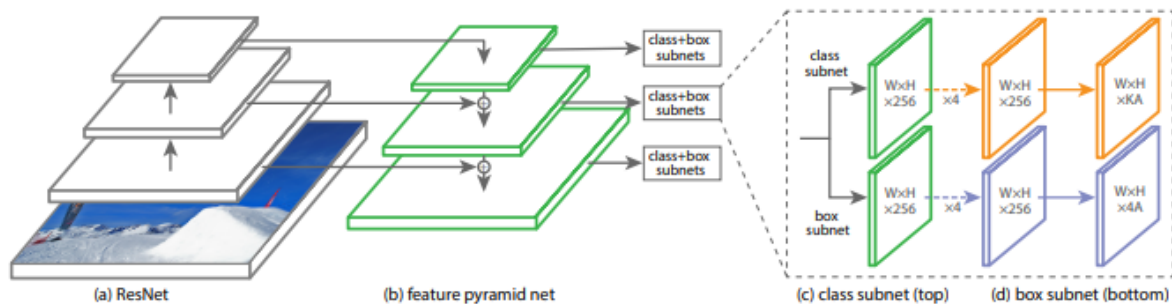
image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large convolutional neural network (CNN), and then (4) classifies each region using class-specific linear SVMs. R-CNN achieves a mean average precision (mAP) of 53.7% on PASCAL VOC 2010. For comparison, [39] reports 35.1% mAP using the same region proposals, but with a spatial pyramid and bag-of-visual-words approach. The popular deformable



part models perform at 33.4%. On the 200-class ILSVRC2013 detection dataset, R-CNN’s mAP is 31.4%, a large improvement over OverFeat [34], which had the previous best result at 24.3%.

RetinaNet

RetinaNet is a single, unified network composed of a backbone network and two task-specific subnetworks. The backbone is responsible for computing a convolutional feature map over an entire input image and is an off-the-shelf convolutional network. The first subnet performs convolutional object classification on the backbone’s output; the second subnet performs convolutional bounding box regression. The two subnetworks feature a simple design that we propose specifically for one-stage, dense detection, see Figure 3. While there are many possible choices for the details of these components, most design parameters are not particularly sensitive to exact values as shown in the experiments.



	backbone	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
RetinaNet (ours)	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet (ours)	ResNeXt-101-FPN	40.8	61.1	44.1	24.1	44.2	51.2

Yolo

YOLO is an algorithm that uses neural networks to provide real-time object detection. This algorithm is popular because of its speed and accuracy. It has been used in various applications to detect traffic signals, people, parking meters, and animals.

This article introduces readers to the YOLO algorithm for object detection and explains how it works. It also highlights some of its real-life applications.

What is YOLO?

YOLO is an abbreviation for the term 'You Only Look Once'. This is an algorithm that detects and recognizes various objects in a picture (in real-time). Object detection in YOLO is done as a regression problem and provides the class probabilities of the detected images.

YOLO algorithm employs convolutional neural networks (CNN) to detect objects in real-time. As the name suggests, the algorithm requires only a single forward propagation through a neural network to detect objects.

This means that prediction in the entire image is done in a single algorithm run. The CNN is used to predict various class probabilities and bounding boxes simultaneously.

The YOLO algorithm consists of various variants. Some of the common ones include tiny YOLO and YOLOv4 and YOLOv5

How the YOLO algorithm works

YOLO algorithm works using the following three techniques:

- Residual blocks
- Bounding box regression
- Intersection Over Union (IOU)

Residual blocks

First, the image is divided into various grids. Each grid has a dimension of $S \times S$. The following image shows how an input image is divided into grids.



In the image above, there are many grid cells of equal dimension. Every grid cell will detect objects that appear within them. For example, if an object center appears within a certain grid cell, then this cell will be responsible for detecting it.

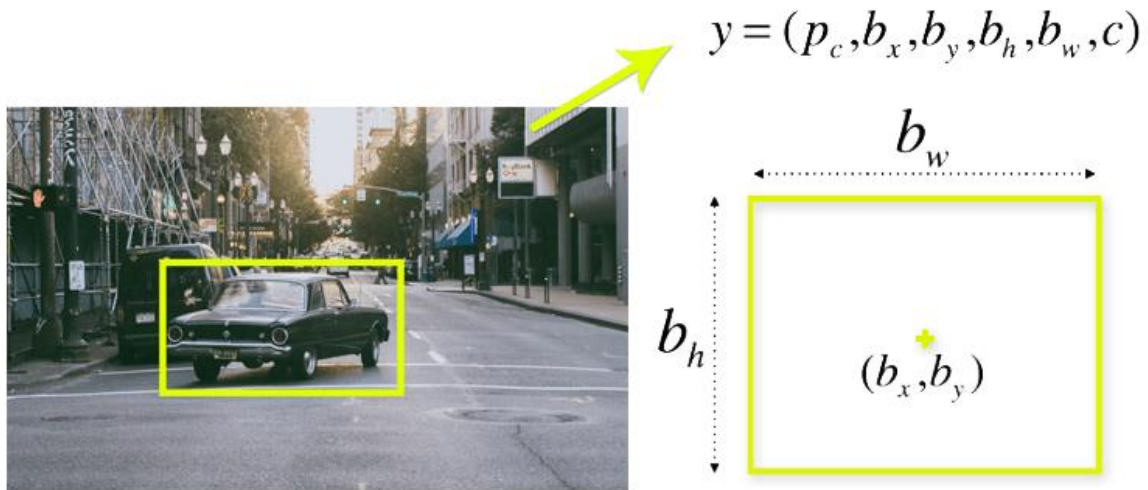
Bounding box regression

A bounding box is an outline that highlights an object in an image.

Every bounding box in the image consists of the following attributes:

- Width (bw)
- Height (bh)
- Class (for example, person, car, traffic light, etc.)- This is represented by the letter c.
- Bounding box center (bx,by)

The following image shows an example of a bounding box. The bounding box has been represented by a yellow outline.



YOLO uses a single bounding box regression to predict the height, width, center, and class of objects. In the image above, represents the probability of an object appearing in the bounding box.

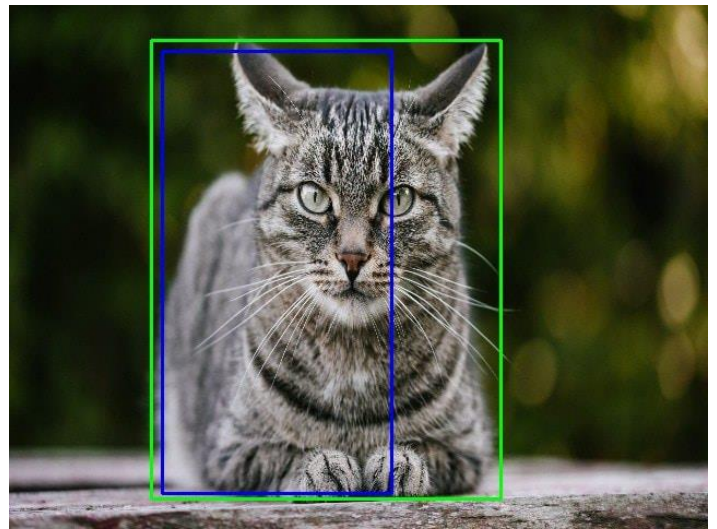
Intersection over union (IOU)

Intersection over union (IOU) is a phenomenon in object detection that describes how boxes overlap. YOLO uses IOU to provide an output box that surrounds the objects perfectly.

Each grid cell is responsible for predicting the bounding boxes and their confidence scores. The IOU is equal to 1 if the predicted bounding box is the same as the real box. This mechanism eliminates bounding boxes that are not equal to the real box.

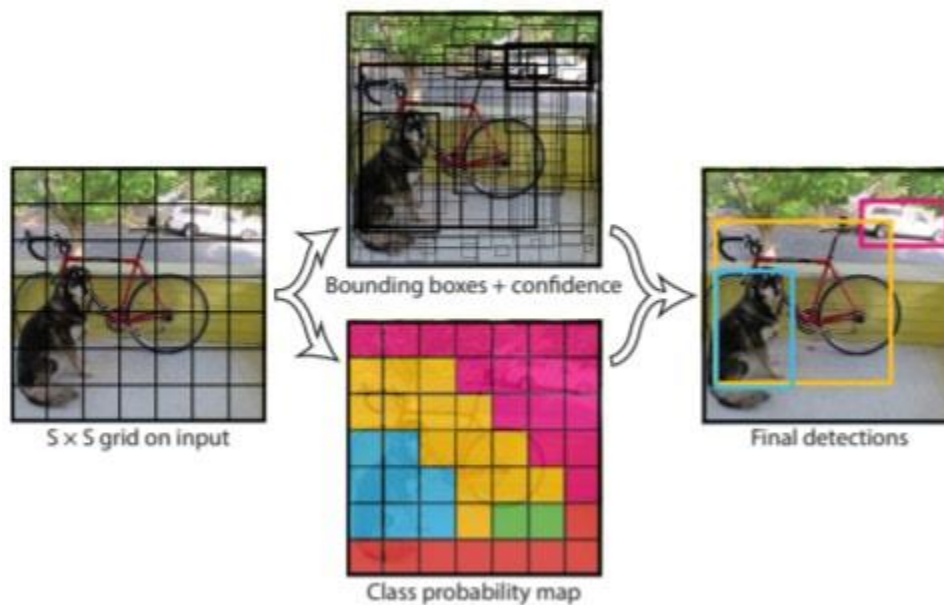
The following image provides a simple example of how IOU works.

In the image above, there are two bounding boxes, one in green and the other one in blue. The blue box is the predicted box while the green box is the real box. YOLO ensures that the two bounding boxes are equal.



Combination of the three techniques

The following image shows how the three techniques are applied to produce the final detection results.



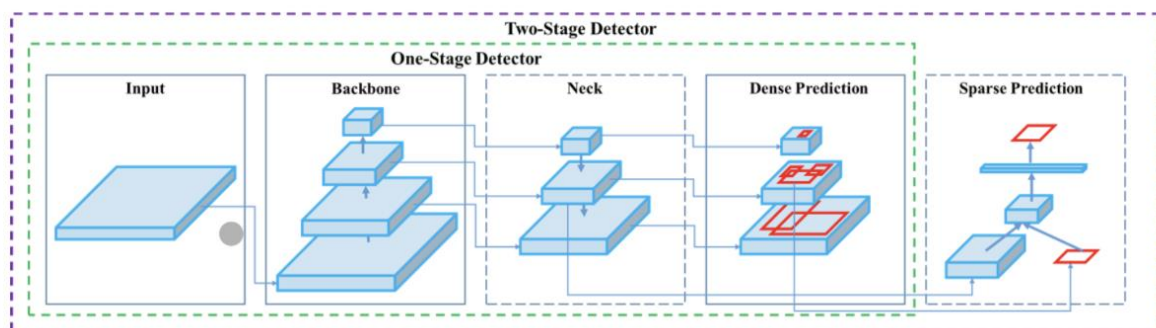
First, the image is divided into grid cells. Each grid cell forecasts B bounding boxes and provides their confidence scores. The cells predict the class probabilities to establish the class of each object.

For example, we can notice at least three classes of objects: a car, a dog, and a bicycle. All the predictions are made simultaneously using a single convolutional neural network.

Intersection over union ensures that the predicted bounding boxes are equal to the real boxes of the objects. This phenomenon eliminates unnecessary bounding boxes that do not meet the characteristics of the objects (like height and width). The final detection will consist of unique bounding boxes that fit the objects perfectly.

For example, the car is surrounded by the pink bounding box while the bicycle is surrounded by the yellow bounding box. The dog has been highlighted using the blue bounding box.

Architecture of an object detection model



Backbone

The backbone network for an object detector is typically pretrained on ImageNet classification. Pretraining means that the network's weights have already been adapted

to identify relevant features in an image, though they will be tweaked in the new task of object detection.

The authors considered the following backbones for the YOLOv4 object detector

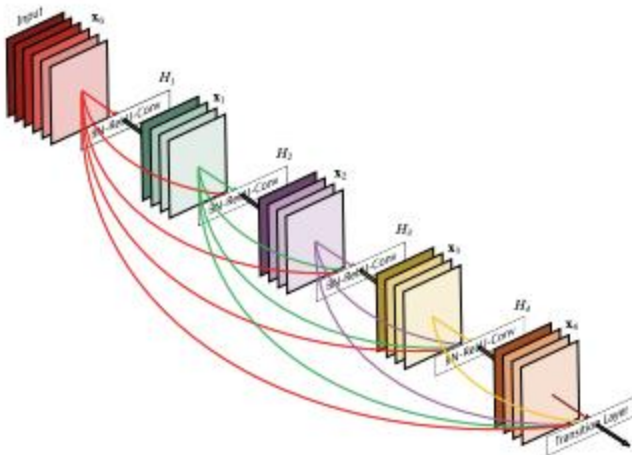
- CSPResNext50
- CSPDarknet53
- EfficientNet-B3

Table 1: Parameters of neural networks for image classification.

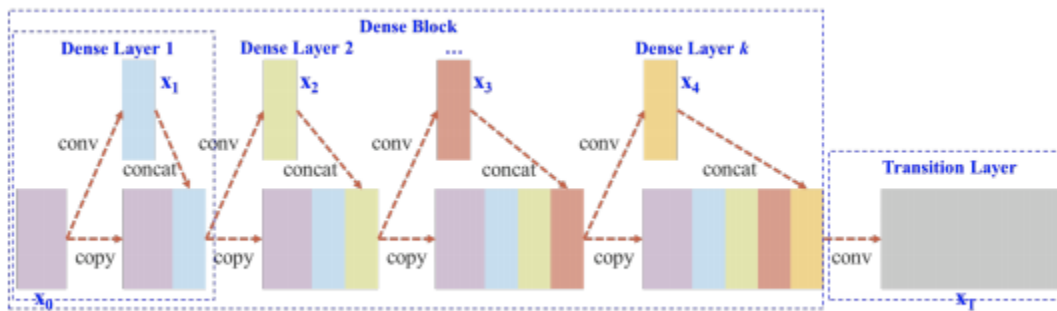
Backbone model	Input network resolution	Receptive field size	Parameters	Average size of layer output (WxHxC)	BFLOPs (512x512 network resolution)	FPS (GPU RTX 2070)
CSPResNext50	512x512	425x425	20.6 M	1058 K	31 (15.5 FMA)	62
CSPDarknet53	512x512	725x725	27.6 M	950 K	52 (26.0 FMA)	66
EfficientNet-B3 (ours)	512x512	1311x1311	12.0 M	668 K	11 (5.5 FMA)	26

([citation](#))

The CSPResNext50 and the CSPDarknet53 are both based on DenseNet. DenseNet was designed to connect layers in convolutional neural networks with the following motivations: to alleviate the vanishing gradient problem (it is hard to backprop loss signals through a very deep network), to bolster feature propagation, encourage the network to reuse features, and reduce the number of network parameters.



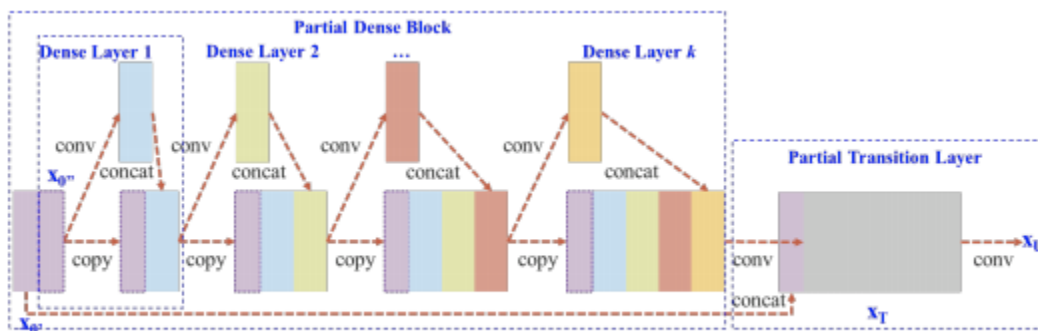
(citation)



(a) DenseNet

(citation)

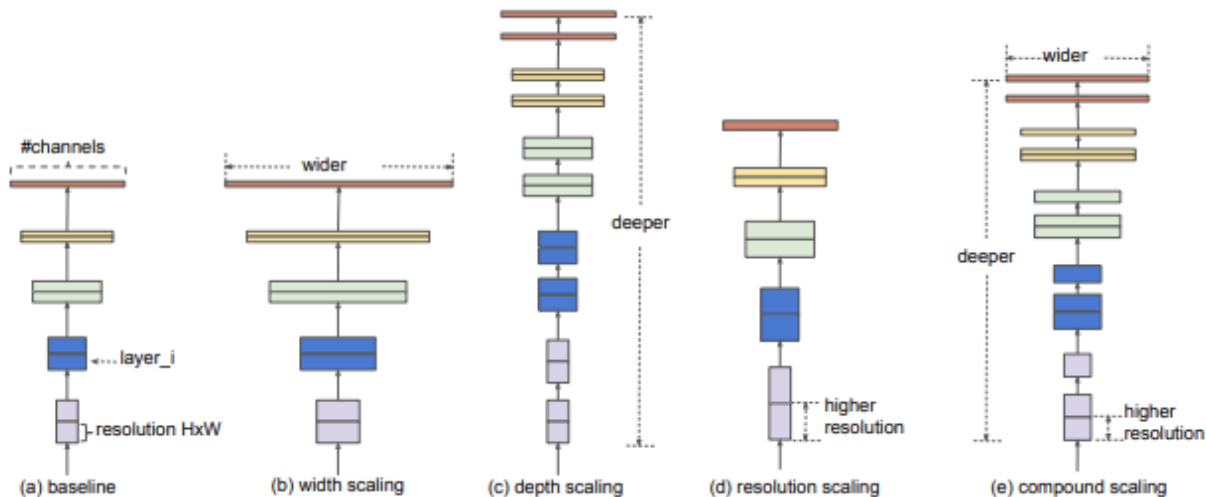
In CSPResNext50 and CSPDarknet53, the DenseNet has been edited to separate the feature map of the base layer by copying it and sending one copy through the dense block and sending another straight on to the next stage. The idea with the CSPResNext50 and CSPDarknet53 is to remove computational bottlenecks in the DenseNet and improve learning by passing on an unedited version of the feature map.



(b) Cross Stage Partial DenseNet

(citation)

EfficientNet was designed by Google Brain to primarily study the scaling problem of convolutional neural networks. There are a lot of decisions you can make when scaling up your ConvNet including input size, width scaling, depth scaling, and scaling all of the above. The EfficientNet paper posits that there is an optimal point for all of these and through search, they find it.



([citation](#))

EfficientNet outperforms the other networks of comparable size on image classification. The YOLOv4 authors posit, however, that the other networks may work better in the object detection setting and decide to experiment with all of them.

Based on their intuition and experimental results (aka A LOT of experimental results), the final YOLOv4 network implements CSPDarknet53 for the backbone network.

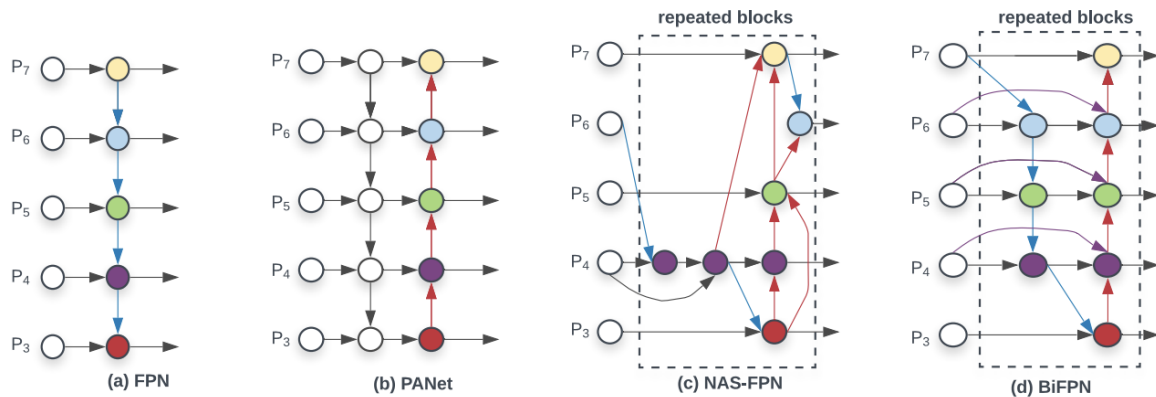
YOLOv4 Neck - Feature Aggregation

The next step in object detection is to mix and combine the features formed in the ConvNet backbone to prepare for the detection step. YOLOv4 considers a few options for the neck including:

- FPN
- PAN

- NAS-FPN
- BiFPN
- ASFF
- SFAM

The components of the neck typically flow up and down among layers and connect only the few layers at the end of the convolutional network.



([cite](#))

Each one of the P_i above represents a feature layer in the CSPDarknet53 backbone.

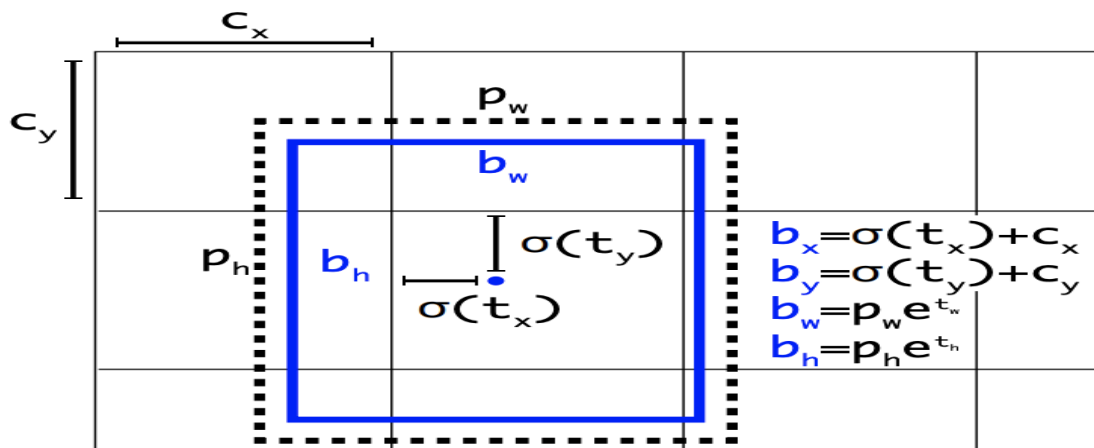
The image above comes from YOLOv4's predecessor, EfficientDet. Written by Google Brain, EfficientDet uses neural architecture search to find the best form of blocks in the neck portion of the network, arriving at NAS-FPN. The EfficientDet authors then tweak it slightly to make the more architecture more intuitive (and probably perform better on their development sets).

YOLOv4 chooses PANet for the feature aggregation of the network. They don't write much on the rationale for this decision, and since NAS-FPN and BiFPN were written concurrently, this is presumably an area of future research.

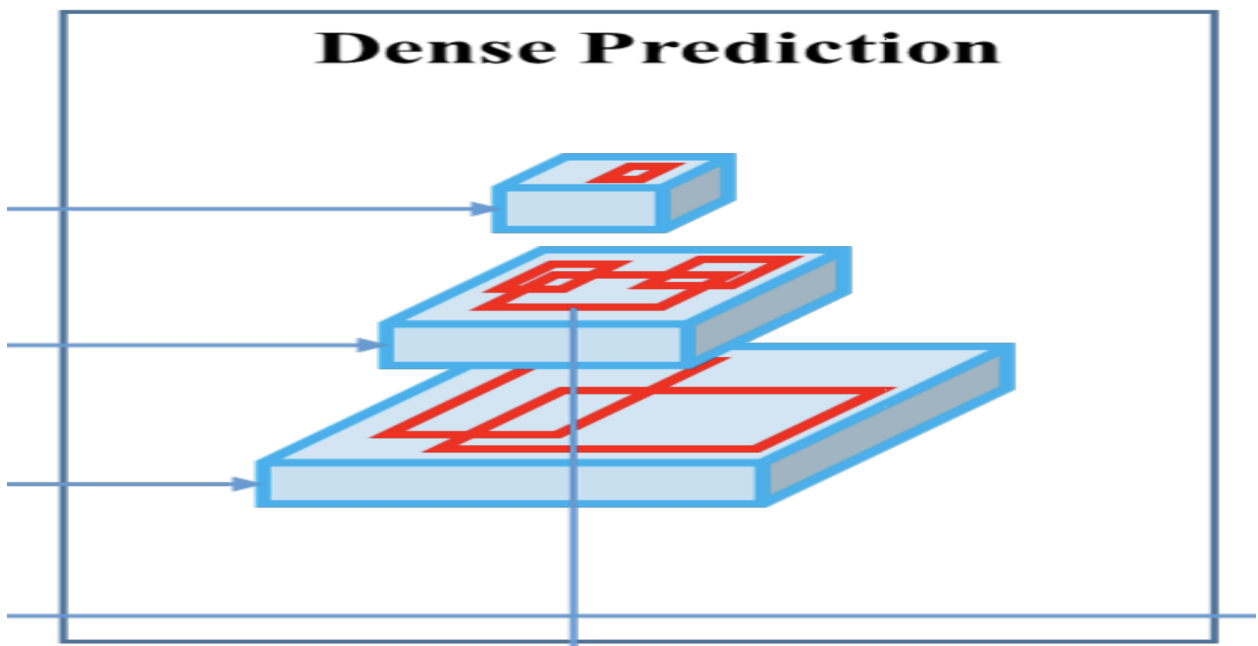
Additionally, YOLOv4 adds a SPP block after CSPDarknet53 to increase the receptive field and separate out the most important features from the backbone.

YOLOv4 Head - The Detection Step

YOLOv4 deploys the same YOLO head as YOLOv3 for detection with the anchor based detection steps, and three levels of detection granularity.



(citation)(Citation)



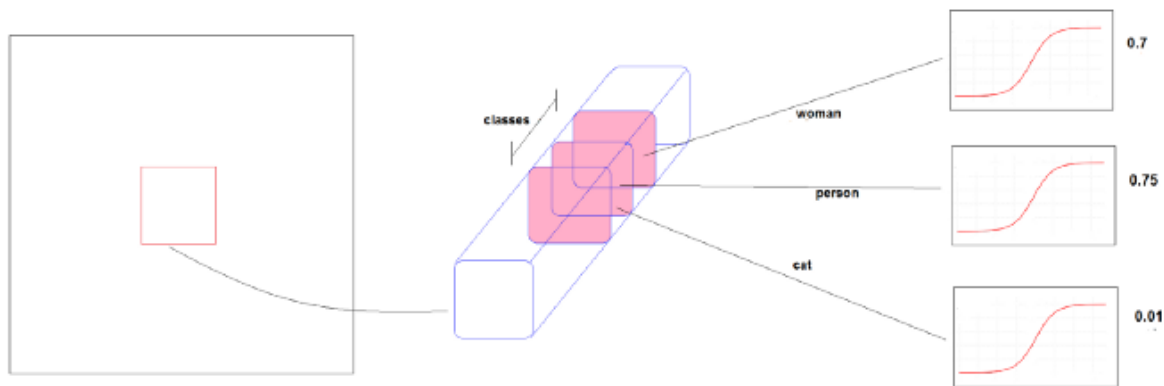
Bounding box prediction:

- Same as YOLO9000, our system predicts bounding boxes using dimension clusters as anchor boxes. The network predicts 4 coordinates for each bounding box, t_x , t_y , t_w , t_h . If the cell is offset from the top left corner of the image by (c_x, c_y) and the bounding box prior has width and height p_w , p_h , then the predictions correspond to
- $b_x = \sigma(t_x) + c_x$, $b_y = \sigma(t_y) + c_y$, $b_w = p_w e^{t_w}$, $b_h = p_h e^{t_h}$ where e is exp
- The above equation can be inverted to compute the ground truth values.
- YOLOv3 predicts an objectness score for each bounding box using logistic regression. This should be 1 if the bounding box prior overlaps a ground truth object by more than any other bounding box prior. If the bounding box prior is not the best but does overlap a ground truth object by more than some threshold we ignore prediction. Our system only assigns one bounding box prior for each ground truth object. if bounding box prior is not assigned to a ground truth object it incurs no loss for coordinate or class predictions, only objectness.

Class prediction:

In some datasets like Open Images Dataset, there are many overlapping labels such as woman and person. Using a softmax for class prediction imposes the assumption that each box has exactly one class which is often not the case.

For this reason, YOLOv3 does not use the softmax; it simply uses the independent logistic classifiers for any class. Binary cross-entropy is being used during training for the class predictions.

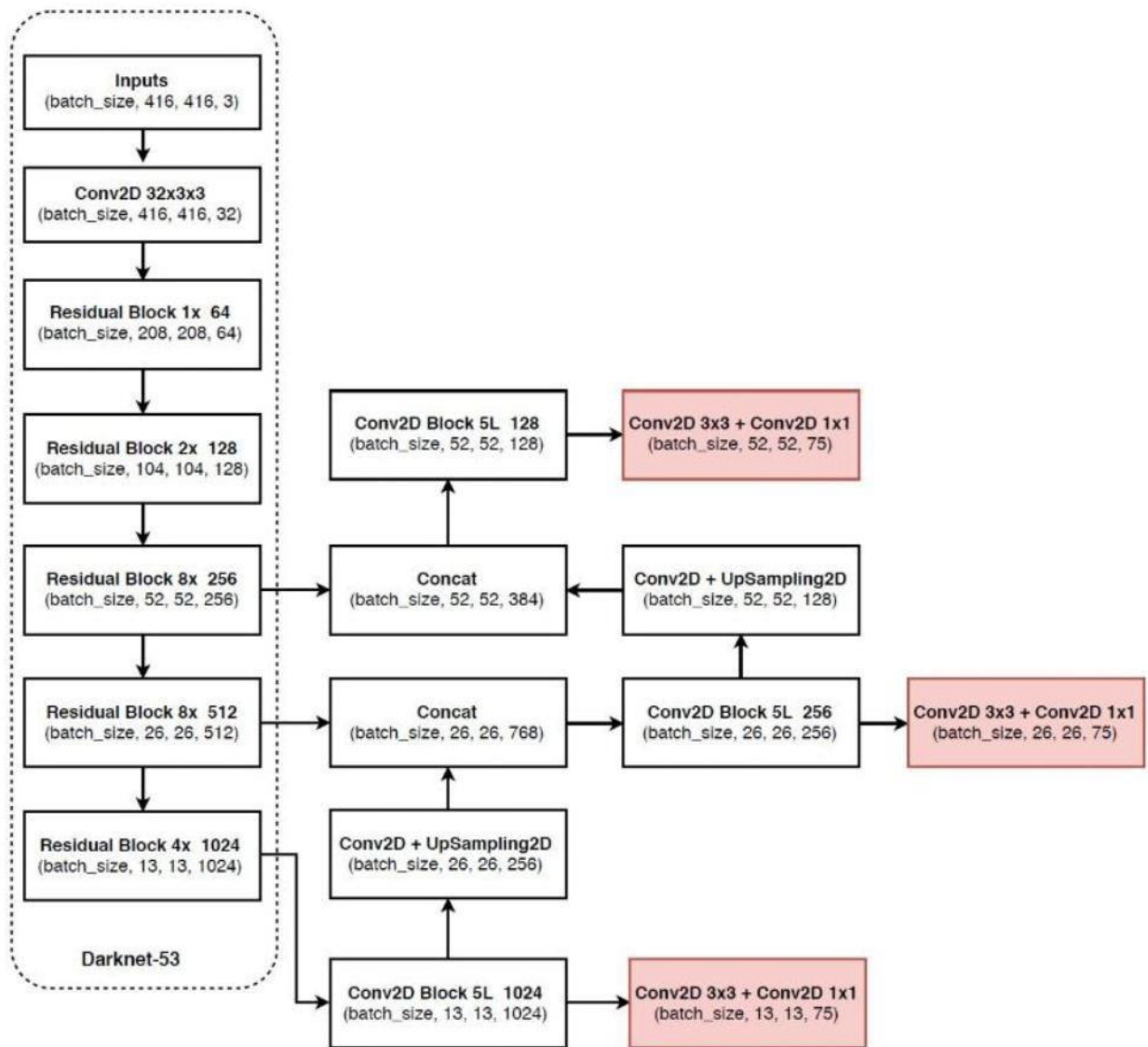


Predictions Across Scales:

- Yolo-v4 makes predictions across 3 different scales. The detection layer is used to make detection at feature maps of three different sizes, having strides 32, 16, 8 respectively. This means that with an input of 416x416, we make detections on scales 13x13, 26x26, and 52x52.
- The network down samples the input image until the first detection layer, where a detection is made using feature maps of a layer with stride 32. Further, layers are upsampled by a factor of 2 and concatenated with feature maps of a previous layers having identical feature map sizes. Another detection is now made at layer with stride 16. The same upsampling procedure is repeated, and a final detection is made at the layer of stride 8.
- At each scale, each cell predicts 3 bounding boxes using 3 anchors, making the total number of anchors used 9. The anchors are different for different scales.
- Prediction across scales helps Yolo-v4 get better at detecting small objects. Upsampling can help the network to learn fine-grained features which are instrumental for detecting small objects.
- Yolo-v4 produces a tensor of size 10647 which is equal to $((52 \times 52) + (26 \times 26) + (13 \times 13)) \times 3$

Feature Extractor:

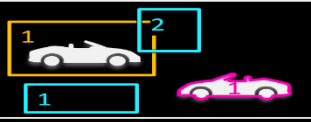

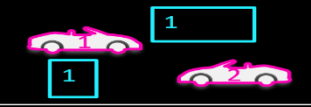
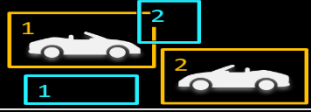

The below diagram shows the details of feature extractor which is Darknet and three different scaled feature vectors output.



Best bounded box selected using this method:

Definitions

- FP - False Positives: "Ghosts", number of issued predictions that were incorrect.
- FN - False Negatives: "Misses", number of targets that should have been predicted, but were missed.
- TP - True Positives: "Hits", correct predictions
- TN - True Negatives: correct prediction of the absence of a class; not useful for detections tasks as in this context, an infinite amount of TNs exists.
- Precision: ~"Prediction Reliability", share of the issued predictions that were correct.
- Recall: "Hit Rate", share of the targets that were predicted correctly.
- F1 Score: The harmonic mean between Precision and Recall, hence a metric reflecting both perspectives.

		Precision (Pr) $\frac{TP}{TP + FP}$	Recall (Rc) $\frac{TP}{TP + FN}$	F1-Score $\frac{2 \cdot Pr \cdot Rc}{Pr + Rc}$
A		$\frac{1}{1 + 2} = 33\%$	$\frac{1}{1 + 1} = 50\%$	$\frac{2 \cdot 33\% \cdot 50\%}{33\% + 50\%} = 40\%$
B		$\frac{2}{2 + 0} = 100\%$	$\frac{2}{2 + 0} = 100\%$	$\frac{2 \cdot 100\% \cdot 100\%}{100\% + 100\%} = 100\%$
C		$\frac{0}{0 + 2} = 0\%$	$\frac{0}{0 + 2} = 0\%$	$\frac{2 \cdot 0\% \cdot 0\%}{0\% + 0\%} = 0\%$
D		$\frac{2}{2 + 2} = 50\%$	$\frac{2}{2 + 0} = 100\%$	$\frac{2 \cdot 50\% \cdot 100\%}{50\% + 100\%} = 67\%$
E		$\frac{1}{1 + 0} = 100\%$	$\frac{1}{1 + 1} = 50\%$	$\frac{2 \cdot 100\% \cdot 50\%}{100\% + 50\%} = 67\%$

The chart above shows Precision and Recall values for various scenarios to illustrate their respective characteristics.

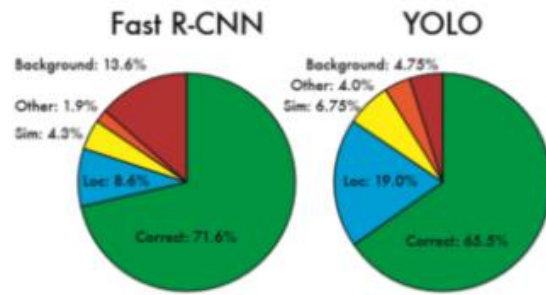
- Scenario A is a "Mixed Bag": Since it contains TPs, FNs and FPs, both Precision and Recall are "somewhere between" 0 and 100% and the F1 score provides a value between them (note how it differs from arithmetic mean) - overall a standard case.
- Scenario B is an "Epic Win": There are only TPs, hence Precision and Recall are at 100%
- Scenario C is a "Catastrophic Failure": There are no TPs, hence Precision and Recall are zero.
- Scenario D is a "Ghost Town": There are TPs and FPs but no FNs, hence Precision is low while recall is high.
- Scenario E is "Prediction Scarcity": There are TPs and FNs but no FPs, hence Precision is high and Recall is low.

Scenario D and E nicely illustrate that

- Precision is a measure for efficiency - if few predictions are wasted, Precision is high, even if that means that only few targets are hit. It favors a targeted approach and may drive a model into frugality - better no predictions than a wrong ones.
- Recall is a measure for effectiveness - the more targets are hit, the better, independent of the amount of wrong predictions made. It favors a spamming approach and may drive a model into prodigality - issuing a flood of predictions in the hope that some may hit the targets.

Comparison between Fast r-RCNN and YOLO algorithm

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	155
YOLO	2007+2012	63.4	45
<hr/>			
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21



Real-Time Systems on PASCAL VOC 2007 .We can notice that YOLO struggles to localize objects correctly,

Data collection

There are many images available on the internet and they have been mentioned previously. We also manually collected images from YouTube, where we downloaded many video clips and converted them to formats using the opencv library with the Python programming language. We also placed annotation on the images manually through custom programs such as CVAT And Labelme, this process takes a lot of time and effort depending on the number of people in one picture, where the number of people ranges from 3 to more than 300 people in one picture, and this is a hard work for us. We did it manually

Data augmentation

The main objective of data augmentation methods is to increase the variability of an image in order to improve the generalization of the model training.

The most commonly used methods are Photometric Distortion, Geometric Distortion, MixUp, CutMix and GANs.

Photometric distortion

Photometric distortion creates new images by adjusting brightness, hue, contrast, saturation and noise to display more varieties of the same image.



figure 3: adjust hue of an image

In the example above we adjusted the Hue(or color appearance parameter) to modify the image and create new samples to create more variability in our training set.

Geometric distorsion

The so-called geometric distortion methods are all the techniques used to rotate the image, flipping, random scaling or cropping.



figure 4: rotation

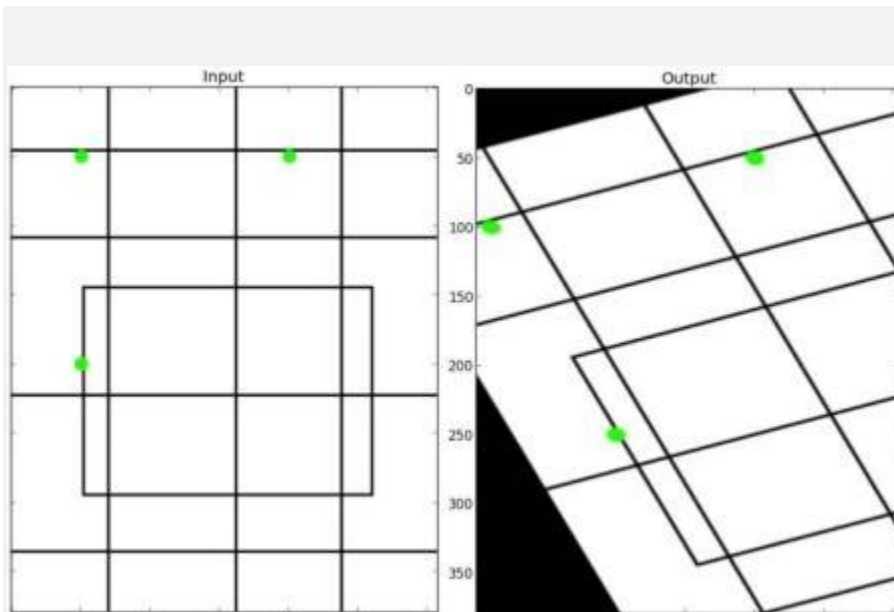


figure 5: affine transformation

In the first example (i.e. figure 4), we rotated the original image by 90° . In the second example (i.e. figure 5), we performed an affine transformation of the original image.

MixUp

Mixup augmentation is a type of augmentation where in we form a new image through weighted linear interpolation of two existing images. We take two images and do a linear combination of them in terms of tensors of those images. Mixup reduces the memorization of corrupt labels, increases the robustness to adversarial examples, and stabilizes the training of generative adversarial networks.

In mixup, two images are mixed with weights: λ and $1-\lambda$. λ is generated from symmetric beta distribution with parameter alpha. This creates new virtual training samples.

In image classification images and labels can be mixed up as following:

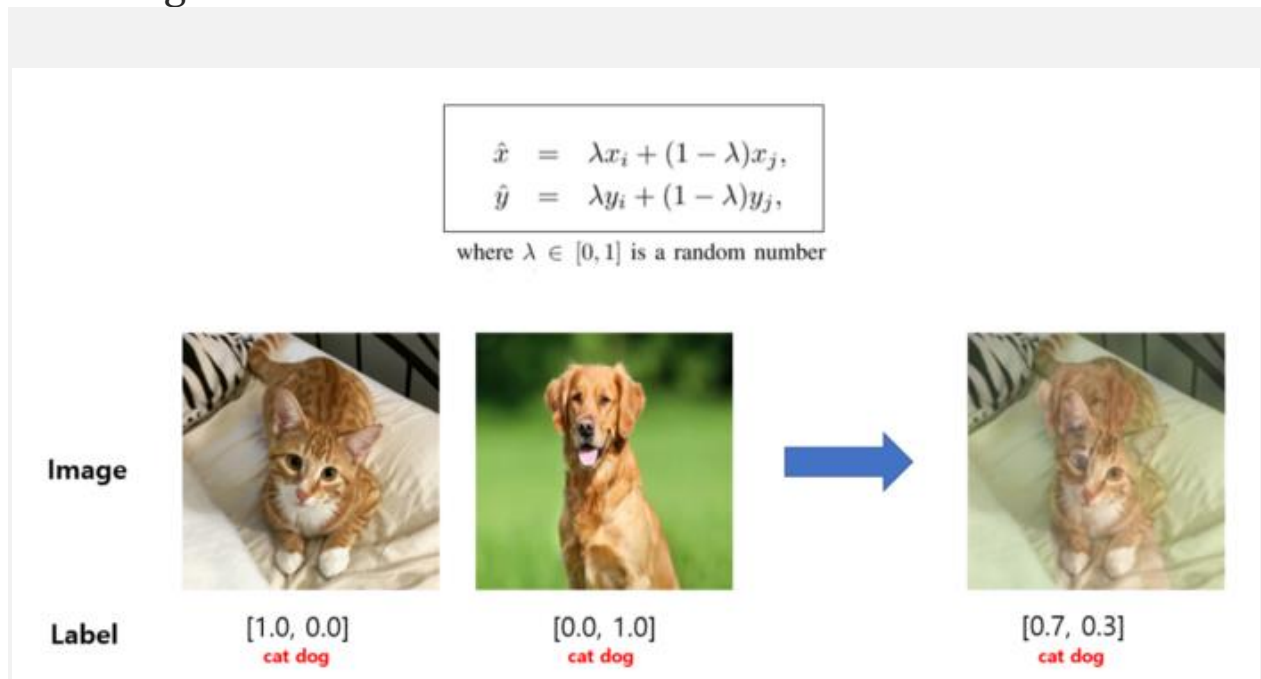
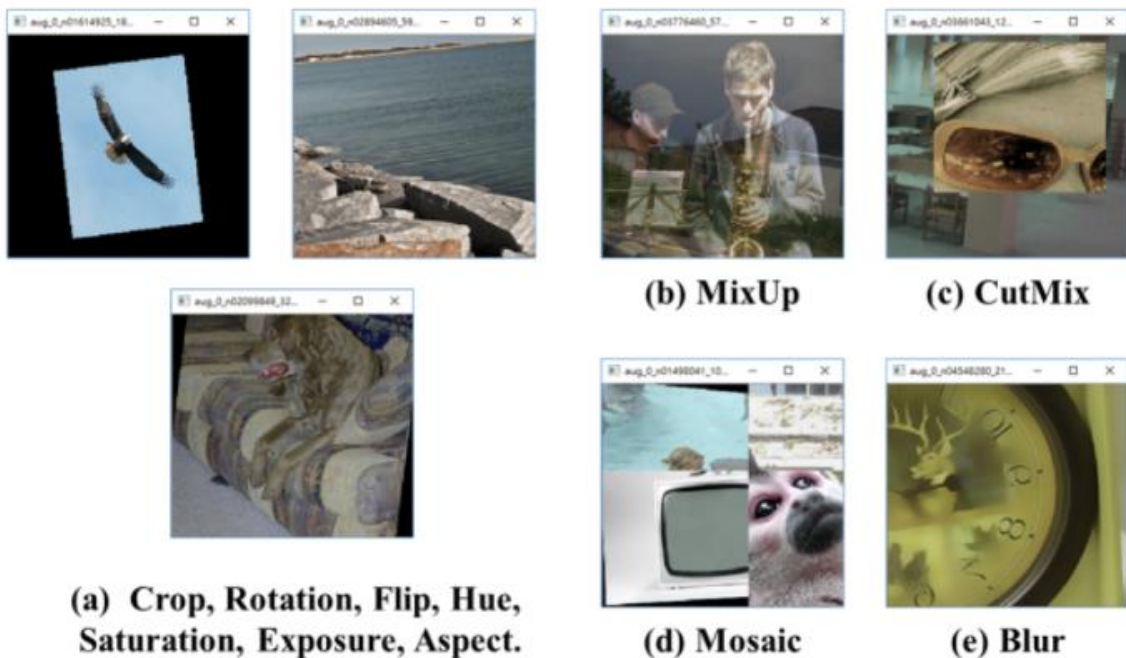


figure 6: MixUp

CutMix

CutMix augmentation strategy: patches are cut and pasted among training images where the ground truth labels are also mixed proportionally to the area of the patches. CutMix improves the model robustness against input corruptions and its out-of-distribution detection performances.





Training the darknet to detection head

The process that follows the process of collecting images is training the work environment on images. The learning process takes time, not a little. The training time varies according to many parameters that we can change. These parameters are considered hyperparameters, as the user is the one who enters them and adjusts them from one training to another and checking the accuracy of the model from Through dedicated methods or through a very complex picture, we will review some of these parameters and explain them:

- Width
 - height
 - batch
 - subdivisions
 - decay
 - momentum
 - channels
 - filters
 - activation
-
- width : width of the image in input layer
 - Height : height of the image in input layer
 - batch: That many images+labels are used in the forward pass to compute a gradient and update the weights via backpropagation.
 - subdivisions: The batch is subdivided in this many "blocks". The images of a block are ran in parallel on the gpu.
 - decay: Maybe a term to diminish the weights to avoid having large values. For stability reasons .
 - channels:
 - momentum: I guess the new gradient is computed by $momentum * previous_gradient + (1-momentum) * gradient_of_current_batch$. Makes the gradient more stable.
 - adam: Uses the adam optimizer? Doesn't work for me though

- `burn_in`: For the first x batches, slowly increase the learning rate until its final value (your *learning_rate* parameter value). Use this to decide on a learning rate by monitoring until what value the loss decreases (before it starts to diverge).
 - `policy=steps`: Use the steps and scales parameters below to adjust the learning rate during training
 - `steps=500,1000`: Adjust the learning rate after 500 and 1000 batches
 - `scales=0.1,0.2`: After 500, multiply the LR by 0.1, then after 1000 multiply again by 0.2
 - `angle`: augment image by rotation up to this angle (in degree)
-
- `filters`: How many convolutional kernels there are in a layer.
 - `activation`: Activation function, relu, leaky relu, etc. See `src/activations.h`
 - `stopbackward`: Do backpropagation until this layer only. Put it in the panultimate convolution layer before the first yolo layer to train only the layers behind that, e.g. when using pretrained weights.
 - `random`: Put in the yolo layers. If set to 1 do data augmentation by resizing the images to different sizes every few batches. Use to generalize over object sizes.

`batch=64`

`[net]`

```
batch = 64
subdivisions=32
width=608
height=608
channels=3
momentum=0.9
decay=0.0005
```

```
batch=64
subdivisions=16
width=416
height=416
channels=1
momentum=0.9
```

```
angle=0
saturation = 1.5
exposure = 1.5
hue=.1
```

```
learning_rate=0.001
```

```
max_batches = 2000
policy=steps
steps=-
1,100,80000,100000
scales=.1,10,.1,.1
```

```
decay=0.0005
```

```
angle=0
saturation = 1.5
exposure = 1.5
hue=.1
```

```
learning_rate=0.001
```

```
max_batches = 6000
policy=steps
steps=-1,100,80000,100000
scales=.1,10,.1,.1
```


All this parameter learning from experiments and from

During the training process, there is an adjustment to the network weights so that the training summary is a model that identifies the people. This model includes the network weights only. The general structure of the network is saved in a file with the cfg suffix, where this file includes many parameters after each training era. The environment By calculating avgLoss, so that this value is initially maximum, and through the training process, this value gradually decreases. This value can increase during training and decrease as well, and this is normal, as the model trained on a certain pattern of images and received a different image, so he made a lot of mistakes, and this raises this value one of the best mechanisms What is used during training is to change the size of the image during training so that I ensure that it does not fall into the case of the overFit, i.e. the case of memorization only

Some types of yolo change the network structure during training, and this is a very good mechanism. We previously explained the training mechanism in detail and how to predict the box and choose the best surrounding box for the purpose. As for the values that I chose, they refer to experimentation in addition to some

recommendations from the main developer of the environment. Below are some pictures that include the training process during different eras and a review of the different results

```
+ Code + Text

 (next mAP calculation at 3560 iterations)
Last accuracy mAP@0.5 = 25.10 %, best = 25.10 %
... 3268: 17.012991, 13.425767 avg loss, 0.001000 rate, 5.581487 seconds, 156864 images, 4.466440 hours left
Loaded: 0.000046 seconds

(next mAP calculation at 3560 iterations)
Last accuracy mAP@0.5 = 25.10 %, best = 25.10 %
3269: 9.532199, 13.036410 avg loss, 0.001000 rate, 5.097796 seconds, 156912 images, 4.448629 hours left
Loaded: 0.000044 seconds

Wrong annotation: x = 0, y = 0, < 0 or > 1, file: data/obj/D04124.txt

(next mAP calculation at 3560 iterations)
Last accuracy mAP@0.5 = 25.10 %, best = 25.10 %
3270: 12.985675, 13.031337 avg loss, 0.001000 rate, 5.523713 seconds, 156960 images, 4.428655 hours left
Resizing, random_coef = 1.40
```

Result:

Testing on image

(predictions:21652): Gtk-WARNING **: 21:07:03.291: cannot open display:





(predictions:1427): Gtk-WARNING **: 20:03:37.576: cannot open display:



Map result in different cases:

```
[95] Done! Loaded 162 layers from weights-file
```

```
calculation mAP (mean average precision)...
2376
detections_count = 792729, unique_truth_count = 198297
class_id = 0, name = h, ap = 25.10%      (TP = 47125, FP = 30523)

for conf_thresh = 0.25, precision = 0.61, recall = 0.24, F1-score = 0.34
for conf_thresh = 0.25, TP = 47125, FP = 30523, FN = 151172, average IoU = 43.21 %

IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
```

```
Done! Loaded 162 layers from weights-file
```

```
calculation mAP (mean average precision)...
2376
detections_count = 655757, unique_truth_count = 198297
class_id = 0, name = h, ap = 25.20%      (TP = 42207, FP = 21994)

for conf_thresh = 0.25, precision = 0.66, recall = 0.21, F1-score = 0.32
for conf_thresh = 0.25, TP = 42207, FP = 21994, FN = 156090, average IoU = 48.95 %

IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
mean average precision (mAP@0.50) = 0.252014, or 25.20 %
Total Detection Time: 272 Seconds
```

```
Set -points flag:
```

```
`-points 101` for MS COCO
`-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data)
`-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset
```

```
calculation mAP (mean average precision)...
2376
detections_count = 792729, unique_truth_count = 198297
class_id = 0, name = h, ap = 25.10%      (TP = 47125, FP = 30523)

for conf_thresh = 0.25, precision = 0.61, recall = 0.24, F1-score = 0.34
for conf_thresh = 0.25, TP = 47125, FP = 30523, FN = 151172, average IoU = 43.21 %

IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
mean average precision (mAP@0.50) = 0.250986, or 25.10 %
```


Problem before and in training time :

1 .data collection

The difficulty of collecting data lies in two parts, the first is the images and the second is the annotation, as there is no general dataset published in this field, and from it we have collected images from videos filmed by people on the YouTube platform and we have marked them as this process takes a long time and varies according to the image And the number of people in it

2 .data Annotating

The difficulty here is where the number of faces is many, and this requires a long time. Sometimes one image takes about an hour, containing 1400 box

3 .normalize annotation to be between 0 and 1 in HUI-CROWD++

This Annotation data contains many values of interest, including the first 4 values, which are x, y, w, h

But these values are in the natural values, i.e. confined between 0 and the dimensions of the image, but the darknet formula does not accept these values, as it must be between 0 and 1 through mathematical operations

$X=x/(\text{width of image})$

$Y=y/(\text{height of image})$

$W=w/(\text{width of image})$

$H=h/(\text{height of image})$

Darknet format (class_id ,x,y,w,h)

Class_id=the index of the class name

(x ,y) center of the box

w: width of the box

h :height of the box

4 :cuda out of memory

It is very clear that the more we increase the size of the image, the easier the selection, so that the single box contains as much space as possible, but while we increased the size of the images, we fell into this problem, which is the fullness of the RAM size. After searching for a solution to this problem, we found several solutions, the first of which is reducing the image dimensions or raising subdivision value

5: colab time

Collab gives the average user 12 hours of work with a small amount of storage and little RAM with the possibility of deleting the session work in the event of any defect in the Internet connection, the work returns to the beginning. As for Collab Pro, it gives 24 working hours with higher storage and higher RAM, and here we have trained using the GPU

Discussion

These results shown above mean that we have reached a good accuracy, but it is not sufficient. This is due to the fact that the number of images must be increased to the largest possible number to cover all cases, and there is also some error annotation that must be modified and corrected in order to ensure proper training on sound data.

Conclusion

The process of learning through pictures is a very difficult process, but there is a great development in this field that may lead to good and satisfactory results, but counting people in crowds remains a very difficult process and requires more than just pictures and training.

References:

- [1] "AllGoVision," [Online]. Available: <http://allgovision.com/>
- . [2] "IOmniscient," [Online]. Available: http://www.iomniscient.com/index.php?option=com_content&view=article&id=154&Itemid=5
- . [3] "IPSOTEK," [Online]. Available: <http://www.ipsotek.com/>
- . [4] "Qognify," [Online]. Available: <http://www.qognify.com/video-analytics/>. [Accessed 10 12 2016]
- . [5] "Crowd Management Systems," Visio INGENII, 2015. [Online]. Available: <https://www.visioingenii.com/crowd-management-systems.php>
- . [6] A. B. Chan, Z. J. Liang and N. Vasconcelos, "Privacy preserving crowd monitoring: Counting people without people models or tracking," i
- [7] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. In ICLR, 2014. 1, 2, 4, 10
- YOLOv4: Optimal Speed and Accuracy of Object Detection: <https://arxiv.org/pdf/2004.10934.pdf>
 - SPP: <https://arxiv.org/pdf/1406.4729.pdf>
 - PaNet: <https://arxiv.org/pdf/1803.01534.pdf>
 - CSPNET: <https://arxiv.org/pdf/1911.11929.pdf>

- CBAM: Convolutional Block Attention
Module: <https://arxiv.org/pdf/1807.06521.pdf>
- EfficientDet: Scalable and Efficient Object
Detection: <https://arxiv.org/pdf/1911.09070.pdf>
- SELF-SUPERVISED ADVERSARIAL
TRAINING: <https://arxiv.org/pdf/1911.06470.pdf>