DenseBox: Unifying Landmark Localization with End to End Object Detection

Lichao Huang
Institute of Deep Learning
Baidu Research
huanglichao01@baidu.com

Abstract

We propose DenseBox, a unified end-to-end Convolutional Neural Networks (CNNs) framework for object detection. Unlike previous detection methods such as R-CNN which requires region proposal generation as the first step, our network directly predicts bounding boxes and object class confidences through all the locations and scales of the input images. We show that such a single network, if optimized carefully, works extremely well in detecting important common objects such as faces and cars, where traditional R-CNN could fail due to the small size of objects and heavy occlusion. Our system is simple and fast enough to be trained from scratch without the need of pre-training. Since DenseBox is a fully convolutional neural network artchitecture, it is straightforward to incorporate multi-task learning. We demonstrate that object detection performance could be significantly improved by incorporating landmark localization during joint training and inference. We present experimental results on public benchmark datasets including MALF (Multi-Attribute Labeled Faces) face detection and KITTI car detection, that indicate our approach is the state-of-the-art system for object detection, outperforming R-CNN while being order of magnitude faster.

1 Introduction

As one of the core problems in image understanding, object detection has some significant breakthrough recent years after the widely applied deep convolutional neural networks. Before the success of convolutional neural networks, the best performing detectors in many benchmarks were based on a combination of handcrafted image feature such as HOG [3] , SIFT [17] , and the Fisher vector [2] , etc. These systems such as deformable part-based models (DPM) [6] use sliding window framework to apply classifier at every object locations and scales. Recently, the convolutional neural network based methods such as R-CNN [9, 8] bring a revolution on the field of object detection , providing a remarkable gain in detection accuracy compared to classic sliding window approaches. Conceptually, R-CNN contains two phases. First, region proposal methods are used to generate potential bounding boxes in the image. Then, a convolutional classifier is applied to each proposed bounding box.

However, the different stages in R-CNN pipeline cannot be optimized jointly, and to classify thousands of proposal bounding boxes of one image, it usually requires half a minute. The second issue has been solved in the latest incarnation of R-CNN, the Faster R-CNN [21] which shares the convolutional feature computation among different regions. Then the proposal generation becomes the new bottleneck. Recently some methods try to solve the first issue by one of the following two ways: 1) The Faster-R-CNN trains a region proposal network shared with CNN classifier; 2) YOLO [20] presents a single network end-to-end optimized directly on detection performance, and Lenc et al. [12] proposed a simplified SPP-CNN [] that does not need proposal generation. All of them focus on acceleration of testing, and developing an end-to-end framework for detection.

In this work, we focus on one question: To what extent can an one-stage CNN-based detection system perform? Although similar to YOLO [20], MultiBox [] and OverFeat [24], our system is more carefully designed for a set of specific problems such as face detection and car detection. Unlike general object detection in PASCAL VOC or ImageNet, the target objects like faces and cars could be very small but crucial to real world application (*i.e.* self-driving car). However, general proposal based detection methods could fail due to the the small resolution of objects. Our system is designed end-to-end to detect objects over all possible locations and scales in an image.

To this end, we present a novel fully convolutional neural network based object detector, called DenseBox, that does not require proposal generation and is able to be optimized end-to-end during training. We further integrate landmark localization task into the system through multi-task learning, and demonstrate that landmark localization is helpful and crucial for object detection. Experimental results show that our method results in the state-of-the-art performance on MALF(Multi-Attribute Labelled Faces) [30] detection dataset and KITTI [7] car detection dataset, suggesting that the purely fully convolutional networks for object detection can work very well when we design and train carefully.

2 Related Work

The literature on object detection is vast, and in this section we will focus on approaches exploiting class-agnostic ideas and addressing scalability.

Object detection long history.

Sliding window based detection.

R-CNN based detection.

Convolutional Neural Networks.

Finally, we capitalize on the recent advances in Deep Learning, most noticeably the work by Krizhevsky et al. [11]. We extend their bounding box regression approach for detection to the case of handling multiple objects in a scalable manner. DNN-based regression applied to object masks has been investigated by Szegedy et al. [15]. This last approach achieves state-of-art detection performance on VOC2007 but does not scale up to multiple classes due to the cost of a single mask regression: in that setup, one needs to execute 5 networks per class at inference time, which is not scalable for most real-world applications.

Several recent papers have proposed ways of using deep networks for locating class-specific or class-agnostic bounding boxes [20, 17, 3, 19]. In the OverFeat method [17], a 4-d output fc layer (for each or all classes) is trained to predict the box coordinates for the localization task (which assumes a single object). The fc layer is then turned into a conv layer for detecting multiple class-specific objects. The MultiBox methods [3, 19] generate region proposals from a network whose last fc layer simultaneously predicts multiple (e.g., 800) boxes, which are used for R-CNN [6] object detection. Their proposal network is applied on a single image or multiple large image crops (e.g., 224224) [19]. We discuss OverFeat and MultiBox in more depth later in context with our method. Shared computation of convolutions [17, 7, 2, 5] has been attracting increasing attention for efficient, yet accurate, visual recognition. The OverFeat paper [17] computes conv features from an image pyramid for classification, localization, and detection. Adaptively-sized pooling [7] on shared conv feature maps is proposed for efficient region-based object detection [7, 15] and semantic segmentation [2]. Fast R-CNN [5] enables end-to-end training of adaptive pooling on shared conv features and shows compelling accuracy and speed.

Several recent papers have proposed ways of using deep networks for locating class-specific or class-agnostic bounding boxes [20, 17, 3, 19]. In the OverFeat method [17], a 4-d output fc layer (for each or all classes) is trained to predict the box coordinates for the localization task (which assumes a single object). The fc layer is then turned into a conv layer for detecting multiple class-specific objects. The MultiBox methods [3, 19] generate region proposals from a network whose last fc layer simultaneously predicts multiple (e.g., 800) boxes, which are used for R-CNN [6] object detection. Their proposal network is applied on a single image or multiple large image crops (e.g., 224224) [19]. We discuss OverFeat and MultiBox in more depth later in context with our method. Shared computation of convolutions [17, 7, 2, 5] has been attracting increasing attention for efficient,

yet accurate, visual recognition. The OverFeat paper [17] computes conv features from an image pyramid for classification, localization, and detection. Adaptively-sized pooling [7] on shared conv feature maps is proposed for efficient region-based object detection [7, 15] and semantic segmentation [2]. Fast R-CNN [5] enables end-to-end training of adaptive pooling on shared conv features and shows compelling accuracy and speed.

The application of neural networks for detection tasks such as face detection has a long history. The first work may date back to early in 1994 when Vaillant et al.[28] proposed to train a convolutional neural network to detect face in image window. Later in 1996 and 1998 Rowley et al.[22, 23] presented many neural network based face detection system to detect upright frontal face in image pyramid. There is no way to compare the performance of those ancient detectors with todays detection systems on face detection benchmarks. Even so, they are still worth revisiting, as we find many similarities in design with our DenseBox.

Currently, most state-of-the-art object detection approaches[18, 14, 4, 8]rely on R-CNN, which divides detection into two steps: salient object proposal generation and region proposal classification. Several recent works such as YOLO and Faster R-CNN have jointed region proposal generation with classifier in one stage or two stages. It is pointed out by [5] that R-CNN with general proposal methods designed for general object detection could results in inferior performance in detection task such as face detection, due to loss recall for small-sized faces and faces in complex appearance variations. They share similarities with our method, and we will discuss them with our method in more detail in later context.

3 DenseBox for Detection

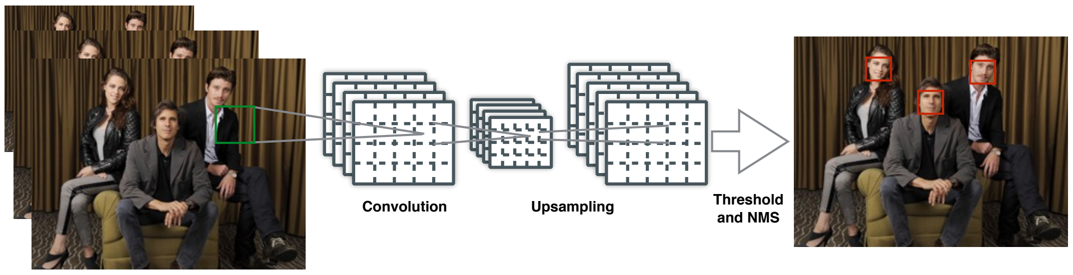


Figure 1: **The DenseBox Detection Pipeline.** 1) Image pyramid is fed to the network. 2) After several layers of convolution and pooling, upsampling feature map back and apply convolution layers to get final output. 3) Convert output feature map to bounding boxes , and apply non-maximum suppression to all bounding boxes over the threshold.

The whole detection system is illustrated in Fig 1. The single convolutional network simultaneously output multiple predicted bounding boxes and class confidence. All components of object detection in R-CNN are modeled as a fully convolutional network except the non-maximum suppression step, so region proposal generation is unnecessary. In the test, the system takes an image (at the size of $m \times n$) as input, and output a $\frac{m}{4} \times \frac{n}{4}$ feature map with 5 channels. If we define the left top and right bottom points of the target bounding box in output coordinate space as $p_t = (x_t, y_t)$ and as $p_b = (x_b, y_b)$ respectively, then each pixel i located at (x_i, y_i) in the output feature map describe a bounding box with a 5-dimensional vector $\hat{t_i} = \{\hat{s}, dx^t = \hat{x}_i - x_t,, \hat{dy^t} = y_i - y_t, \hat{dx^b} = x_i - x_b, \hat{dy^b} = y_i - y_b\}_i$, where $\hat{s}$ is the confidence score of being an object and $\hat{dx^t}$, $\hat{dy^t}$,$\hat{dx^b}$, $\hat{dy^b}$ denote the distance between output pixel location with the boundary of target bounding box. Finally every pixel in the output map is converted to bounding box with score, and non-maximum suppression is applied to those boxes whose scores pass the threshold.

3.1 Ground Truth Generation

It is unnecessary to put the whole image into the network for training because it would take most computation time in convolving on background. A wise strategy is to crop large patches containing faces and sufficient background information for training. In this paper, we train our network on single scale, and apply it to multiple scales for evaluation.

Generally speaking, our proposed network is trained in a segmentation-like way. In training, the patches are cropped and resized to 240×240 with a face in the center roughly has the height of 50 pixels. The output ground truth in training is a 5-channel map sized 60×60 , with the down-sampling factor of 4. The positive labeled region in the first channel of ground truth map is a filled circle with radius r_c, located in the center of a face bounding box. The radius r_c is proportional to the bounding box size, and its scaling factor is set to be 0.3 to the box size in output coordinate space, as show in Fig 2. The remaining 4 channels are filled with the distance between the pixel location of output map between the left top and right bottom corners of the nearest bounding box.

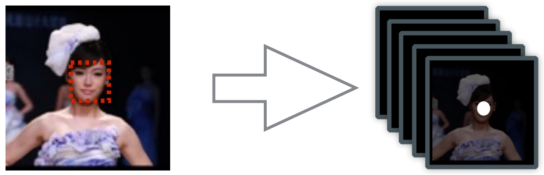


Figure 2: **The Ground Truth Map in Training .** The left image is the input patch, and the right one is its ground truth map.

Note that if multiple faces occur in one patch, we keep those faces as positive if they fall in a scale range(e.g. 0.8 to 1.25 in our setting) relative to the face in patch center. Other faces are treated as negative samples. The pixels of first channel, which denote the confidence score of class, in the ground truth map are initialized with 0, and further set to 1 if within the positive label region. We also find our ground truth generation is quite similar to the segmentation work[19] by Pinheiro et.al. In their method, the pixel label is decided by the location of object in patch, while in DenseBox, the pixel label is determined by the receptive field. Specifically, if the output pixel is labeled to 1 if it satisfies the constraint that its receptive field contains an object roughly in the center and in a given scale. Each pixel can be treated as one sample , since every 5-channel pixel describe a bounding box.

3.2 Model Design

Our network architecture illustrated in Fig 3 is derived from the VGG 19 model used for image classification[25]. The whole network has 16 convolution layers, with the first 12 convolution layers initialized by VGG 19 model. The output of conv4_4 is feed into four 1×1 convolution layers, where the first two convolution layers output 1-channel map for class score, and the second two predict the relative position of bounding box by 4-channel map. The last 1×1 convolution layers act as fully connected layers in a sliding-window fashion.

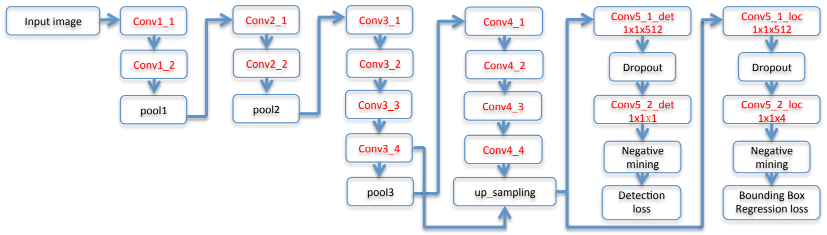


Figure 3: **Network architecture of DenseBox.** The rectangles with red names contain learnable parameters.

Multi-Level Feature Fusion. Recent works[1, 15] indicate that using features from different convolution layers can enhance performance in task such as edge detection and segmentation. Part-level feature focus on local details of object to find discriminative appearance parts, while object-level or high-level feature usually has a larger receptive field in order to recognize object. The larger receptive field also brings in context information to predict more accurate result. In our implementation, we concatenate feature map from conv3_4 and conv4_4. The receptive field (or sliding window size) of conv3_4 is 48×48, almost the same size of the face size in training, and the conv4_4 have a much larger receptive field, around 118×118 in size, which could utilize global textures and context for detection. Note that the feature map size of conv4_4 is half of the map generated by conv3_4, hence we use a bilinear up-sampling layer to transform them to the same resolution.

3.3 Multi-Task Training.

We use the ImageNet pre-trained VGG 19 network to initialize DenseBox. Actually, in initialization, we only keep the first 12 convolution layers(from conv1_1 to conv4_4), and the other layers in VGG 19 are replaced by four new convolution layers with xavier initialization.

Like Fast R-CNN, our network has two sibling output branches. The first outputs the confidence score $\hat{y}$ (per pixel in the output map) of being a target object. Given the ground truth label $y^* \in \{0, 1\}$, the classification loss can be defined as follows.

$$\mathcal{L}_{cls}(\hat{y}, y^*) = \|\hat{y} - y^*\|^2 \tag{1}$$

Here we use $L2$ loss in both face and car detection task. We did not try other loss functions such as hinge loss or cross-entropy loss, which seems to be a more appropriate choice, as we find the simple $L2$ loss work well in our task.

The second branch of outputs the bounding-box regression loss, denoted as $\mathcal{L}_{loc}$. It targets on minimizing the $L2$ loss between the predicted location offsets $\hat{d} = (\hat{d}_{tx}, \hat{d}_{ty}, \hat{d}_{tx}, \hat{d}_{ty})$ and the targets $d^* = (d^*_{tx}, d^*_{ty}, d^*_{tx}, d^*_{ty})$, as formulized by:

$$\mathcal{L}_{loc}(\hat{d}, d^*) = \sum_{i \in \{tx, ty, bx, by\}} \left\| \hat{d}_i - d^*_i \right\|^2 \tag{2}$$

3.3.1 Balance Sampling.

The process of selecting negative samples is one of the crucial parts in learning. If simply using all negative samples in a mini-batch will bias prediction towards negative samples as they dominate in all samples. In addition, the detector will degrade if we penalize loss on those samples lying in the margin of positive and negative region. Here we use a binary mask for each output pixel to indicate whether it is selected in training.

Ignoring Gray Zone. The gray zone is defined on the margin of positive and negative region. It should not be considered to be positive or negative, and its loss weight should be set to 0. For each non-positive labeled pixel in the output coordinate space, its ignore flag f_{ign} is set to 1 only if there is any pixel with positive label within $r_{near} = 2$ pixel length.

Hard Negative Mining. Analogous to hard-negative mining procedure in SVM, we make learning more efficient by searching the badly predicted samples rather than random samples. After negative mining, the badly predicted samples are very likely to be selected, so that gradient descent learning on those samples leads more robust prediction with less noise. Specifically, negative mining can be performed efficiently by online bootstrap. In the forward propagation phase, we sort the loss (Eq 1) of output pixels in decending order, and assign the top 1% to be hard-negative. In all experiments, we keep all positive labeled pixels(samples) and the ratio of positive and negative to be 1:1. Among all negative samples, half of them are sampled from hard-negative samples, and the remaining half are selected randomly from non-hard negative. For convenience, we set a flag $f_{sel} = 1$ to those pixels (samples) selected in a mini-batch.

Loss with Mask. Now we can define the mask $M(\hat{t}_i)$ for each sample $\hat{t}_i = \{\hat{y}_i, \hat{d}_i\}$ as a function of flags mentioned above:

$$M(\hat{t}_i) = \begin{cases} 0 & f^i_{ign} = 1 \text{ or } f^i_{sel} = 0 \\ 1 & \text{otherwise} \end{cases} \tag{3}$$

Then if we combine the classification (Eq 1) and bounding box regression (Eq 2) loss with masks, our full multi-task loss can be represented as ,

$$\mathcal{L}_{det}(\theta) = \sum_i \Big(M(\hat{t}_i)\mathcal{L}_{cls}(\hat{y}_i, y^*_i) + \lambda_{loc}[y^*_i > 0] M(\hat{t}_i)\mathcal{L}_{loc}(\hat{d}_i, d^*_i) \Big) \tag{4}$$

where θ is the set of parameters in the network, and the Iverson bracket function $[y_i^* > 0]$ is activated only if the ground truth score y_i^* is positive. It is obvious that the bounding box regression loss should be ignored for negative samples (background), since there is no notation for them. The balance between classification and regression tasks is controlled by the parameter λ_{loc}. In our experiments, we normalize the regression target d^* by dividing by the standard object height, which is $50/4$ in ground truth map, and $\lambda_{loc} = 3$ works well in all experiments under this normalization.

Other Implementation Details. In training, an input patch is considered to be "positive patch" if it contains an object centered in the center at a specific scale. These patches only contain negative samples around the positive samples. To fully explore the negative samples in the whole dataset, we also randomly crop patches at random scale from training images, and resize them to the same size and feed them to the network. We call this kind of patch as "random patch", and the ratio of "positive patch" and "random patch" in training is 1:1. In addition, to further increase the robustness of our model, we also randomly jitter every patch before feeding them into the network. Specifically, we apply left-right flip, translation shift (of 25 pixels), and scale deformation (from $[0.8, 1.25]$).

We use mini-batch SGD in training and the batch size is set to 10. The loss and output gradients must be scaled by the number of contributing pixels, so that both loss and output gradients are comparable in multi-task learning. The global learning rate starts with 0.001, and it is reduced by a factor of 10 at every 100K iterations. We follow the default momentum term weight 0.9 and the weight decay factor 0.0005.

3.4 Refine with Landmark Localization.

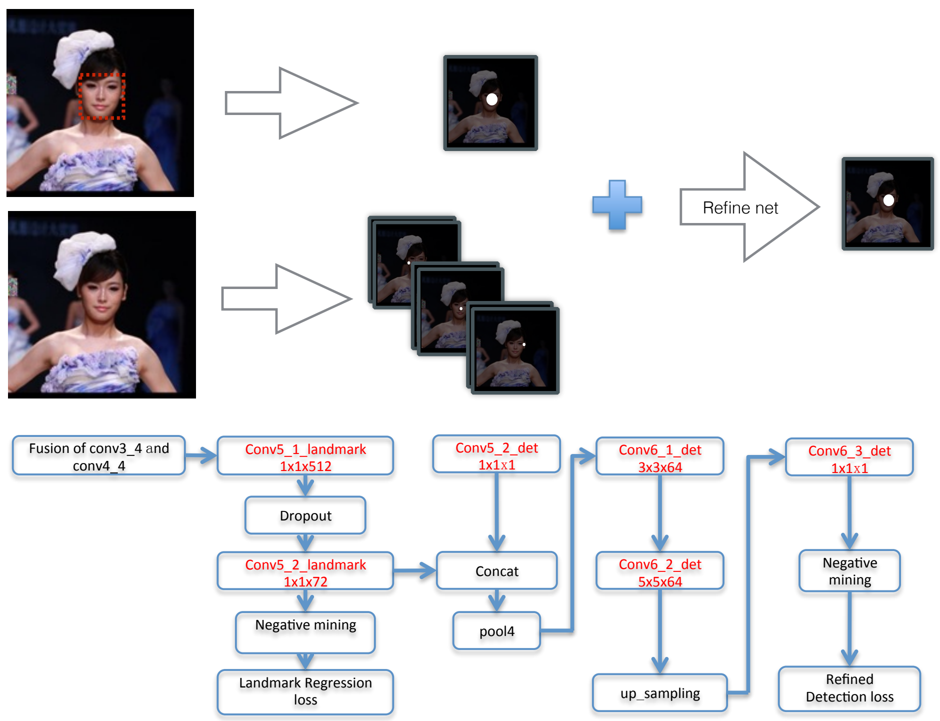


Figure 4: **Top:** The pipeline of DenseBox with landmark localization. **Bottom:** The network structure for landmark localization.

In this part, we show that landmark localization can be achieved in DenseBox just by stacking a few layers owe to the fully convolution architecture. Moreover, we can refine detection results through the fusion of landmark heatmaps and face score map. As shown in Fig 4, we incorporate another sibling branch output for landmark localization. Suppose there are N landmarks, the landmark localization branch outputs N response maps, with each pixel represent the confidence score of being a landmark at that location. The appearance of ground-truth maps used for this task is quite similar to the ground-truth for detection. For a landmark instance l_i^k, the ith instance of landmark k, its ground-truth is a positive labeled region located at the corresponding location on the kth response map in the output coordinate space. Note that the radius r_l should be relative small (e.g. $r_l = 1$)to prevent loss of accuracy. Similar to classification task, the landmark localization loss $\mathcal{L}_{lm}$is defined as a $L2$ loss between predicted values and labels, and we still apply the negative mining and ignore region discussed in the previous section.

The final output refine branch, taking the classification score map and landmark localization maps as input, targets on refine of the detection results. An appropriate solution could be using high-level spatial model to learn the constraints of landmark confidence and bounding box score, to further increase the performance of detections. Tompson et.al.[27] proposed a MRF-like model using modified convolution (SoftPlus convolution) with non-negative output to connect the distribution of spatial location for each body part. However, their model also include Log and Exp stages, which make model difficult to train. In our implementation, we use convolutions with ReLU activation to approximate the spatial model. If we denote the refine detection loss as $\mathcal{L}_{rf}$, which is almost the same as the classification loss $\mathcal{L}_{cls}$ mentioned before but the predict map is from the refine branch, the full loss becomes as ,

$$\mathcal{L}_{full}(\theta) = \lambda_{det}\mathcal{L}_{det}(\theta) + \lambda_{lm}\mathcal{L}_{lm}(\theta) + \mathcal{L}_{rf}(\theta) \tag{5}$$

where λ_{det} and λ_{lm} controll the balance of the three tasks. They are assigned to 1 and 0.5 respectively in our experiments.

3.5 Comparison

The highlight of DenseBox is that it frames object detection as a regression problem and provides an end-to-end detection framework. Several recent works such as YOLO and Faster R-CNN have jointed region proposal generation with classifier together. Here we compare DenseBox to other related detection systems, pointing out the key similarities and differences.

Traditional NN-based Face Detector. The neural network-based face detectors refer to those face detection system using neural network before the recent break-through results of CNNs for image classification. Applying neural networks for face detection has a long history, and the early works date back to 1990s[28]. Rowley et al.[22] train neural network-based detectors which only is activated on faces with specific size, and apply detectors on the image pyramid with sliding-window fashion. Our DenseBox is very similar to them in the detection pipeline, excepting that we use modern CNNs as detectors. Hence the DenseBox could be called as " Modern NN-based detector in one sense.

OverFeat. OverFeat[24] designed by Sermanet et al. might be the first work that train a convolution neural network to perform classification and localization together after the success application of deep CNNs for image classification[11]. It also apply fully convolutional network on test time, an equivalent but much efficient way to perform sliding window detection. However it still disjoints classification and localization in training, and need complex post-processing to produce detection results. Our method is very similar to OverFeat but a multi-task jointly learned end-to-end detection network.

Deep Dense Face Detector (DDFD) The DDFD, psoposed by Farfade et.al.[5], is a face detection system based on convolutional neural networks. It claims to have superior performance over R-CNN on face detection task due to the reason that proposal generation in R-CNN may miss some face regions. Although the DDFD is a complete detection pipeline, the DDFD is not an end-to-end framework since it separate the class probability prediction and bounding box localization as two tasks and two stages. Our DenseBox can be optimized directly for detection , and can be easily improved by incorporating landmark information.

Faster R-CNN. The faster R-CNN[21] still use region proposals to find objects in an image. Unlike the its former variants, the region proposals in faster R-CNN is produced by region proposal networks(RPNs) sharing convolutional feature computation with classifiers in the second stage. The PRN shares many similarities with our method DenseBox. However, The PRN needs predefined anchors while ours does not. The PRN is trained on multi-scale objects while the DenseBox presented in this paper is trained on one scale with jitter-augmentation, which means our method need to evaluate feature at multiple scales. Moreover, the training schemes are quite different between DenseBox and PRN.

MultiBox. The MultiBox[4] trains a convolutional neural network to generate proposals instead of selective search. Both DenseBox and MultiBox are trained to predict bounding boxes in an image, they generate bounding boxes in different way. As compared in [21] , the MultiBox method generates 800 non-translation-invariant anchors, whereas our DenseBox output translation-invariant bounding boxes like RPN. As the down-sampling factor of output map is 4, DenseBox will densely generate one bounding box with score at every 4 pixels.

YOLO. Redmon et al.[20] propose a unified object detection pipeline, called YOLO. Both DenseBox and YOLO can be trained end-to-end from images, but the model design differs in the output layers. The YOLO system takes a 448×448 image as input, and outputs 7×7 grid cells, only 49 bounding boxes per image. Our DenseBox uses up-sampling layers to keep a relative high-resolution output, with a down-sampling scale factor of 4 in our model. This enables our network capable to detect very small objects and highly overlapped objects, which YOLO is unable to deal with.

4 Experiments

In this section, we demonstrate the performance of DenseBox on MALF(Multi-Attribute Labelled Faces) dataset[30] and KITTI[7] car detection task. We also evaluate our method on those tasks with or without the help of landmark annotation, showing that multi-task learning with landmark localization can significantly boost the performance. We compare our results with current the state-of-the-art systems, which shows that our method achieves competitive results on object detection tasks. Nothe that we do not compare the performances of our DenseBox with original R-CNN directly on those task, but we highlight the performances of other methods which claim to use R-CNN or those methods have alrealy compared themselves to R-CNN.

4.1 MALF Detection Task

The MALF detection test dataset contains 5,000 images in collected from the Internet. Unlike the widely used FDDB[10] face detection benchmark, which is collected from news photos and the pose tends to be frontal, the face images in MALF have much larger diversity, making it closer to real world application than FDDB.

Training and Testing. We train two models described in section 3 on 31,337 Internet-collected images with 81,024 faces annotated with 72 landmarks illustrated in Fig 5. One model only use bounding box information, while the other model utilize both bounding box and landmark information for comparison. They are both initialized with ImageNet pre-trained VGG19 model. The faces in training are roughly scaled to 50 pixels in height, and the scale jitter range is $[0.8, 1.25]$, the same as described in section 3.3. On testing, we first selectively down sample images so that for each image the longest image side does not exceed 800 pixels. Then we test our model on each image at several scales. The test scale starts from 2^{-3} to $2^{1.2}$ with the step of $2^{0.3}$. This setting enable our models to detect faces from 20 pixels to 400 pixels in height. The non-maximum suppression IOU threshold in face detection is set to 0.5. Under this configuration, it taks several seconds to process one image in MALF dataset on an Nvidia K40 GPU. **Results.** We illustrate the results of three versions of DenseBox on MALF dataset. The “DenseBoxNoLandmark denotes DenseBox without landmark in training. “DenseBoxLandmark is the model incorporating landmark localization, and “DenseBoxEnsemble is the result of ensembling 10 DenseBox with landmarks from different batch iterations. As shown in Fig 6, landmark localization gives a significant performance boost on face detection. We also notice that the models trained with different batch iterations still have high diversity since another significant boost has been seen by model ensemble. Then we compare our best model with other state-of-the-art methods on MALF. Surprisingly, our model achieves the best performance, with mean recall rate of 87.26%, almost outperform DDFD by 10%, which claims to have better performance than R-CNN on face detection task.

4.2 KITTI Car Detection Task

The KITTI object detection benchmark consists of 7481 training images and 7518 test images. The total number of objects in training sums up to 51,867, in which cars only accounts for 28,742. The key difficulty of KITTI car detection task is that a great amount of cars are in small size (height ¡ 40

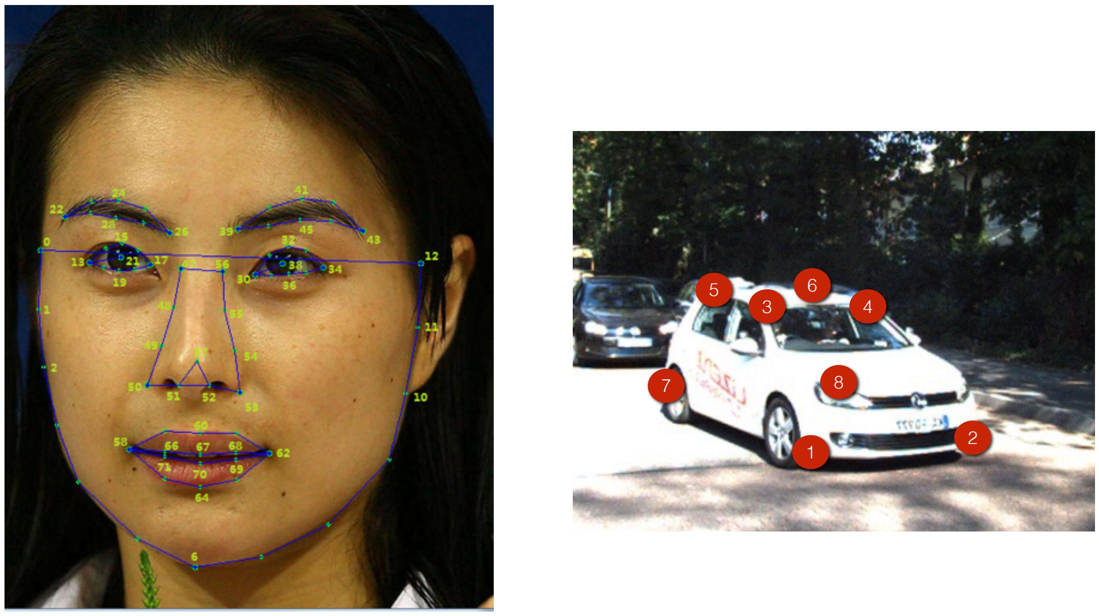
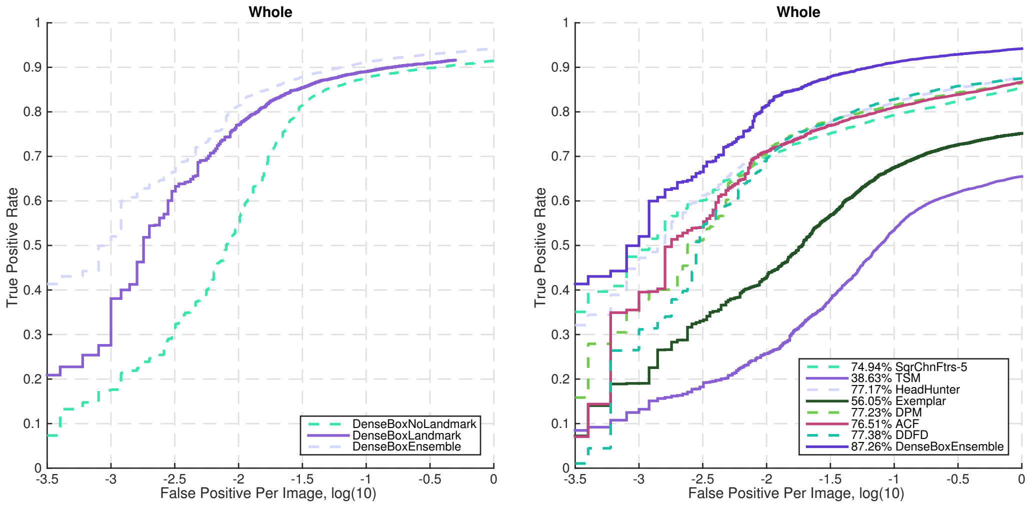


Figure 5: **Left:** 72 landmarks for face. **Right:** 8 landmarks for car.



(a) (b)

Figure 6: **Result on MALF dataset.** (a) Comparison of different versions of DenseBox; (b)The curves and mean recall rate of DenseBox and other methods;

pixels) and occluded. To overcome this difficulty, previous works such as[13] need careful part and occlusion modeling.

Training and Testing. As well as in face detection task, we train two models(one without landmark, the other with landmark) on the KITTI object detection training set. Since KITTI does not provide landmarks for car, we selectively annotate 8 landmarks for large cars (height ¿ 50 pixels). The landmarks of car is shown in Fig 5, and we finally annotate 7790 cars, roughly 27% of the total cars. The testing procedure is the same as in face detection, except that we do not down sample car images. The evaluation metric of KITTI car detection task is different from general objecte detection. KITTI requires an overlap of 70% for true positive bounding box, while other tasks such as face detection only requires 50% overlap. This strict criteria requests high accurate car localization. On KITTI, we set the non-maximum suppression IOU threshold to 0.75.

Results.

Method	*Moderate*	*Easy*	*Hard*
Regionlets [16]	76.45%	84.75%	59.70%
AOG [13]	74.26%	84.24%	60.51%
3DVP [29]	75.77%	87.46%	65.38%
spCov_LBP	77.40%	87.19%	60.60%
DeepInsight	84.40%	84.59%	76.09%
NIPS ID 331	87.14%	88.33%	76.11%
DJML	88.79%	91.31%	77.73%
DenseBox (without landmark)	85.07%	82.33%	76.27%
DenseBox (withlandmark)	85.74%	83.63%	76.71%

Table 1: The Averate Presision on KITTI Car Detection Task

Results. Table 1 shows the results of DenseBox and other methods. We can see that partially annotated landmark information (27%) still can boost detection performance. On average, model with landmark localization slightly outperforms no-landmark model by 0.9% in average precision. The promotion on performance is not as great as in face detection. The reason could be that the landmark information is not sufficient. The insufficient lies on both the amount (27% in car while 100% in face) and the quality (8 landmarks in car whereas 74 in face). Compared with other methods, the DenseBox still achieves competitive results. DenseBox defeats traditional detection system such as Regionlets and spCov by a large margin. Our average precision on moderate car is 85.74%, slightly better than DeepInsight, which use R-CNN framework with ImageNet pre-trained GoogLeNet[26]. Our model has been ranked as the top 1 for 4 months until an anonymous submission titled "NIPS ID 331, which use stereo information for training and testing. Recently a method named "DJML overtakes all other methods.

5 Conclusion

We have presented the DenseBox, a unified end-to-end detection pipeline for detection. The performance can be boosted easily by incorporating landmark information. We also analysis our method and other related object detection system, highlighting the difference and the contribution of DenseBox. The DenseBox achieves impressive performance on both face detection and car detection task, demonstrating its high suitable for situation where proposal generation might fail. The key problem of DenseBox is the speed. The original DenseBox presented in this paper needs several seconds to process one image. But this has been addressed by now. We will present another paper describing a real-time detection system on KITTI and face detection, called DenseBox2.

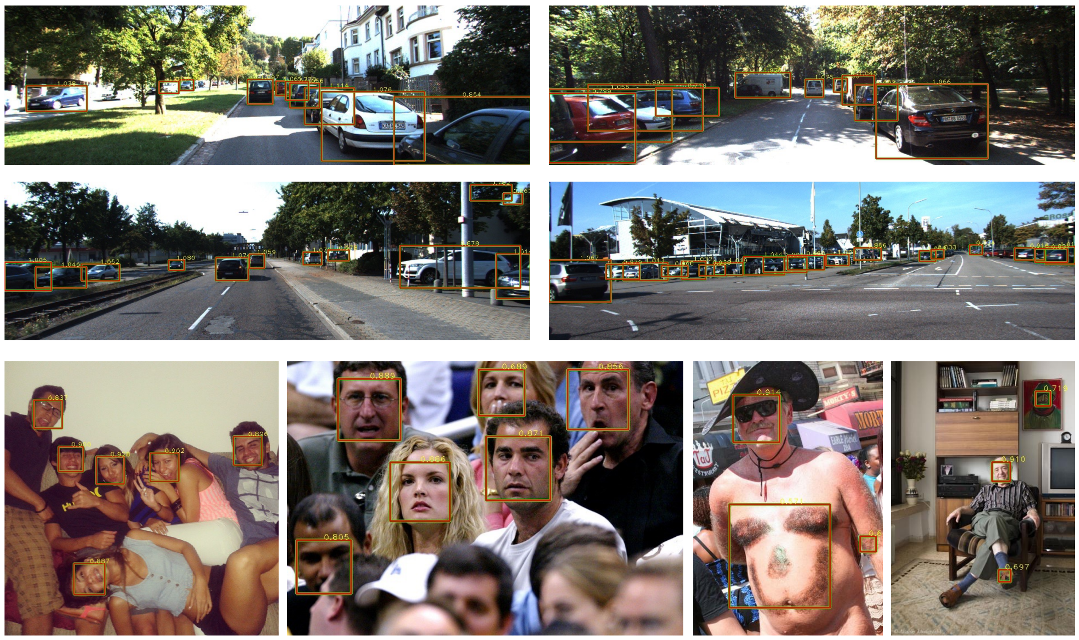


Figure 7: **Examples on both the MALF detection set KITTI car detection set.** The numbers above the bounding boxes are the confidence score. Our system works very well in complex scene where objects are small and highly occluded. However, it still could miss some objects and generate false alarm.

References

[1] G. Bertasius, J. Shi, and L. Torresani. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. *arXiv preprint arXiv:1412.1123*, 2014.

[2] R. G. Cinbis, J. Verbeek, and C. Schmid. Segmentation driven object detection with fisher vectors. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2968–2975. IEEE, 2013.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[4] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 2155–2162. IEEE, 2014.

[5] S. S. Farfade, M. Saberian, and L.-J. Li. Multi-view face detection using deep convolutional neural networks. *arXiv preprint arXiv:1502.02766*, 2015.

[6] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010.

[7] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[8] R. Girshick. Fast r-cnn. *arXiv preprint arXiv:1504.08083*, 2015.

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587. IEEE, 2014.

[10] V. Jain and E. G. Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. *UMass Amherst Technical Report*, 2010.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[12] K. Lenc and A. Vedaldi. R-CNN minus R. *arXiv preprint*, abs/1506.06981, 2015.

[13] B. Li, T. Wu, and S.-C. Zhu. Integrating context and occlusion for car detection by hierarchical and-or model. In *Computer Vision–ECCV 2014*, pages 652–667. Springer, 2014.

[14] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A convolutional neural network cascade for face detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5325–5334, 2015.

[15] W. Liu, A. Rabinovich, and A. C. Berg. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*, 2015.

[16] C. Long, X. Wang, G. Hua, M. Yang, and Y. Lin. Accurate object detection with location relaxation and regionlets re-localization. In *Computer Vision–ACCV 2014*, pages 260–275. Springer, 2015.

[17] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[18] W. Ouyang, P. Luo, X. Zeng, S. Qiu, Y. Tian, H. Li, S. Yang, Z. Wang, Y. Xiong, C. Qian, et al. Deepid-net: multi-stage and deformable deep convolutional neural networks for object detection. *arXiv preprint arXiv:1409.3505*, 2014.

[19] P. O. Pinheiro, R. Collobert, and P. Dollar. Learning to segment object candidates. *arXiv preprint arXiv:1506.06204*, 2015.

[20] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *arXiv preprint*, abs/1506.02640, 2015.

[21] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015.

[22] H. Rowley, S. Baluja, T. Kanade, et al. Neural network-based face detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(1):23–38, 1998.

[23] H. Rowley, S. Baluja, T. Kanade, et al. Rotation invariant neural network-based face detection. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pages 38–44. IEEE, 1998.

[24] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[26] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

[27] J. J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in Neural Information Processing Systems*, pages 1799–1807, 2014.

[28] R. Vaillant, C. Monrocq, and Y. Le Cun. Original approach for the localisation of objects in images. *IEE Proceedings-Vision, Image and Signal Processing*, 141(4):245–250, 1994.

[29] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. Data-driven 3d voxel patterns for object category recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1903–1911, 2015.

[30] B. Yang, J. Yan, Z. Lei, and S. Z. Li. Fine-grained evaluation on face detection in the wild. In *Automatic Face and Gesture Recognition (FG), 11th IEEE International Conference on*. IEEE, 2015.