Other detection methods (YOLO, SSD, and AttentionNet)

AttentionNet: Aggregating Weak Directions for Accurate Object Detection

AttentionNet: Aggregating Weak Directions for Accurate Object Detection

Donggeun Yoo
dgyoo@rcv.kaist.ac.kr
KAIST

Sunggyun Park
sunggyun@kaist.ac.kr
KAIST

Joon-Young Lee*
jylee@rcv.kaist.ac.kr
KAIST

Anthony S. Paek
apaek@lunit.io
Lunit Inc.

In So Kweon
iskweon@kaist.ac.kr
KAIST

Abstract

We present a novel detection method using a deep convolutional neural network (CNN), named AttentionNet. We cast an object detection problem as an iterative classification problem, which is the most suitable form of a CNN. AttentionNet provides quantized weak directions pointing a target object and the ensemble of iterative predictions from AttentionNet converges to an accurate object boundary box. Since AttentionNet is a unified network for object detection, it detects objects without any separated models from the object proposal to the post bounding-box regression. We evaluate AttentionNet by a human detection task and achieve the state-of-the-art performance of 65% (AP) on PASCAL VOC 2007/2012 with an 8-layered architecture only.

1. Introduction

After the recent advance [16] of deep convolutional neural network (CNN) [7], CNN based object classification methods in computer vision has reached human-level performance on ILSVRC classification [18]: top-5 error of 4.94% [15], 6.67% [22], 6.8% [21], and 5.1% for human [18]. These successful methods, however, have limited range of application since most of the images are not object-centered. Thus, current research focus in visual recognition is moving towards richer image understanding such as object detection or pixel-level object segmentation. Our focus lies in the object detection problem.

Although many researchers had proposed various CNN-based techniques for object detection [24, 19, 12, 9], it is still challenge to estimate object bounding boxes beyond the object existence. Even the most successful framework so far, Region-CNN [12] reported top scores [17, 21, 22] in ILSVRC'14, but it is relatively far from human-level accuracy compared to the classification. One major limitation of this framework is that the proposal quality highly affects the detection performance. Another side of detection with CNN, regression models are also applied to detection [24, 19, 9], but direct mapping from an image to an exact bounding box is relatively difficult for a CNN. We thought that there must be a room for modification and improvement for the use of CNN as a regressor.

Figure 1. Real detection examples of our detection framework. Starting from an image boundary (dark blue bounding box), our detection system iteratively narrows the bounding box down to a final human location (red bounding box).

In this paper, we introduce a novel and straightforward detection method by integrating object existence estimation and bounding box optimization into a single convolutional network. Our model is on a similar line of a detection-by-regression approach, but we adopt a successful CNN-classification model rather than a CNN-regression model. We estimate an exact bounding box by aggregating many weak predictions from the classification model, such as an ensemble method combines many weak learners to produce a strong learner. We modify a traditional classification model to a suitable form, named ***AttentionNet***, for estimating an exact bounding box. This network provides quantized directions pointing a target object from the top-left (TL) and bottom-right (BR) corner of an image.

Fig. 1 shows real detection examples of our method. Starting from an entire image, the image is recursively cropped according to the predicted directions at TL/BR and fed to *AttentionNet* again, until the network converges to a bounding box fitting a target object. *Each direction is weak but the ensemble of directions is sufficient to estimate an exact bounding box.* The difficulty of estimating an exact bounding box at once with a regression model is solved by a classification model. For multiple instances, we also place our framework on the sliding window paradigm but introduce an efficient method to cope with those.

Compared with the previous state-of-the-art methods, a single AttentionNet does everything but yields state-of-the-art detection performance. Our framework does not involve any separated models for object proposals nor post bounding box regression. A single AttentionNet 1) detects regions where a single instance is included only, 2) provides directions to an instance at each region, 3) and also correct the miss-localizations such as bounding box regression [11].

AttentionNet we demonstrate in this paper is not scalable to multiple classes yet. Extension of this method to multiple classes with a single AttentionNet is ongoing. We therefore demonstrate single-class object detection on public datasets to verify the strength of our model. We primarily evaluate our detection framework by the human detection task and extend to a non-human class.

Contributions Our contributions are three folds:

1. We suggest a novel detection method, which estimates an exact bounding box by aggregating weak predictions from AttentionNet.
2. Our method does not include any separated models such as the object proposal, object classifiers and post bounding box regressor. AttentionNet does all these.
3. We achieve the state-of-the-art performance on single-class object detection tasks.

2. Related Works

Object detection has been actively studied for the last few decades. One of the most popular approaches is part-based models due to their strength in handling pose variations and occlusions. Deformable Part Model (DPM), proposed by Felzenszwalb *et al.* [11], is a flexible model composed of object parts combined with deformation cost to manage severe deformations. Poselets, proposed by Bourdev *et al.* [2], is another part-based model demonstrating competitive performance. Poselets has numerous part-wise HOG detectors covering wide pose variations. Activated parts vote the location of objects.

In recent years, CNN-based approach leads to the successful development of object detection with drastic advances of deep CNN [7]. Large scale ImageNet [8] database and the raise of parallel computing contribute to a breakthrough in detection [24, 19, 12, 17, 21, 22] as well as classification [16].

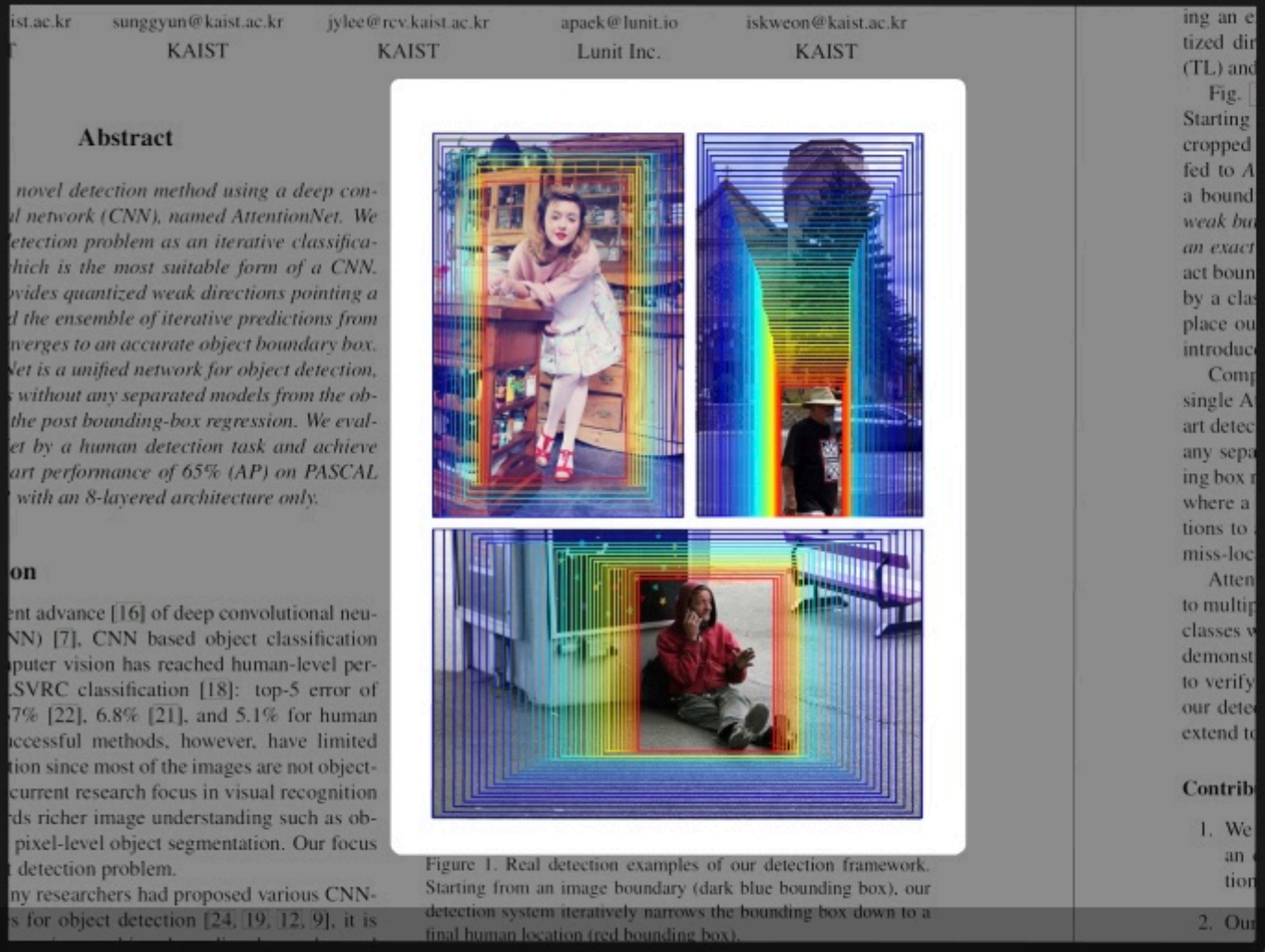
The state-of-the-art method in object detection is the Region-CNN (R-CNN), which represents a local region by CNN activations [12]. Specifically, R-CNN framework proceeds as follows. First it extracts local regions which probably contain an object by using an object proposal methods [26]. The local regions, called object proposals, are warped into a fixed size and fed to a pre-trained CNN. Then each proposal is represented by mid-level CNN activations (e.g. 7th convolutional layer) and evaluated by separated classifiers (e.g. SVMs). The object proposals are then merged and fed to a bounding box regressor [11] to correct miss-localizations. Despite its efficiency and successful performance, it has a limitation that proposal quality highly affects detection performance. If the proposal model fails to propose a suitable candidate, the rest procedures will not have the opportunity to detect it. For this reason, [9, 23] proposed a new class-agnostic proposal method with a CNN regression model to improve the proposal quality while reducing the number of proposals. Also, R-CNN is a cascaded model composed of individual components as object proposal, feature extraction, classification, and bounding box regression, therefore these should be individually engineered for the best performance.

Apart from R-CNN framework, there is another CNN-based approach, which considers object detection as a regression problem. Szegedy *et al.* [24] trains a CNN which maps an image to a rectangular mask of an object. Sermanet *et al.* [19] also employ a similar approach but their CNN directly estimates bounding box coordinates. These methods are free from object proposals, but it is a still debatable to leave all to a CNN trained with a mean-square cost to produce an exact bounding box.

Compared to the previous CNN methods, our method does not rely on object proposals since we actively explorer objects by iterative classifications. Also, the proposed network has a unified classification architecture, which is verified from many CNN applications and also does not need to tune up individual components.

*This work was done when he was in KAIST. He is currently working in Adobe Research.

Cast an object detection problem as an iterative classification problem.



ist.ac.kr sunggyun@kaist.ac.kr jylee@rcv.kaist.ac.kr apaek@lunit.io iskweon@kaist.ac.kr

KAIST KAIST Lunit Inc. KAIST

Abstract

novel detection method using a deep con-
l network (CNN), named AttentionNet. We
etection problem as an iterative classifica-
hich is the most suitable form of a CNN.
vides quantized weak directions pointing a
d the ensemble of iterative predictions from
verges to an accurate object boundary box.
let is a unified network for object detection,
without any separated models from the ob-
the post bounding-box regression. We eval-
et by a human detection task and achieve
art performance of 65% (AP) on PASCAL
with an 8-layered architecture only.

on

nt advance [16] of deep convolutional neu-
NN) [7], CNN based object classification
puter vision has reached human-level per-
LSVRC classification [18]: top-5 error of
7% [22], 6.8% [21], and 5.1% for human
ccessful methods, however, have limited
ion since most of the images are not object-
current research focus in visual recognition
ds richer image understanding such as ob-
pixel-level object segmentation. Our focus
detection problem.
ny researchers had proposed various CNN-
s for object detection [24, 19, 12, 9], it is

Figure 1. Real detection examples of our detection framework. Starting from an image boundary (dark blue bounding box), our detection system iteratively narrows the bounding box down to a final human location (red bounding box).

ing an e
tized dir
(TL) and
Fig.
Starting
cropped
fed to A
a bound
weak bu
an exact
act boun
by a clas
place ou
introduc
Comp
single A
art detec
any sepa
ing box r
where a
tions to
miss-loc
Atten
to multip
classes w
demonst
to verify
our dete
extend t

Contrib

1. *We*
   an
   tion
2. *Our*

Contributions

1. Proposed a novel detection method, which estimates an exact bounding box by aggregating weak predictions.

Contributions

1. Proposed a novel detection method, which estimates an exact bounding box by aggregating weak predictions.

2. Does not separate object proposal, classifiers, post bounding box regressions.

Contributions

1. Proposed a novel detection method, which estimates an exact bounding box by aggregating weak predictions.
2. Does not separate object proposal, classifiers, post bounding box regressions.
3. State-of-the-art on single class object detection tasks.

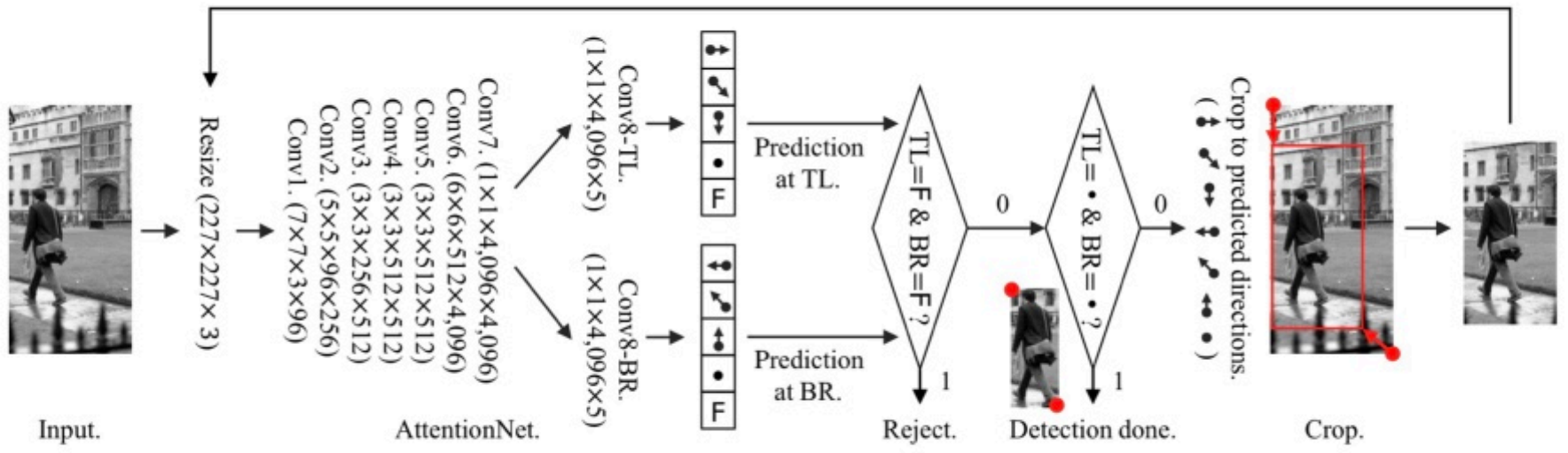


Figure 2. A pipeline of our detection framework. AttentionNet is composed of two final layers for top-left (TL) and bottom-right (BR) of the input image domain. Each of them outputs a direction (→ ↘ ↓ for TL, ← ↖ ↑ for BR) where each corner of the image should go to for the next step, or a "stop" sign (•), or "non-human" sign (F). When AttentionNet outputs "non-human" in both layers, the image is rejected. The image is cropped according to the weak directions and fed to AttentionNet again, until it meets "stop" in both layers.

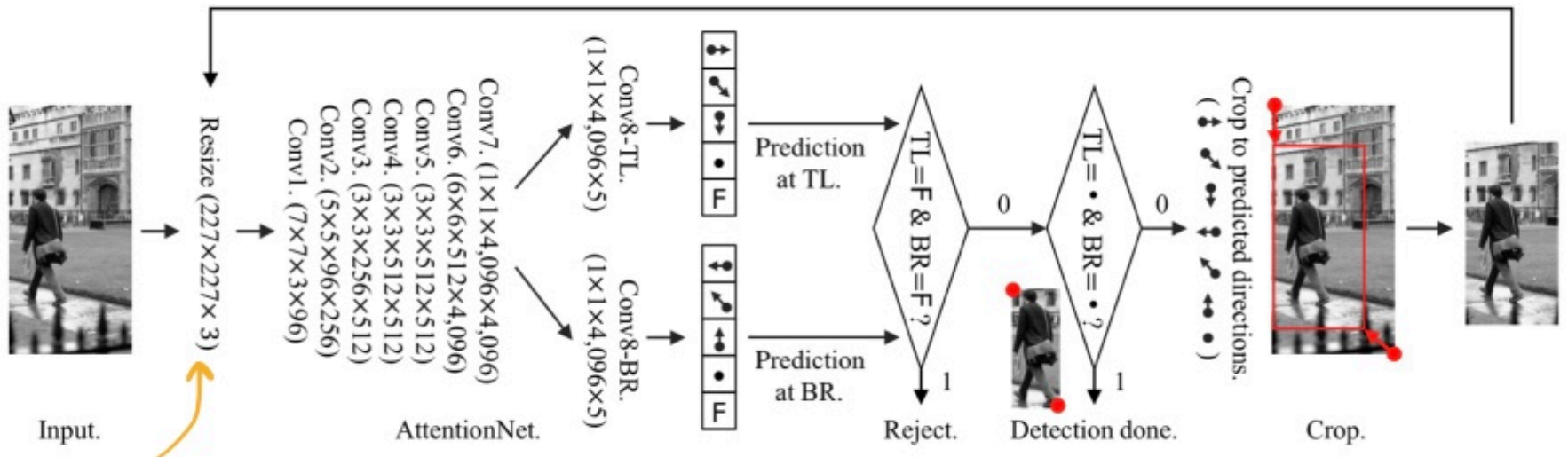


Figure 2. A pipeline of our detection framework. AttentionNet is composed of two final layers for top-left (TL) and bottom-right (BR) of the input image domain. Each of them outputs a direction (→ ↘ ↓ for TL, ← ↖ ↑ for BR) where each corner of the image should go to for the next step, or a "stop" sign (•), or "non-human" sign (F). When AttentionNet outputs "non-human" in both layers, the image is rejected. The image is cropped according to the weak directions and fed to AttentionNet again, until it meets "stop" in both layers.

1. Warp the given image to a fixed size

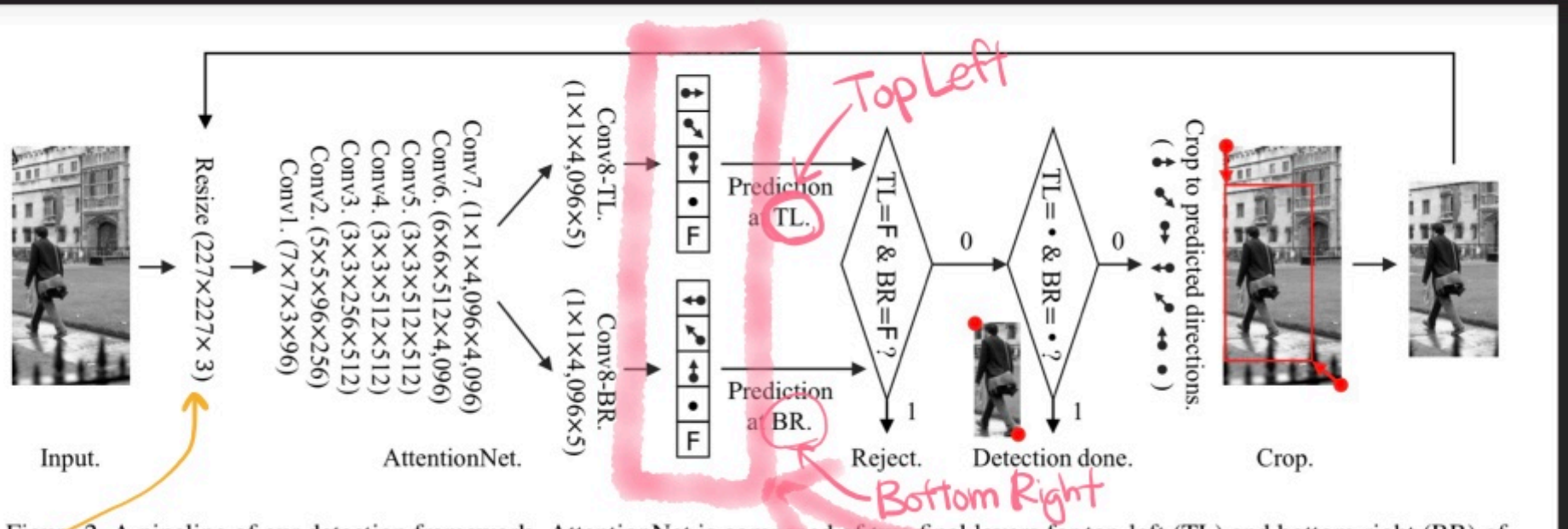


Figure 2. A pipeline of our detection framework. AttentionNet is composed of two final layers for top-left (TL) and bottom-right (BR) of the input image domain. Each of them outputs a direction (→ ↘ ↓ for TL, ← ↖ ↑ for BR) where each corner of the image should go to for the next step, or a "stop" sign (•), or "non-human" sign (F). When AttentionNet outputs "non-human" in both layers, the image is rejected. The image is cropped according to the weak directions and fed to AttentionNet again, until it meets "stop" in both layers.

1. Warp the given image to a fixed size
2. Outputs of the CNN are (5x2) dimensional vector
   → Modify the shape of the bounding box

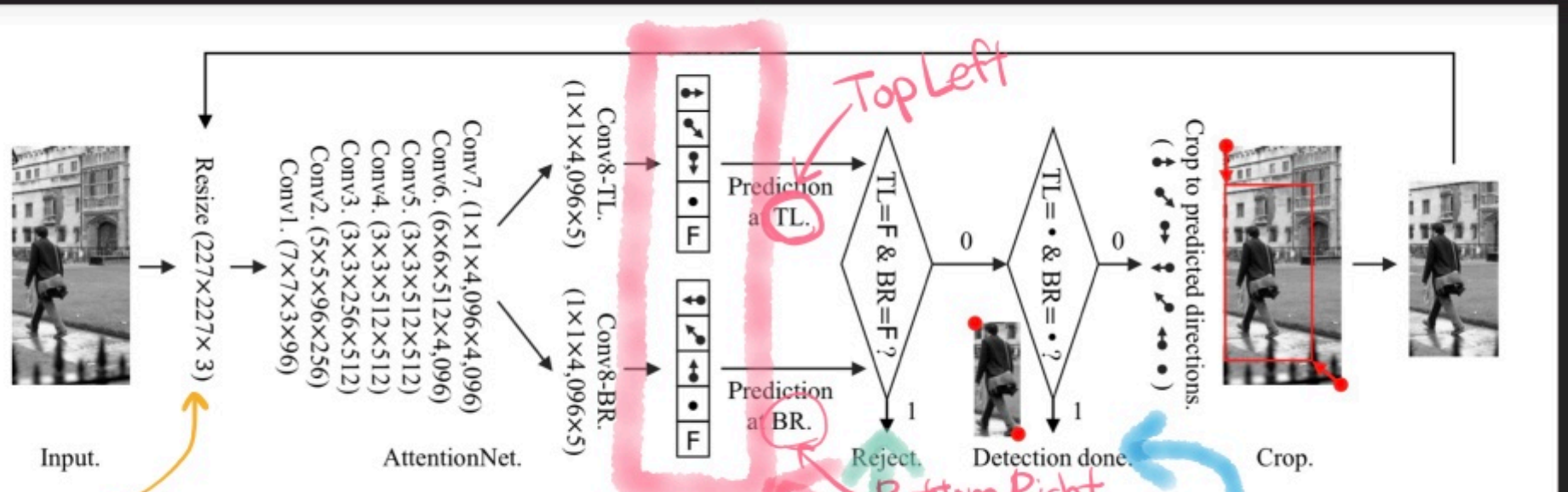


Figure 2. A pipeline of our detection framework. AttentionNet is composed of two final layers for top-left (TL) and bottom-right (BR) of the input image domain. Each of them outputs a direction (→ ↘ ↓ for TL, ← ↖ ↑ for BR) where each corner of the image should go to for the next step, or a "stop" sign (•), or "non-human" sign (F). When AttentionNet outputs "non-human" in both layers, the image is rejected. The image is cropped according to the weak directions and fed to AttentionNet again, until it meets "stop" in both layers.

1. Warp the given image to a fixed size
2. Outputs of the CNN are (5x2) dimensional vector
   → Modify the shape of the bounding box
3. If both predict "F" → reject
   "T" → detected

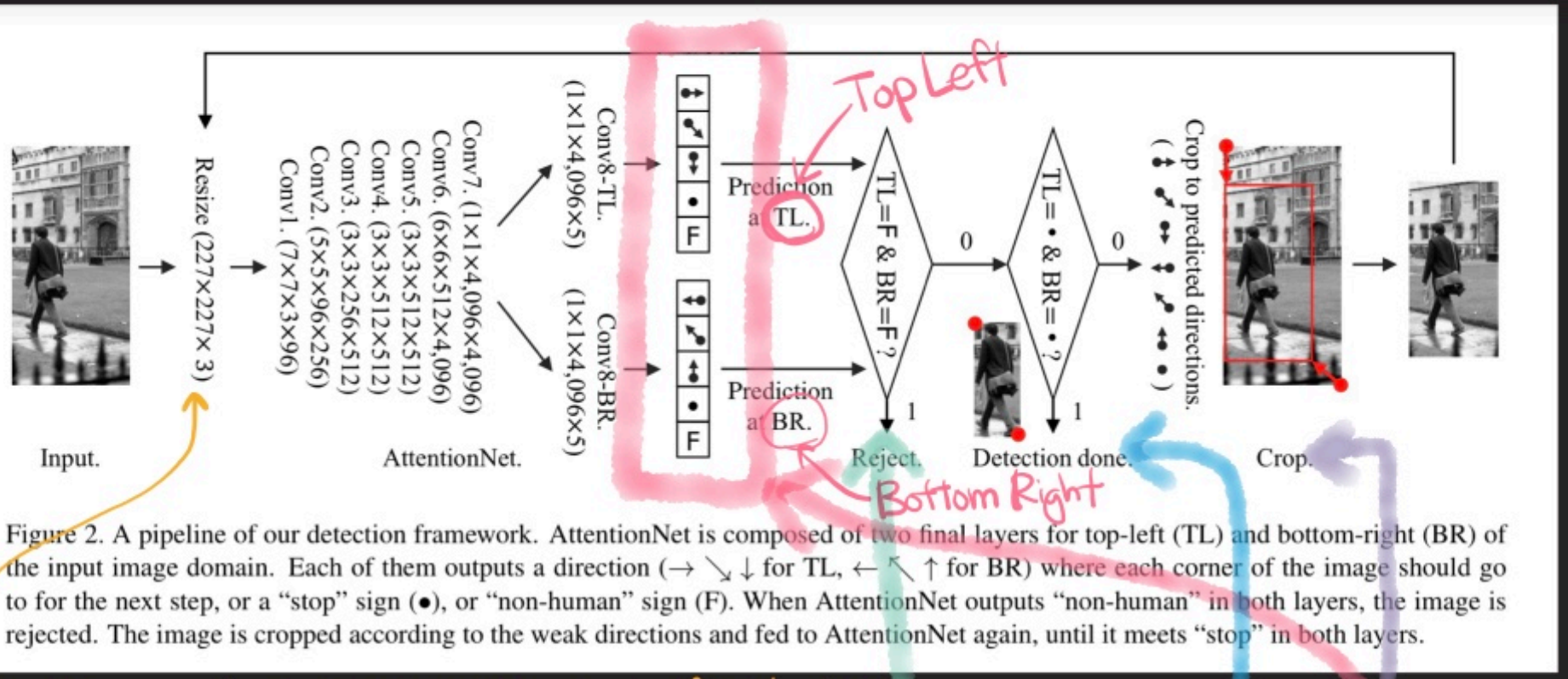


Figure 2. A pipeline of our detection framework. AttentionNet is composed of two final layers for top-left (TL) and bottom-right (BR) of the input image domain. Each of them outputs a direction (→ ↘ ↓ for TL, ← ↖ ↑ for BR) where each corner of the image should go to for the next step, or a "stop" sign (•), or "non-human" sign (F). When AttentionNet outputs "non-human" in both layers, the image is rejected. The image is cropped according to the weak directions and fed to AttentionNet again, until it meets "stop" in both layers.

1. Warp the given image to a fixed size
2. Outputs of the CNN are (5x2) dimensional vector
   → Modify the shape of the bounding box
3. If both predict "F" → reject
   "T" → detected
4. Otherwise, change the shape of the bounding box.

Training phase

Figure 3. Real examples of crop-augmentation for training AttentionNet. The target instance is the right man. Dashed cyan bounding boxes are ground-truths, and the blue bounding boxes are the augmented regions. Red arrows/dots denote their ground truths.

negative region will "reject" current bounding box.

1. Positive region must include at least 50% of the area of the target.

Training phase

Figure 3. Real examples of crop-augmentation for training AttentionNet. The target instance is the right man. Dashed cyan bounding boxes are ground-truths, and the blue bounding boxes are the augmented regions. Red arrows/dots denote their ground truths.

negative region will "reject" current bounding box.

1. Positive region must include at least 50% of the area of the target.

2. For region with multiple targets, go with the largest one.

Training phase

Figure 3. Real examples of crop-augmentation for training AttentionNet. The target instance is the right man. Dashed cyan bounding boxes are ground-truths, and the blue bounding boxes are the augmented regions. Red arrows/dots denote their ground truths.

negative region will "reject" current bounding box.

1. Positive region must include at least 50% of the area of the target.

2. For region with multiple targets, go with the largest one.

3. Initial crop is done with multiple scales and ratios.

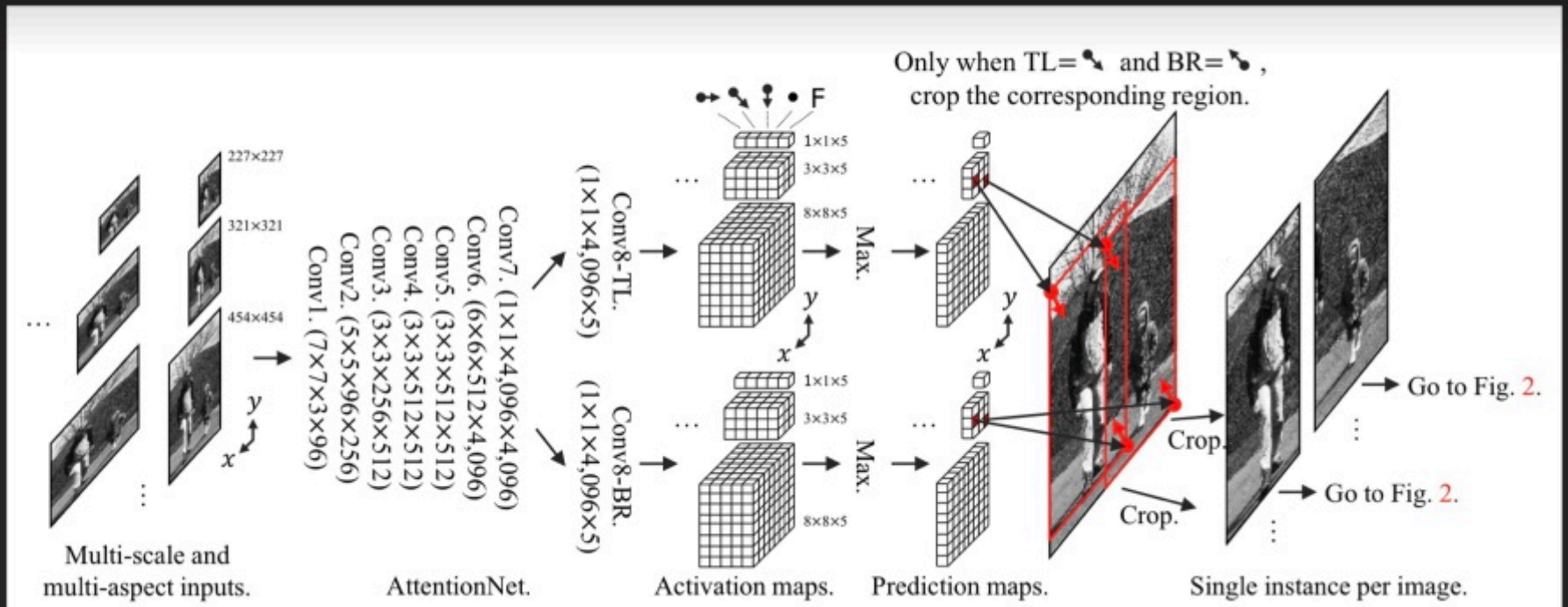


Figure 4. Extracting single-instance regions, where a single instance is included only. Multiple inputs with multiple scales/aspects are fed to AttentionNet, and prediction maps are produced. Only image regions satisfying $\{\searrow_{TL}, \nwarrow_{BR}\}$ are regarded as the single-instance regions. These regions are fed to AttentionNet again for final detection. Note, the CNN here and that in Fig. 2 are the same one, not separated.

Handling multiple inputs with fully convolutional network (fcn).

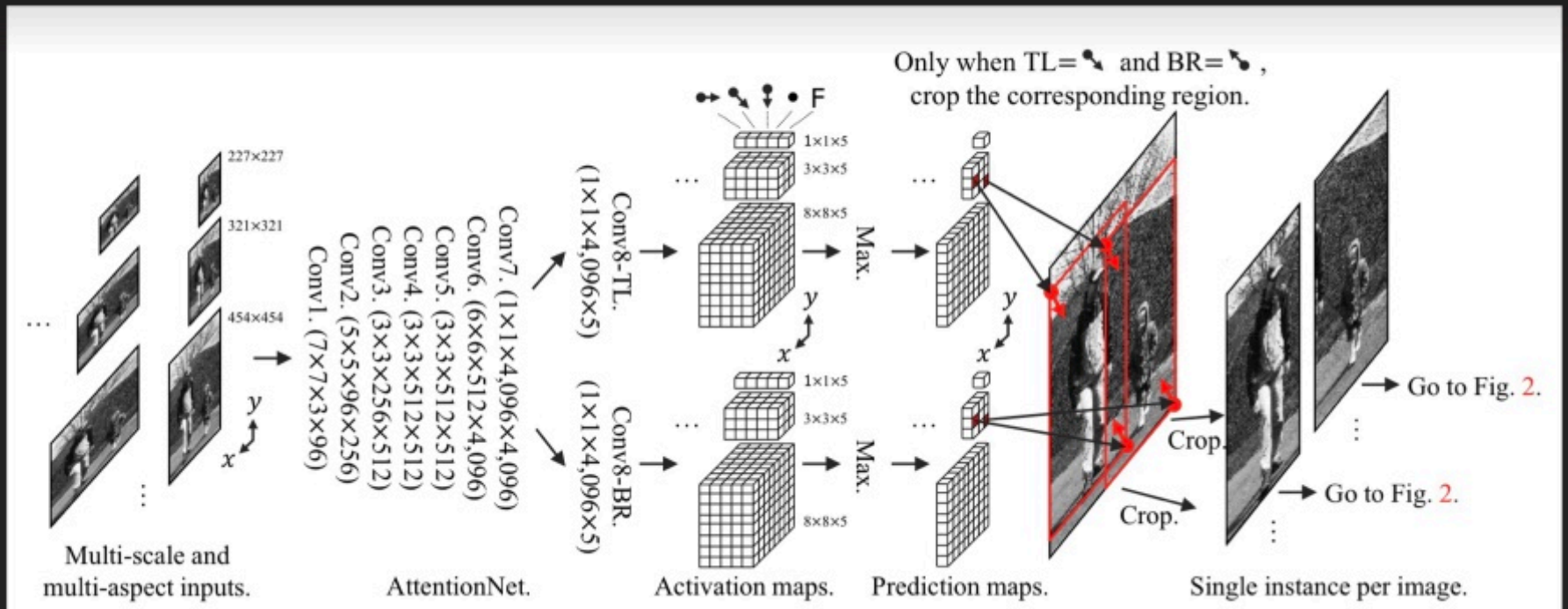


Figure 4. Extracting single-instance regions, where a single instance is included only. Multiple inputs with multiple scales/aspects are fed to AttentionNet, and prediction maps are produced. Only image regions satisfying $\{\searrow_{TL}, \nwarrow_{BR}\}$ are regarded as the single-instance regions. These regions are fed to AttentionNet again for final detection. Note, the CNN here and that in Fig. 2 are the same one, not separated.

Handling multiple inputs with fully convolutional network (fcn).

Bounding boxes with $\{\searrow_{TL}, \nwarrow_{BR}\}$ are cropped and fed to the network

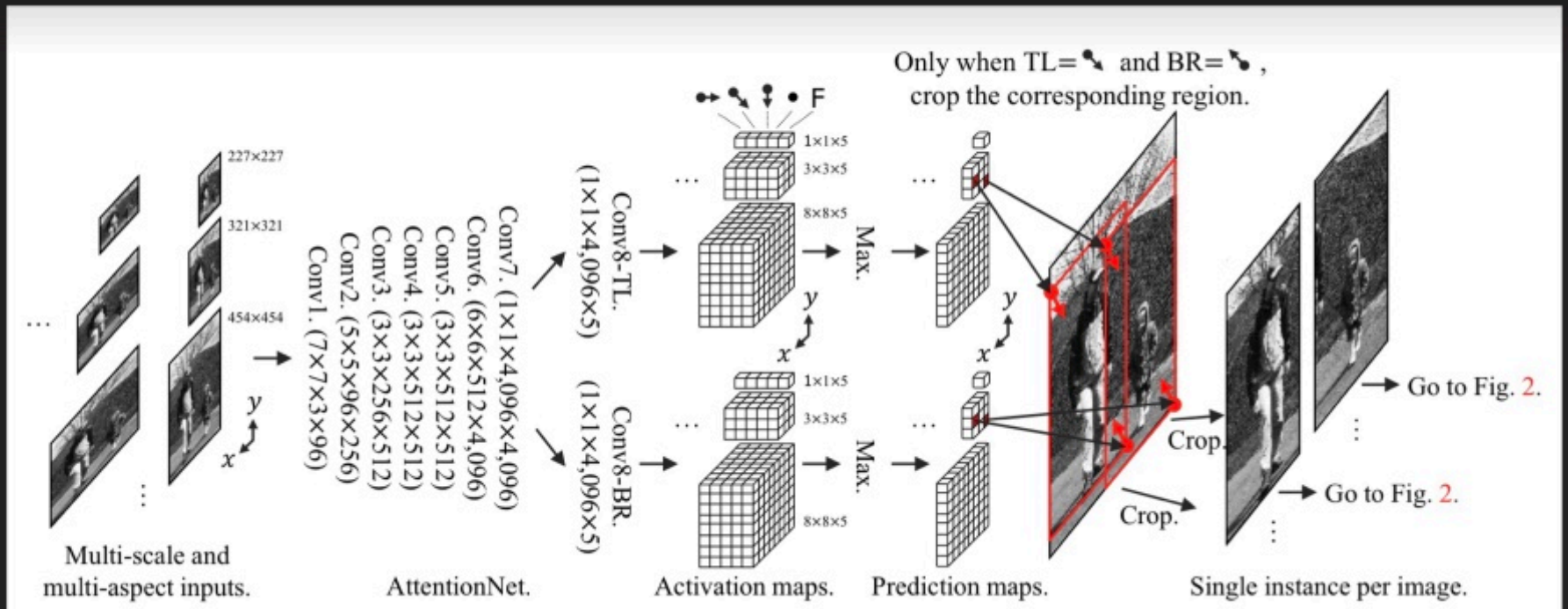
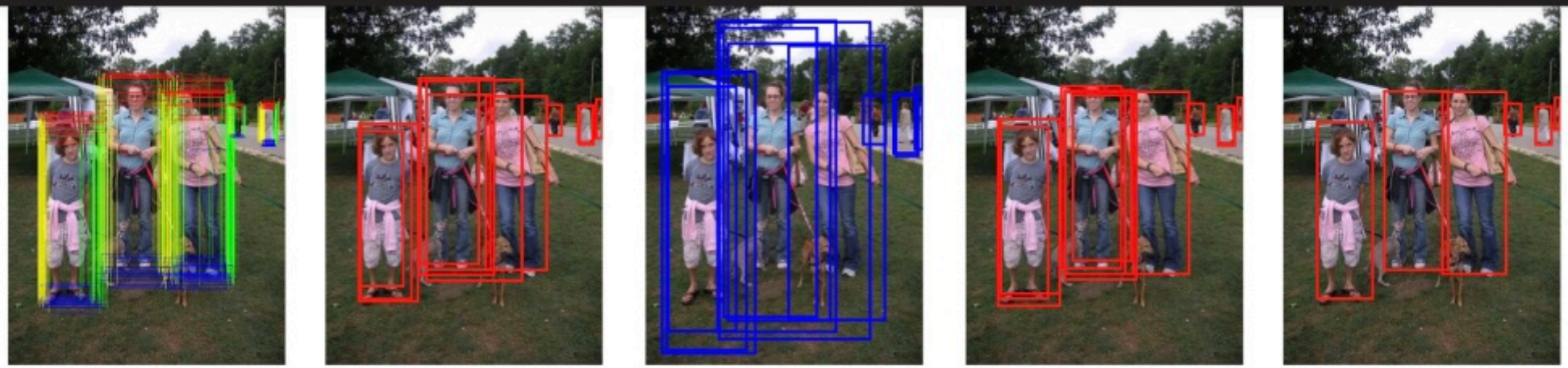


Figure 4. Extracting single-instance regions, where a single instance is included only. Multiple inputs with multiple scales/aspects are fed to AttentionNet, and prediction maps are produced. Only image regions satisfying $\{\searrow_{TL}, \nwarrow_{BR}\}$ are regarded as the single-instance regions. These regions are fed to AttentionNet again for final detection. Note, the CNN here and that in Fig. 2 are the same one, not separated.

Handling multiple inputs with fully convolutional network (fcn).

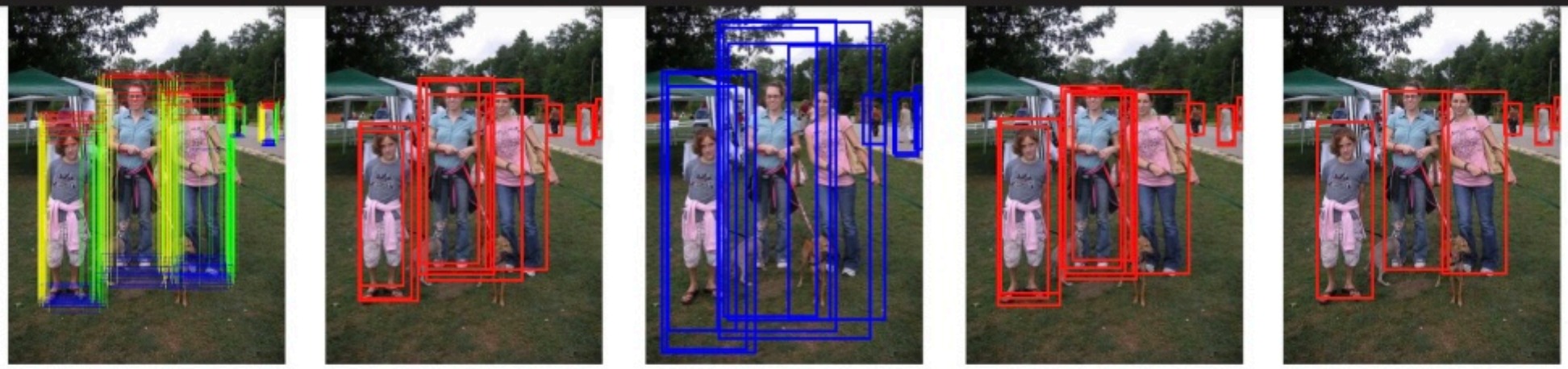
Bounding boxes with $\{\searrow_{TL}, \nwarrow_{BR}\}$ are cropped and fed to the network



(a) Initial detections. (b) Initial merge. (c) Re-initialize. (×2.5) (d) Re-detections. (e) Final merge.

Figure 6. Real examples of our detection procedure, including initial results (a∼b) and refinement (c∼e). Initially detected candidates come from Fig. 4 followed by Fig. 2 are merged by an intersection over union (IoU) of 0.8. We extend each merged box to 2.5-times larger size, and feed them to Fig. 2 again. Finally we merge the second results by an IoU of 0.5.

1. Multiple detection results will be produced.

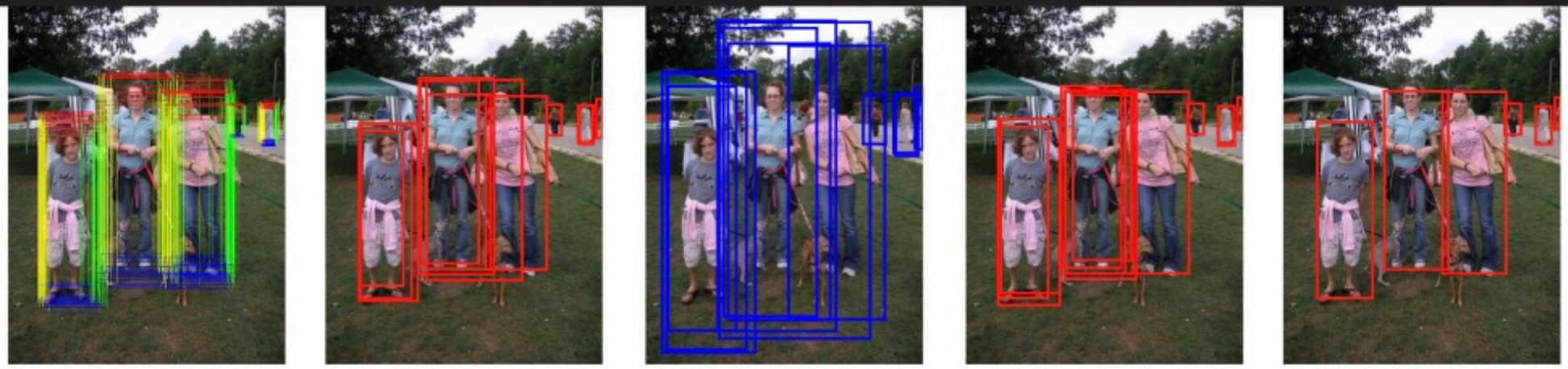


(a) Initial detections. (b) Initial merge. (c) Re-initialize. (×2.5) (d) Re-detections. (e) Final merge.

Figure 6. Real examples of our detection procedure, including initial results (a∼b) and refinement (c∼e). Initially detected candidates come from Fig. 4 followed by Fig. 2 are merged by an intersection over union (IoU) of 0.8. We extend each merged box to 2.5-times larger size, and feed them to Fig. 2 again. Finally we merge the second results by an IoU of 0.5.

1. Multiple detection results will be produced.

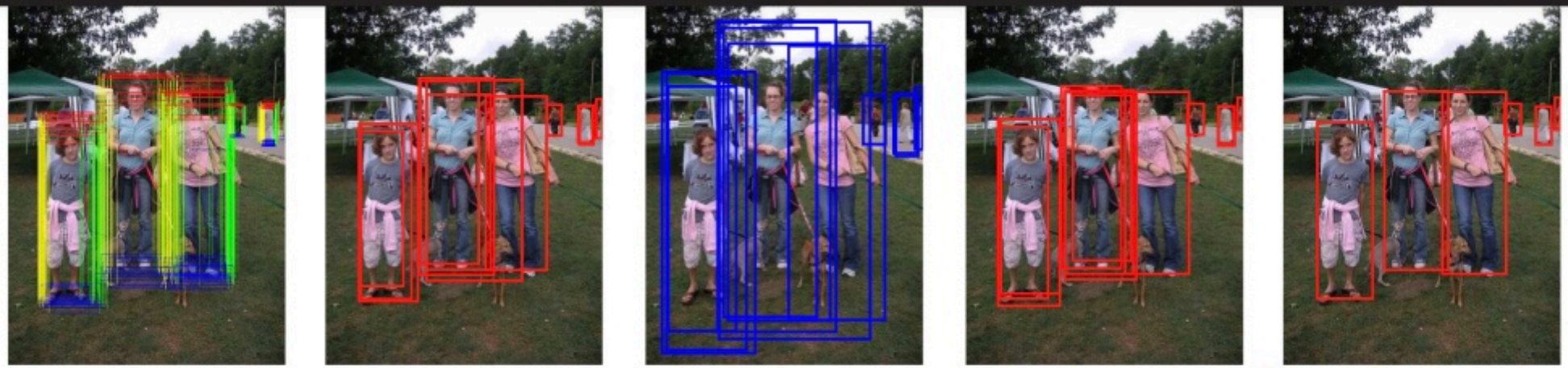
2. Initial merge : bounding boxes with IoU bigger-than α_0 are averaged.



(a) Initial detections. (b) Initial merge. (c) Re-initialize. (×2.5) (d) Re-detections. (e) Final merge.

Figure 6. Real examples of our detection procedure, including initial results (a∼b) and refinement (c∼e). Initially detected candidates come from Fig. 4 followed by Fig. 2 are merged by an intersection over union (IoU) of 0.8. We extend each merged box to 2.5-times larger size, and feed them to Fig. 2 again. Finally we merge the second results by an IoU of 0.5.

1. Multiple detection results will be produced.
2. Initial merge : bounding boxes with IoU bigger-than α_0 are averaged.
3. Enlarge IoU of enlarged bounding boxes

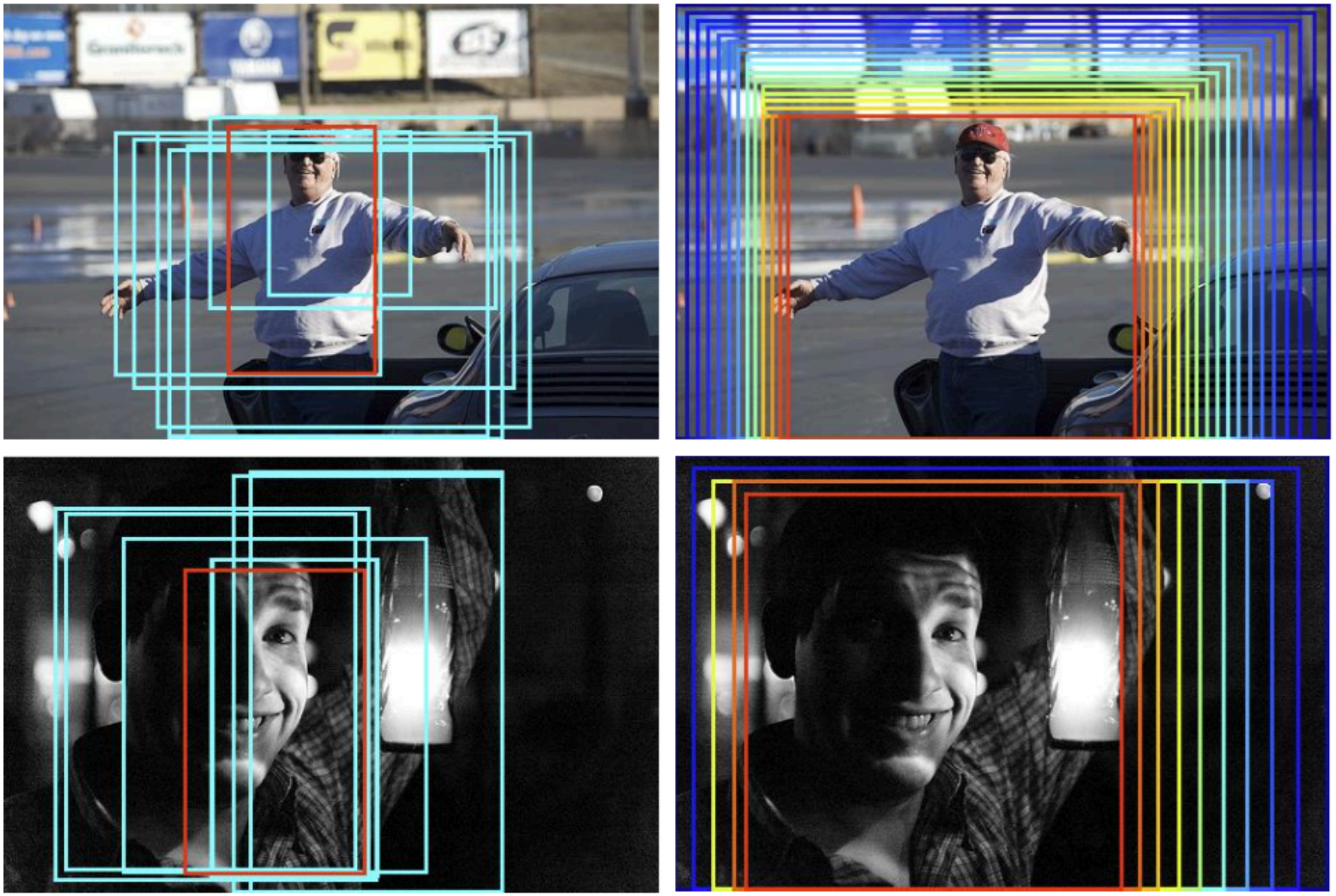


(a) Initial detections. (b) Initial merge. (c) Re-initialize. (×2.5) (d) Re-detections. (e) Final merge.

Figure 6. Real examples of our detection procedure, including initial results (a∼b) and refinement (c∼e). Initially detected candidates come from Fig. 4 followed by Fig. 2 are merged by an intersection over union (IoU) of 0.8. We extend each merged box to 2.5-times larger size, and feed them to Fig. 2 again. Finally we merge the second results by an IoU of 0.5.

1. Multiple detection results will be produced.
2. Initial merge : bounding boxes with IoU bigger-than α_0 are averaged.
3. Enlarge IoU of enlarged bounding boxes
4. Run again & merge

Results



You Only Look Once: Unified, Real-Time Object Detection

Joseph Redmon*, Santosh Divvala*†, Ross Girshick¶, Ali Farhadi*†
University of Washington*, Allen Institute for AI†, Facebook AI Research¶
http://pjreddie.com/yolo/

Abstract

We present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

Our unified architecture is extremely fast. Our base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors. Compared to state-of-the-art detection systems, YOLO makes more localization errors but is less likely to predict false positives on background. Finally, YOLO learns very general representations of objects. It outperforms other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains like artwork.

1. Introduction

Humans glance at an image and instantly know what objects are in the image, where they are, and how they interact. The human visual system is fast and accurate, allowing us to perform complex tasks like driving with little conscious thought. Fast, accurate algorithms for object detection would allow computers to drive cars without specialized sensors, enable assistive devices to convey real-time scene information to human users, and unlock the potential for general purpose, responsive robotic systems.

Current detection systems repurpose classifiers to perform detection. To detect an object, these systems take a classifier for that object and evaluate it at various locations and scales in a test image. Systems like deformable parts models (DPM) use a sliding window approach where the classifier is run at evenly spaced locations over the entire image [10].

More recent approaches like R-CNN use region proposal methods to first generate potential bounding boxes in an image and then run a classifier on these proposed boxes. After classification, post-processing is used to refine the bounding boxes, eliminate duplicate detections, and rescore the boxes based on other objects in the scene [13]. These complex pipelines are slow and hard to optimize because each individual component must be trained separately.

Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448×448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

We reframe object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. Using our system, you only look once (YOLO) at an image to predict what objects are present and where they are.

YOLO is refreshingly simple: see Figure 1. A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This unified model has several benefits over traditional methods of object detection.

First, YOLO is extremely fast. Since we frame detection as a regression problem we don't need a complex pipeline. We simply run our neural network on a new image at test time to predict detections. Our base network runs at 45 frames per second with no batch processing on a Titan X GPU and a fast version runs at more than 150 fps. This means we can process streaming video in real-time with less than 25 milliseconds of latency. Furthermore, YOLO achieves more than twice the mean average precision of other real-time systems. For a demo of our system running in real-time on a webcam please see our project webpage: http://pjreddie.com/yolo/.

Second, YOLO reasons globally about the image when making predictions. Unlike sliding window and region proposal-based techniques, YOLO sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their appearance. Fast R-CNN, a top detection method [14], mistakes background patches in an image for objects because it can't see the larger context. YOLO makes less than half the number of background errors compared to Fast R-CNN.

Third, YOLO learns generalizable representations of objects. When trained on natural images and tested on artwork, YOLO outperforms top detection methods like DPM and R-CNN by a wide margin. Since YOLO is highly generalizable it is less likely to break down when applied to new domains or unexpected inputs.

YOLO still lags behind state-of-the-art detection systems in accuracy. While it can quickly identify objects in images it struggles to precisely localize some objects, especially small ones. We examine these tradeoffs further in our experiments.

All of our training and testing code is open source. A variety of pretrained models are also available to download.

2. Unified Detection

We unify the separate components of object detection into a single neural network. Our network uses features from the entire image to predict each bounding box. It also predicts all bounding boxes across all classes for an image simultaneously. This means our network reasons globally about the full image and all the objects in the image. The YOLO design enables end-to-end training and real-time speeds while maintaining high average precision.

Our system divides the input image into an $S \times S$ grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object.

Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts. Formally we define confidence as $\Pr(\text{Object}) * \text{IOU}^{\text{truth}}_{\text{pred}}$. If no object exists in that cell, the confidence scores should be zero. Otherwise we want the confidence score to equal the intersection over union (IOU) between the predicted box and the ground truth.

Each bounding box consists of 5 predictions: x, y, w, h, and confidence. The (x, y) coordinates represent the center of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image. Finally the confidence prediction represents the IOU between the predicted box and any ground truth box.

Each grid cell also predicts C conditional class probabilities, $\Pr(\text{Class}_i|\text{Object})$. These probabilities are conditioned on the grid cell containing an object. We only predict one set of class probabilities per grid cell, regardless of the number of boxes B.

At test time we multiply the conditional class probabilities and the individual box confidence predictions,

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}^{\text{truth}}_{\text{pred}} = \Pr(\text{Class}_i) * \text{IOU}^{\text{truth}}_{\text{pred}} \quad (1)$$

which gives us class-specific confidence scores for each box. These scores encode both the probability of that class appearing in the box and how well the predicted box fits the object.

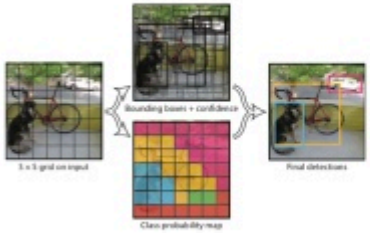


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

For evaluating YOLO on PASCAL VOC, we use $S = 7$, $B = 2$. PASCAL VOC has 20 labelled classes so $C = 20$. Our final prediction is a $7 \times 7 \times 30$ tensor.

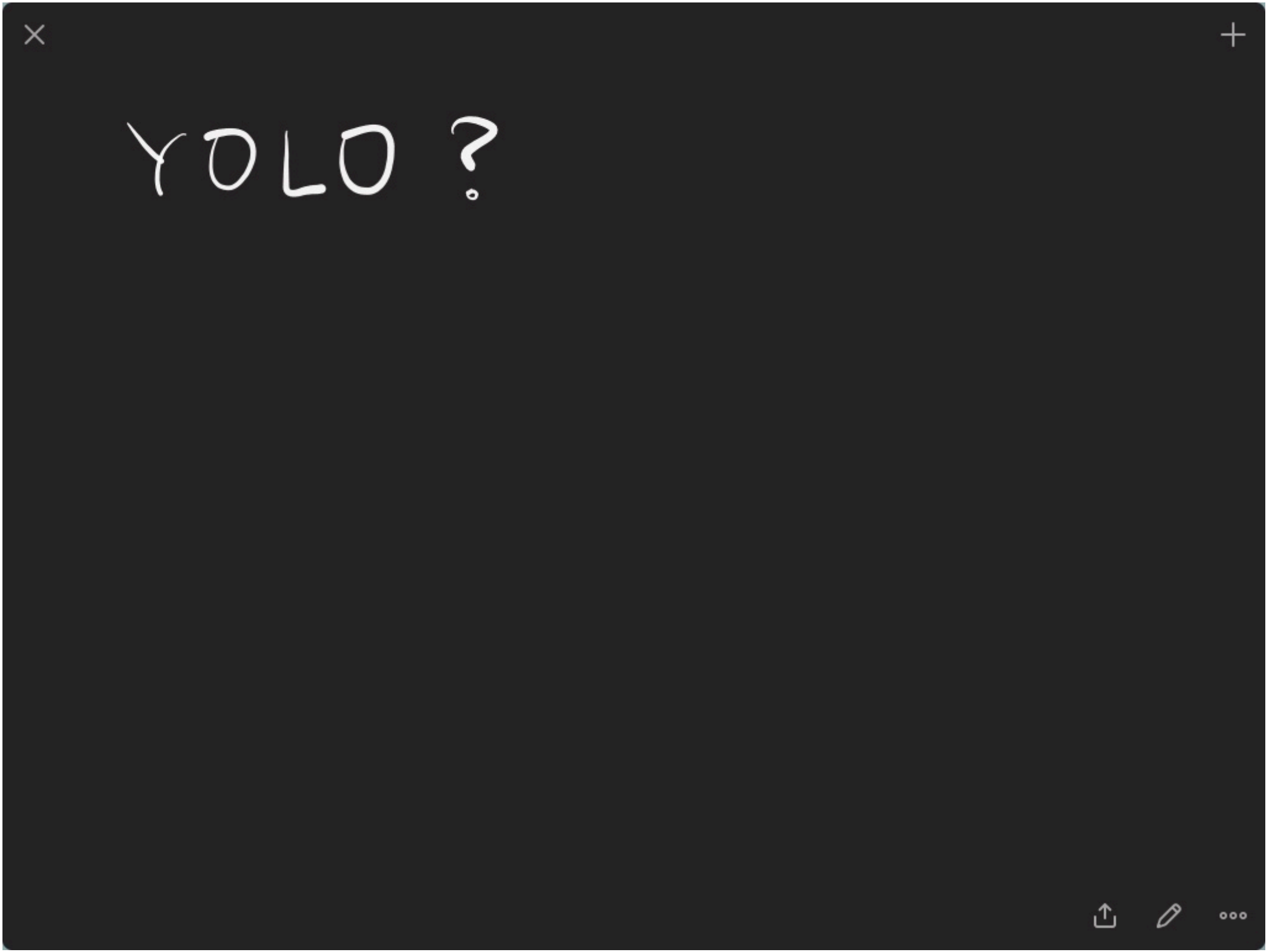
2.1. Network Design

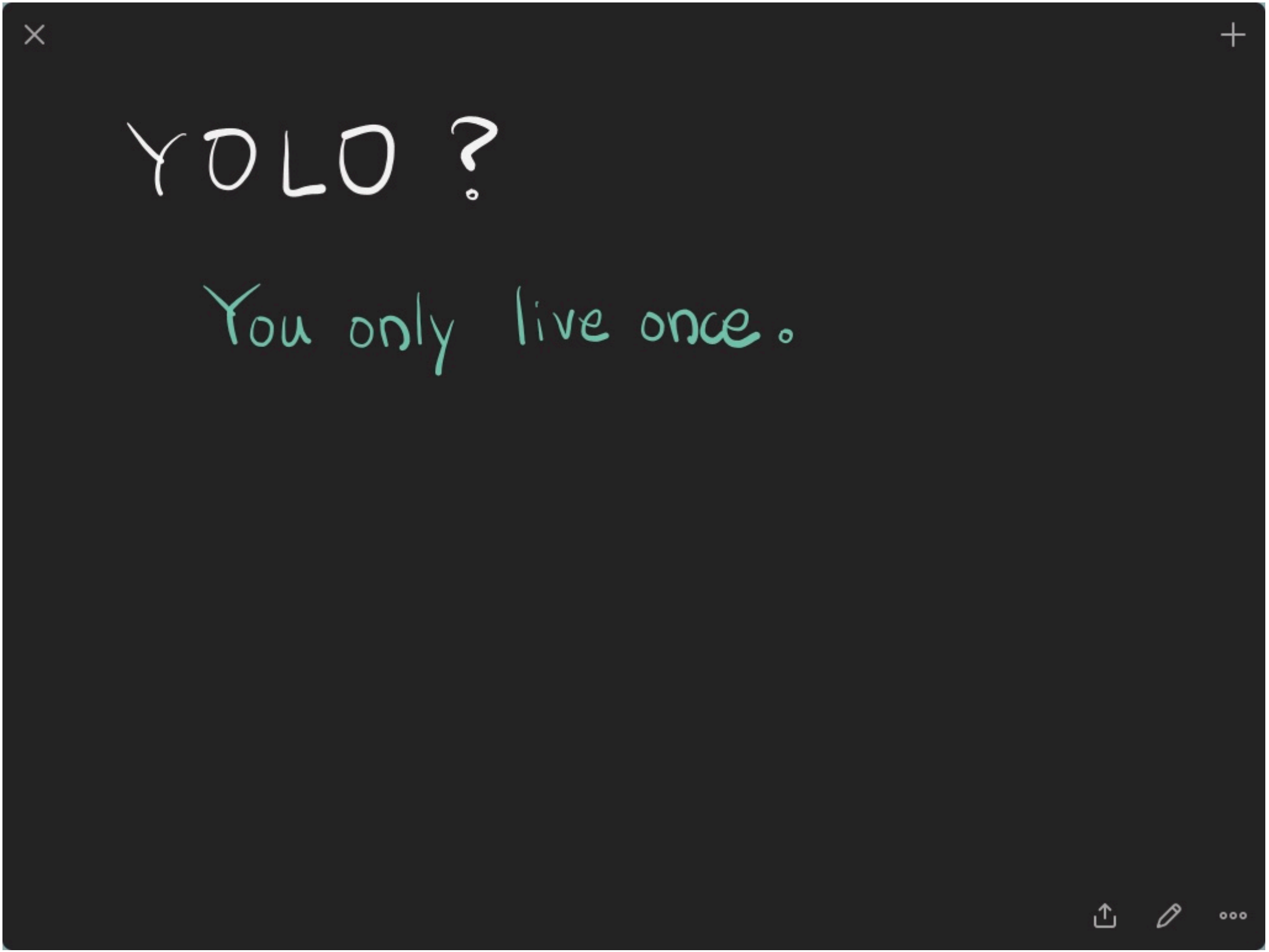
We implement this model as a convolutional neural network and evaluate it on the PASCAL VOC detection dataset [9]. The initial convolutional layers of the network extract features from the image while the fully connected layers predict the output probabilities and coordinates.

Our network architecture is inspired by the GoogLeNet model for image classification [34]. Our network has 24 convolutional layers followed by 2 fully connected layers. Instead of the inception modules used by GoogLeNet, we simply use 1×1 reduction layers followed by 3×3 convolutional layers, similar to Lin et al [22]. The full network is shown in Figure 3.

We also train a fast version of YOLO designed to push the boundaries of fast object detection. Fast YOLO uses a neural network with fewer convolutional layers (9 instead of 24) and fewer filters in those layers. Other than the size of the network, all training and testing parameters are the same between YOLO and Fast YOLO.

YOLO?


YOLO ?


YOLO ?
You only live once.

YOLO ?

You only ~~live~~ once.

You only look once!

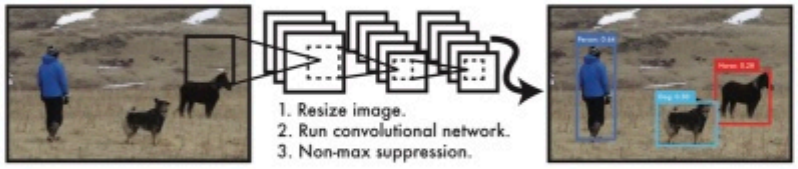


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448×448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

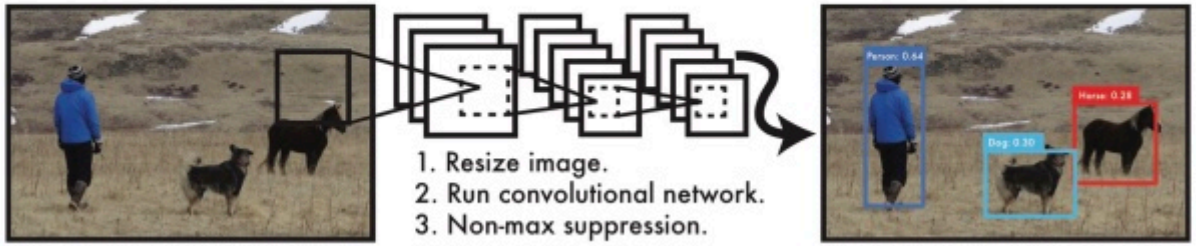


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

Object detection algorithm

Extremely fast!

Baseline: 45 fps

Smaller version: 155 fps

YOLO

You Only **Look** Once

Quiet similar with Faster R-CNN and very FAST!

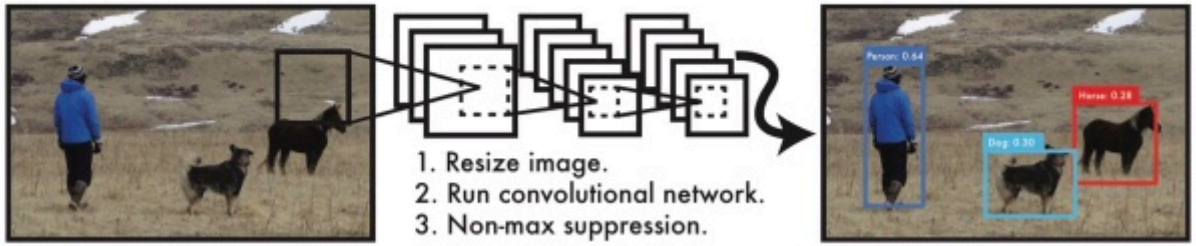


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

Object detection algorithm

Extremely fast!

Baseline: 45 fps

Smaller version: 155 fps

→ Simultaneously predicts multiple bounding boxes and class probabilities.

⇓

Frame "Detection" problem as a "Regression" problem.

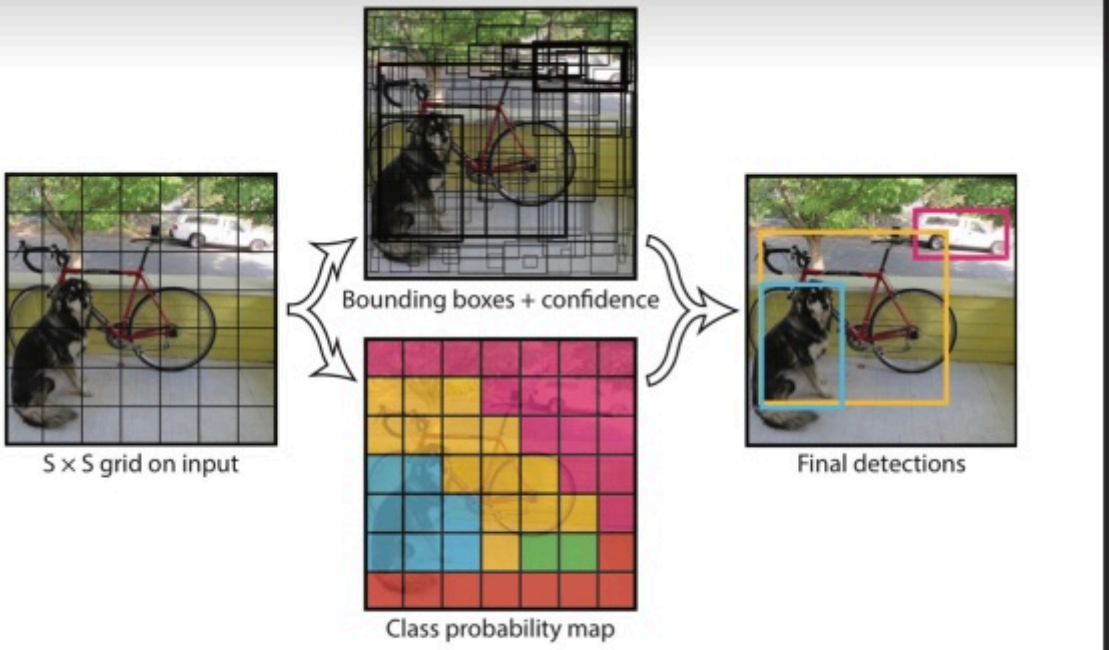


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

This is YOLO.

- No bounding box sampling.

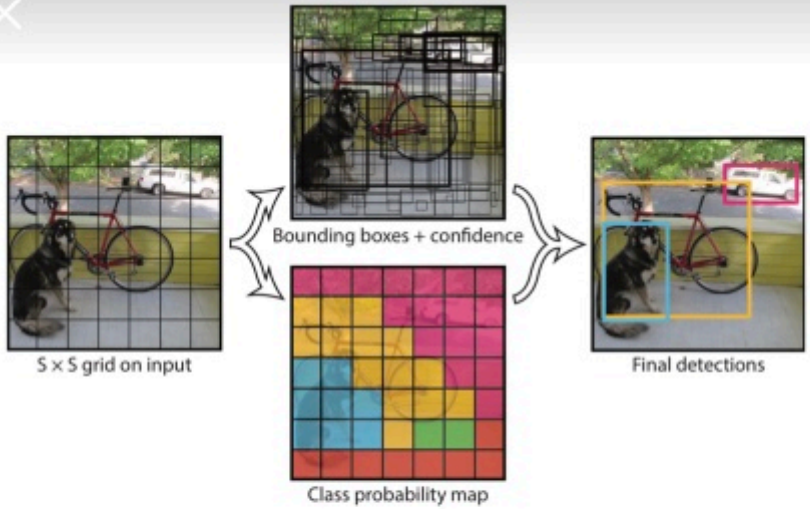


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

1. Given an image, divide it into an $S \times S$ grid.

⇒ If the "center" of an object falls into the grid cell, that grid cell is responsible.

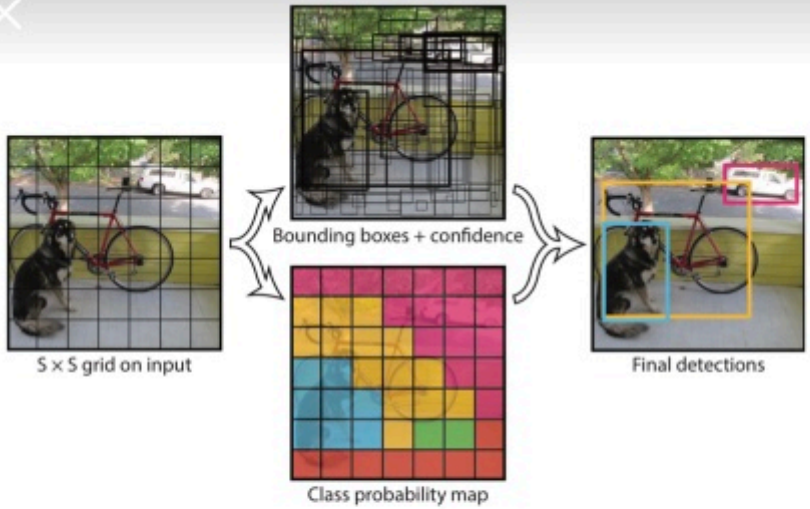


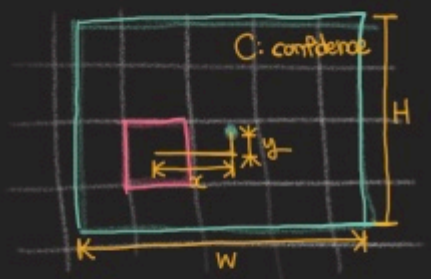
Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

1. Given an image, divide it into an S×S grid.

⇒ If the "center" of an object falls into the grid cell, that grid cell is responsible.

2. Each cell predicts B bounding boxes.

⇒



Five predictions
x, y, w, h, conf

3. Each cell predicts C class probabilites.

⇒ 1 cell → C class probabilites
B×5 bounding boxes

* One cell → One class probability.
: Low false positive!

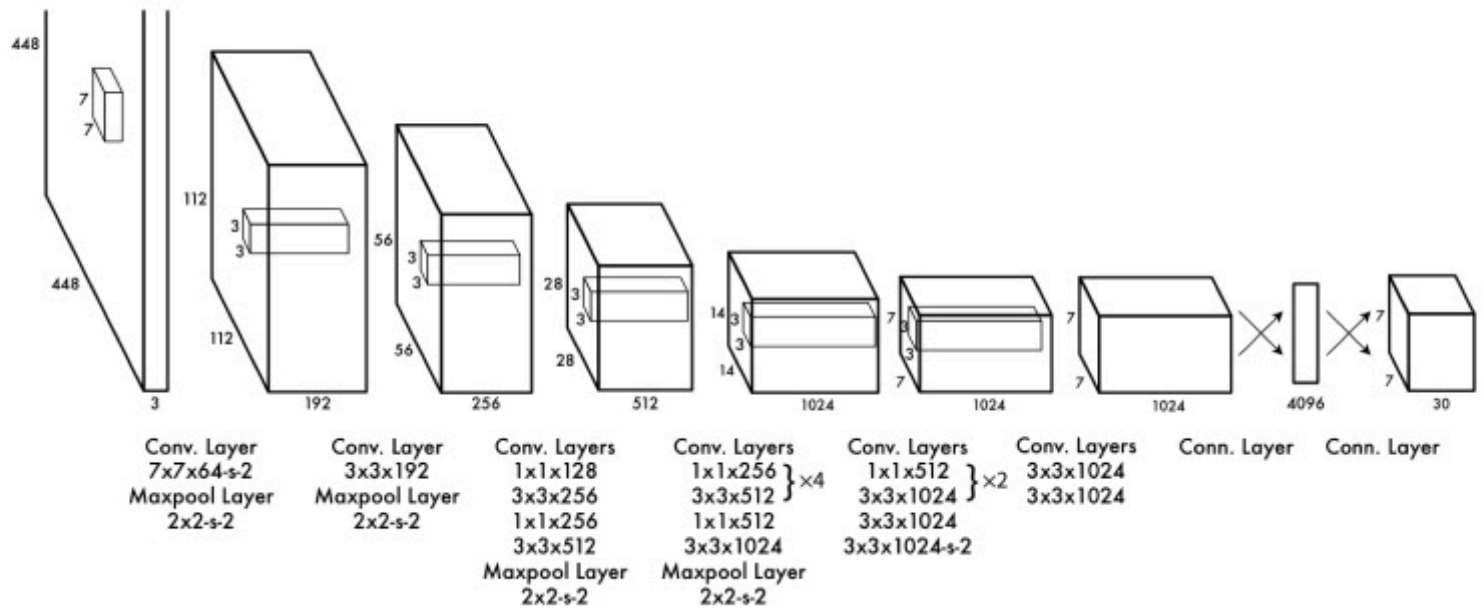


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

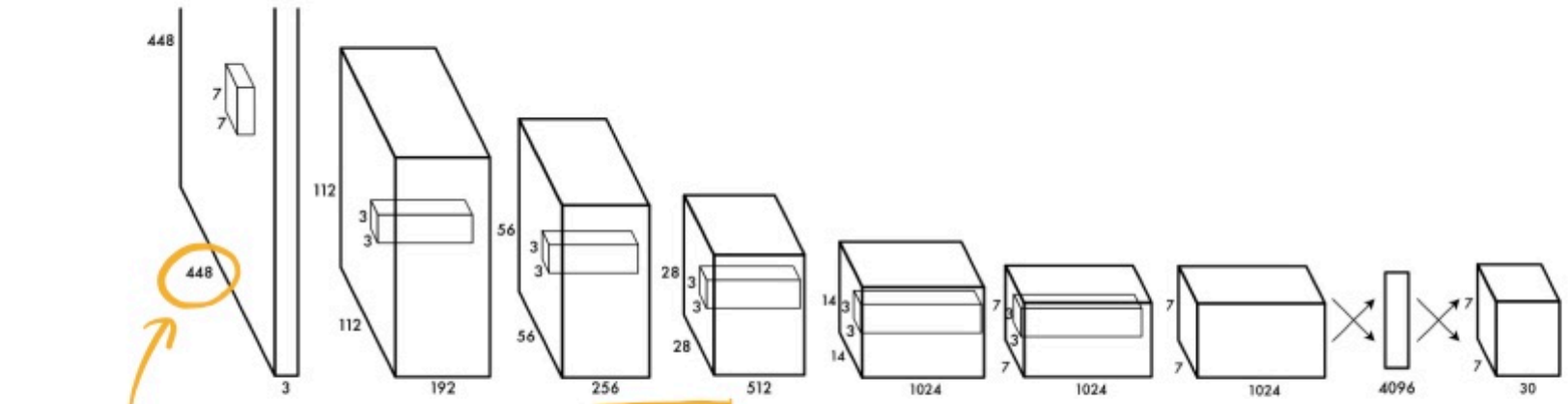


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

Bigger than usual.

Inception like model.

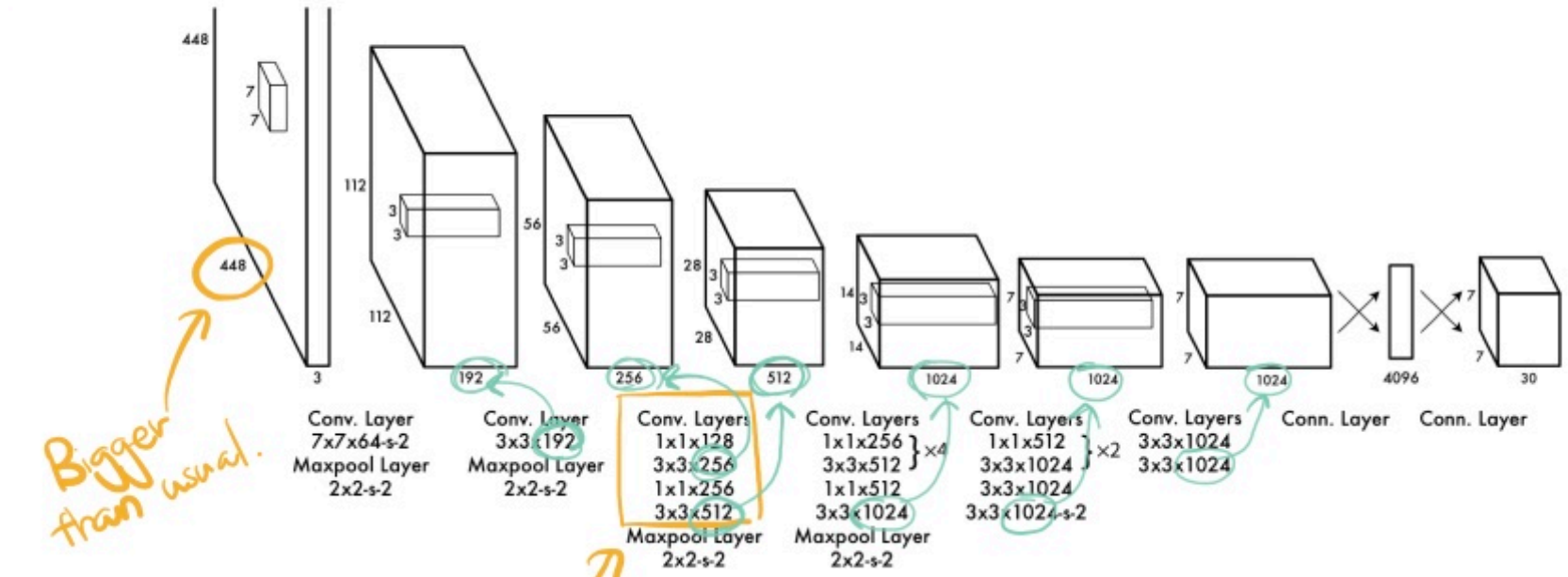


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

param ?

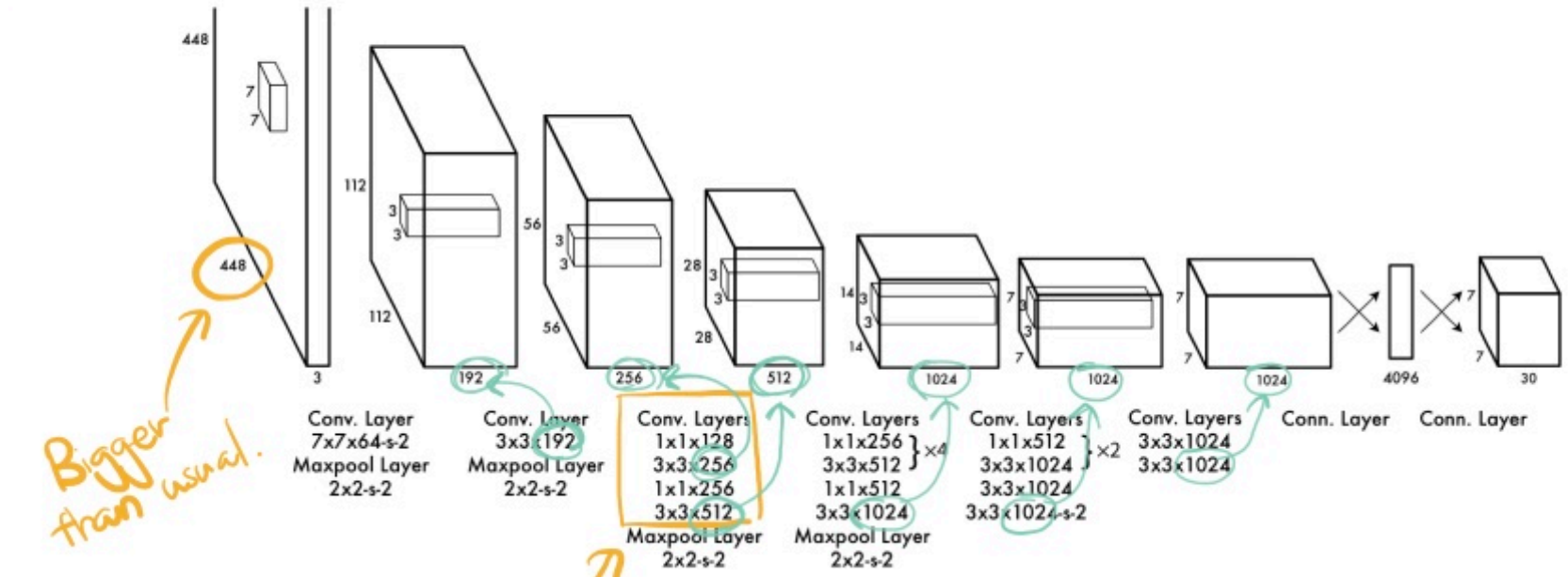
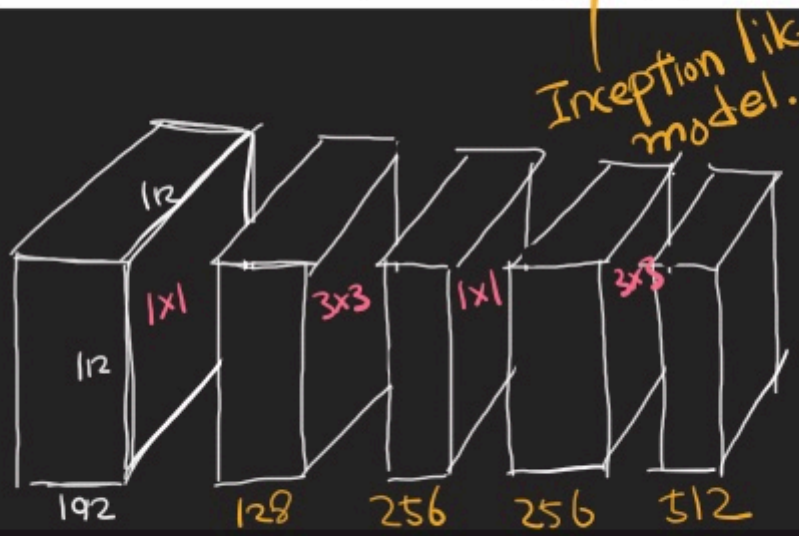


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.



param ?

1x1 x 192 x 128
+
3x3 x 128 x 256
+
1x1 x 256 x 256 + 3x3 x 256 x 512.

YOLO predicts multiple bounding boxes per grid cell.

In training, one predictor is responsible. which has the highest IoU with the ground truth.

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2
\end{aligned}
$$

YOLO predicts multiple bounding boxes per grid cell.

In training, one predictor is responsible.

which has the highest IoU with the ground truth.

for all grid

for all bounding boxes

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

If jth BB is responsible,

center pos

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

width / height

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

confidence

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

← class probability

Limitations?

1. Each grid cell can predict only B(=2) bounding boxes and one class probability.

   ⇒ Not good for small objects that flock together.

2. Uses relatively coarse features.

   ⇒ Locations of BBs are inaccurate.

3. Loss function treats errors in small BB and big BB equally.

   ⇒ Not good for scoring.

Results


person
person
horse
person
person
person
person
bottle
diningtable
person
dog
person
person
cat
bird
bird
aeroplane
car
car
person
boat
aeroplane
aeroplane
aeroplane
aeroplane
aeroplane
aeroplane

SSD: Single Shot MultiBox Detector

SSD: Single Shot MultiBox Detector

Wei Liu[1], Dragomir Anguelov[2], Dumitru Erhan[3], Christian Szegedy[3], Scott Reed[4], Cheng-Yang Fu[1], Alexander C. Berg[1]

[1]UNC Chapel Hill [2]Zoox Inc. [3]Google Inc. [4]University of Michigan, Ann-Arbor
[1]wliu@cs.unc.edu, [2]drago@zoox.com, [3]{dumitru,szegedy}@google.com,
[4]reedscot@umich.edu, [1]{cyfu,aberg}@cs.unc.edu

Abstract. We present a method for detecting objects in images using a single deep neural network. Our approach, named SSD, discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. Additionally, the network combines predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes. SSD is simple relative to methods that require object proposals because it completely eliminates proposal generation and subsequent pixel or feature resampling stages and encapsulates all computation in a single network. This makes SSD easy to train and straightforward to integrate into systems that require a detection component. Experimental results on the PASCAL VOC, COCO, and ILSVRC datasets confirm that SSD has competitive accuracy to methods that utilize an additional object proposal step and is much faster, while providing a unified framework for both training and inference. For 300×300 input, SSD achieves 74.3% mAP[1] on VOC2007 `test` at 59 FPS on a Nvidia Titan X and for 512×512 input, SSD achieves 76.9% mAP, outperforming a comparable state-of-the-art Faster R-CNN model. Compared to other single stage methods, SSD has much better accuracy even with a smaller input image size. Code is available at: `https://github.com/weiliu89/caffe/tree/ssd`.

SSD : Single Shot MultiBox Detector

SSD : Single Shot MultiBox Detector

⇓ No object proposals

⇓ There exists a set of default boxes over different aspect ratios and scales

SSD : Single Shot MultiBox Detector

⇓ No object proposals

⇓ There exists a set of default boxes over different aspect ratios and scales

Existing detectors work as follows:

SSD : Single Shot MultiBox Detector

No object proposals

There exists a set of default boxes over different aspect ratios and scales

Existing detectors work as follows:

Hypothesize bounding boxes → Extract features at each box → Apply high-quality classifier

The state of the art detector, Faster R-CNN, operates at 7 FPS with mAP 73.2% on VOC 2007.

SSD : Single Shot MultiBox Detector

No object proposals

There exists a set of default boxes over different aspect ratios and scales

Existing detectors work as follows:

Hypothesize bounding boxes → Extract features at each box → Apply high-quality classifier

The state of the art detector, Faster R-CNN, operates at 7 FPS with mAP 73.2% on VOC 2007.

YOLO : 45 FPS with mAP 63.4 %

SSD : 59 FPS with mAP 74.3 %

SSD : Single Shot MultiBox Detector

⇓ No object proposals

⇓ There exists a set of default boxes over different aspect ratios and scales

The core of SSD is

1) predicting [category scores
box offsets for a fixed set of default boxes

SSD : Single Shot MultiBox Detector

⇓ No object proposals

⇓ There exists a set of default boxes over different aspect ratios and scales

The core of SSD is

1) predicting [category scores
box offsets for a fixed set of default boxes

2) from each cell of multiple convolutional feature maps.

SSD : Single Shot MultiBox Detector

No object proposals

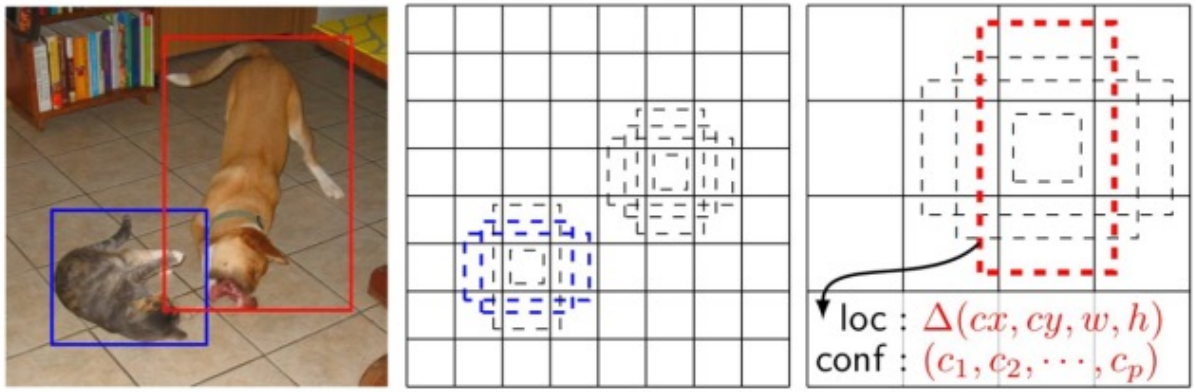
There exists a set of default boxes over different aspect ratios and scales

The core of SSD is

1) predicting [category scores
box offsets] for a fixed set of default boxes → Anchors in Faster R-CNN

2) from each cell of multiple convolutional feature maps.
→ YOLO
→ DeconvNet

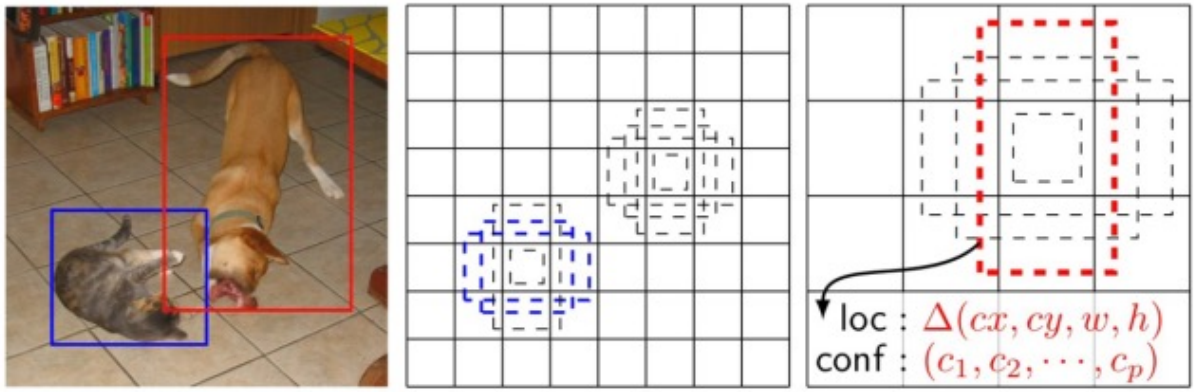
For each cell in feature maps,
we have k default boxes (anchor boxes)



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Fig. 1: **SSD framework.** (a) SSD only needs an input image and ground truth boxes for each object during training. In a convolutional fashion, we evaluate a small set (e.g. 4) of default boxes of different aspect ratios at each location in several feature maps with different scales (e.g. 8×8 and 4×4 in (b) and (c)). For each default box, we predict both the shape offsets and the confidences for all object categories ($(c_1, c_2, \cdots, c_p)$). At training time, we first match these default boxes to the ground truth boxes. For example, we have matched two default boxes with the cat and one with the dog, which are treated as positives and the rest as negatives. The model loss is a weighted sum between localization loss (e.g. Smooth L1 [6]) and confidence loss (e.g. Softmax).

For each cell in feature maps,
we have k default boxes (anchor boxes)
For each box, we have c per-class scores and 4 offsets
⇒ (c+4)k outputs per cell.

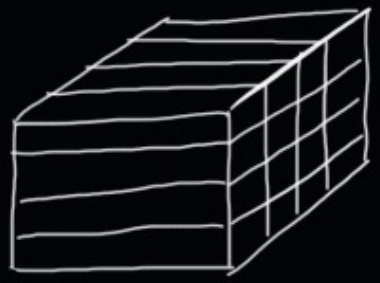


(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

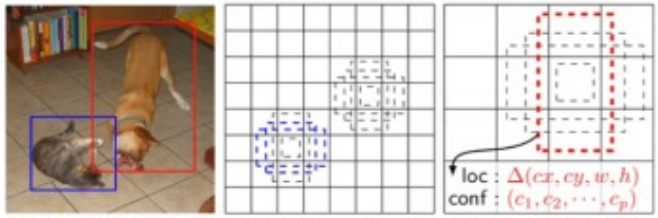
Fig. 1: **SSD framework.** (a) SSD only needs an input image and ground truth boxes for each object during training. In a convolutional fashion, we evaluate a small set (e.g. 4) of default boxes of different aspect ratios at each location in several feature maps with different scales (e.g. 8×8 and 4×4 in (b) and (c)). For each default box, we predict both the shape offsets and the confidences for all object categories ($(c_1, c_2, \cdots, c_p)$). At training time, we first match these default boxes to the ground truth boxes. For example, we have matched two default boxes with the cat and one with the dog, which are treated as positives and the rest as negatives. The model loss is a weighted sum between localization loss (e.g. Smooth L1 [6]) and confidence loss (e.g. Softmax).

For each cell in feature maps,
we have k default boxes (anchor boxes)
For each box, we have c per-class scores and 4 offsets
$\Rightarrow (c+4)k$ outputs per cell.

Input image



4x4 Feature map



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Fig. 1: **SSD framework.** (a) SSD only needs an input image and ground truth boxes for each object during training. In a convolutional fashion, we evaluate a small set (e.g. 4) of default boxes of different aspect ratios at each location in several feature maps with different scales (e.g. 8×8 and 4×4 in (b) and (c)). For each default box, we predict both the shape offsets and the confidences for all object categories ($(c_1, c_2, \cdots, c_p)$). At training time, we first match these default boxes to the ground truth boxes. For example, we have matched two default boxes with the cat and one with the dog, which are treated as positives and the rest as negatives. The model loss is a weighted sum between localization loss (e.g. Smooth L1 [6]) and confidence loss (e.g. Softmax).

For each cell in feature maps,
we have k default boxes (anchor boxes)
For each box, we have c per-class scores and 4 offsets
$\Rightarrow (c+4)k$ outputs per cell.

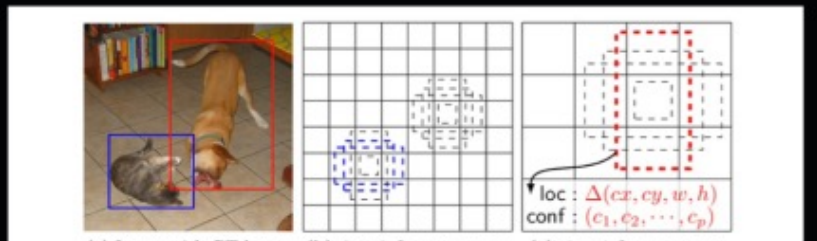


Fig. 1: **SSD framework.** (a) SSD only needs an input image and ground truth boxes for each object during training. In a convolutional fashion, we evaluate a small set (e.g. 4) of default boxes of different aspect ratios at each location in several feature maps with different scales (e.g. 8×8 and 4×4 in (b) and (c)). For each default box, we predict both the shape offsets and the confidences for all object categories ($(c_1, c_2, \cdots, c_p)$). At training time, we first match these default boxes to the ground truth boxes. For example, we have matched two default boxes with the cat and one with the dog, which are treated as positives and the rest as negatives. The model loss is a weighted sum between localization loss (e.g. Smooth L1 [6]) and confidence loss (e.g. Softmax).

1. The size of an input image is different.
2. SSD uses multiple feature maps.
3. While YOLO outputs box positions, SSD outputs box offsets of six default boxes.

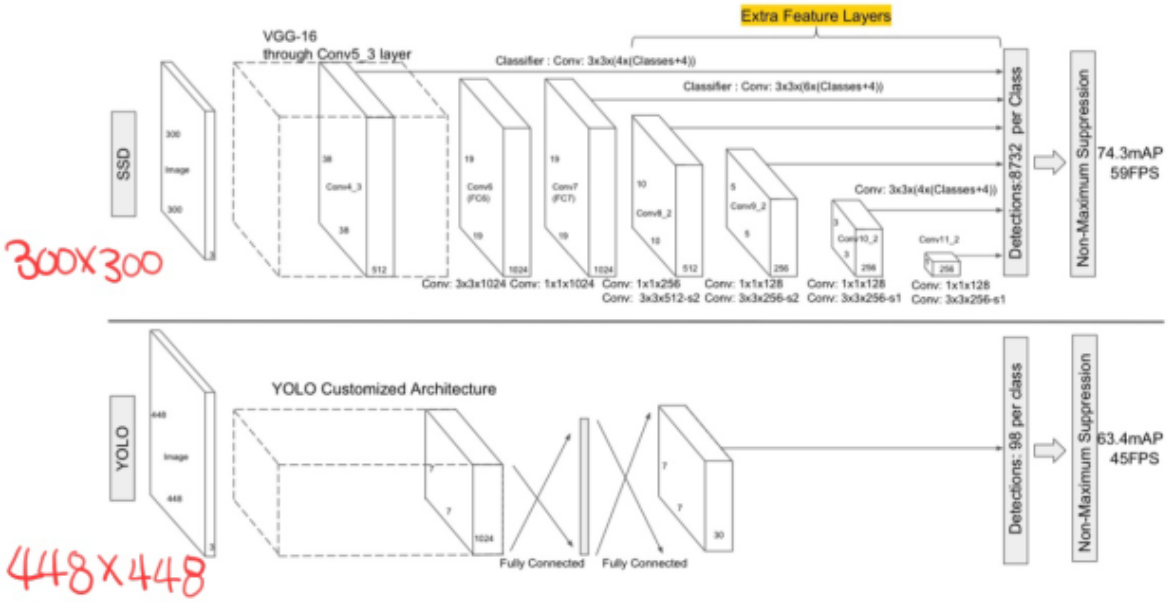


Fig. 2: A comparison between two single shot detection models: SSD and YOLO [5]. Our SSD model adds several feature layers to the end of a base network, which predict the offsets to default boxes of different scales and aspect ratios and their associated confidences. SSD with a 300×300 input size significantly outperforms its 448×448 YOLO counterpart in accuracy on VOC2007 `test` while also improving the speed.

Results