https://doi.org/10.2352/ISSN.2470-1173.2017.10.IMAWM-163

Training Object Detection And Recognition CNN Models Using Data Augmentation

Daniel Mas Montserrat; School of Electrical and Computer Engineering; Purdue University; West Lafayette, Indiana, USA
Qian Lin; HP Labs, HP Inc; Palo Alto, California, USA
Jan Allebach; School of Electrical and Computer Engineering; Purdue University; West Lafayette, Indiana, USA
Edward J. Delp; School of Electrical and Computer Engineering; Purdue University; West Lafayette, Indiana, USA

Abstract

Recent progress in deep learning methods has shown that key steps in object detection and recognition, including feature extraction, region proposals, and classification, can be done using Convolutional Neural Networks (CNN) with high accuracy. However, the use of CNNs for object detection and recognition has significant technical challenges that still need to be addressed. One of the most daunting problems is the very large number of training images required for each class/label. One way to address this problem is through the use of data augmentation methods where linear and nonlinear transforms are done on the training data to create "new" training images. Typical transformations include spatial flipping, warping and other deformations. An important concept of data augmentation is that the deformations applied to the labeled training images do not change the semantic meaning of the classes/labels. In this paper we investigate several approaches to data augmentation. First, several data augmentation techniques are used to increase the size of the training dataset. Then, a Faster R-CNN is trained with the augmented dataset for detect and recognize objects. Our work is focused on two different scenarios: detecting objects in the wild (i.e. commercial logos) and detecting objects captured using a camera mounted on a computer system (i.e. toy animals).

Introduction

Object detection and recognition is one of the most important areas in computer vision since it is a key step for many applications including smart home, smart office, surveillance and robotics. In this paper, we focus on two different scenarios: detecting commercial logos in the wild and detecting objects captured by a high definition camera (i.e. toys). Logo detection is an important task in contextual ad placement (placing relevant ads on webpages, images, and videos), validation of product placement, and online brand management [1]. Object detection can be used as part of educational or entertainment applications providing a richer interaction with the user.

Our work makes use of deep learning methods, in particular the Faster R-CNN (Region-based Convolutional Neural Network) [2]. The network is composed of three main parts: a feature extractor, a region proposal network, and a classifier. The network allows us to detect multiple objects in a scene. This CNN is described in more detail later.

Deep learning methods, such as Faster R-CNN, require approximately 5,000 samples per class [3] to have good performance. This can be a problem in many situations, where only one or few images of the object that we want to detect are available. One way to address this problem is through the use of data augmentation methods where linear and nonlinear transforms are done on the training data to create "new" or synthetic training images. Typical transformations include spatial flipping, warping and other deformations. An important concept of data augmentation is that the deformations applied to the labeled training images do not change the semantic meaning of the labels.

In this paper, we propose a solution to the lack of training data by generating synthetic data using data augmentation methods. The synthetic image data is generated using transformations such as rotations and color changes, and blending them into background images [4].

The main contribution of this paper is combining the use of data augmentation techniques and the Faster R-CNN for object detection with almost non-existent training samples. This technique is used for logo detection in the wild and object detection with multiple toys in various poses.

This paper is organized as follows, in Section 2, we present an overview of related work in object recognition, logo recognition and data augmentation. In Section 3, we propose several techniques for data augmentation and we provide an overview of the datasets used for our experiments. In Section 4, we present the experimental evaluation. In Section 5, we analyze the network using visualization tools. We conclude in Section 6, by presenting conclusions and future improvements.

Overview Of Related Work

Traditionally, the use of hand crafted features, such as SIFT [5] and textures [6], along with statistical classifiers, such as Support Vector Machines (SVM) [7] and Nearest Neighbor (NN) [8], have been the main approaches for object detection and classification.

In the last several years deep learning methods has shown to provide higher accuracy compared to traditional approaches [2, 9, 10, 11, 12]. This improvement has been possible mainly by advances in hardware (e.g. more powerful GPUs) and the availability of large labeled datasets (e.g. ImageNet [9] contains more than 14M images). Deep learning methods have demonstrated impressive results in speech recognition, object recognition and detection and in other domains such as drug discovery and genomics [13, 14]. Deep learning based methods are the leading approaches in object classification (i.e. ImageNet [9]) and object detection competitions (i.e. Pascal VOC [15] and MS COCO [16]).

One deep learning approach that has achieved high accuracies in classification and detection is the Convolutional Neural Network (CNN) [13, 3, 12]. This network combines convolu-

tional filter layers and non-linearity layers to learn and extract features (feature extraction subnet) and fully-connected layers to classify them (decision subnet) [17, 12].

The feature extraction subnet can contain many convolutional layers and each layer contains multiple filters. The filters of the first convolutional layers are able to detect simple features such as color or edges. Filters in deeper layers learn more complex features (e.g. some layers can detect complex shapes such as faces, wheels, and animals). Between convolutional layers, non-linear layers such as the Rectified Linear Unit (ReLU) or Max-pooling are included. ReLU layers compute the maximum between 0 and their input value. Max-pooling layers perform a non-linear down-sampling. The down-sampling process consists of partitioning the input image into non-overlapping rectangles and selecting the maximum value inside each rectangle. The outputs of the convolutional layers are usually known as feature or activation maps.

The decision subnet can contain multiple fully connected layers. Fully connected layers consist of a set of matrix multiplications followed by non-linear operations (typically a ReLU). The size of the last fully connected layer output is equivalent to the number of classes. The network outputs a probability or confidence value for each class. The decision subnet usually requires a fixed input size. This issue is addressed by initially cropping or resizing the input images before the feature extraction subnet.

The weights and parameters of the convolutional and fully connected layers are learned from training samples using Backpropagation [13] in combination with gradient-based methods such as Stochastic Gradient Descent (SGD) [14]. The learning process starts by assigning random values to the weights and parameters of the network. Then, two different stages, propagation and weight update, are repeated over a fixed number of iterations. First, an input image is propagated forward through the network until it reaches the output layer. Then, the output of the last layer is compared with the ground truth value using a loss function. The loss function is a function that generates an error measure. If the predicted output is close to the desired output, the error measure will be small. If the predicted output differs a lot from the desired one, the error will be large. The error value is backpropagated through the whole network using SGD. The SGD is an optimization method that updates the values of the weights and parameters of the network in order to minimize the loss function. The loss function gives a measure of the training error.

The training error represents how well the network fits the training data. Typically, the training error underestimates the testing error. Testing error is the error that results when the network is used for a new observation that was not used in the training process. If the gap between test and training error is large, we say that the network is overfitting. That means that the network has learned the training data but is not able to generalize to new examples.

A common practice in deep learning is to train a CNN with a large generic dataset (e.g. ImageNet) and then use the weights and parameters obtained in the training process as an initialization. This process is known as fine-tuning.

In our work, we make use of two common CNN models: the Zeiler & Fergus (ZF) net [18] and the VGG16 (Visual Geometry Group) net [19]. The ZF Network has 5 pairs of convolutional and ReLU layers followed by 2 fully connected layers. The 1st convolutional layer has 96 filters with size 7×7, the 2nd convolutional layer has 256 filters of size 5×5, the 3rd, 4th and 5th convolutional layers have 256, 384 and 384 filters respectively. Each filter has a size of 3×3. VGG16 is a deeper model containing 5 sets of layers. Each set contains two convolutional and ReLU layers followed a max pooling layer. The number of filters in the convolutional layers are 65, 128, 256, 512 and 512 respectively. All the filters have a size of 3×3. VGG16 ends with 3 fully connected layers.

CNN are good for image classification but they can not localize objects inside the image. The Region-Based Convolutional Neural Network (R-CNN) [20] is a network able to locate and classify several objects in images of any size by combining CNNs and external region proposal methods. A region proposal method is a method that finds a set of regions, typically defined with bounding boxes, that might contain objects of interest. Typical region proposal methods are Selective Search [21] and EdgeBoxes [22]. Selective Search splits the image in several regions of interest by using similarity measures based on color and visual features like SIFT [5]. EdgeBoxes finds regions of interest using object contours information. In the R-CNN, each region of interest is cropped and resized to 227×277 pixels. Then, the resized image is used as input of a CNN consisting of five convolutional layers and two fully connected layers. The CNN assigns to each region of interest a class and a confidence score.

The R-CNN, and many other object detection methods, processes the bounding boxes generated by region proposals methods using Non-Maximum Suppression (NMS) [20]. NMS rejects a bounding box if it has a large overlap with another bounding box with higher confidence. If the overlap is higher than a threshold, the bounding box is rejected. Typically, the threshold is a parameter learned by the network in the training process.

The main problem with R-CNN is that it is computationally complex since every image is processed as many times as regions of interest are detected. Previous work such as the Spatial Pyramid Pooling Network (SPPnet) [10] address this problem by using pooling. SPPnet starts with a CNN (e.g. VGG16 or ZF) followed by a spatial pyramid pooling layer and fully connected layers. The spatial pyramid pooling layer uses max-pooling for each region of interest using grids with multiple sizes. The regions of interest are computed using Selective Search. The images are processed only one time by the CNN. Then, spatial pyramid pooling is use to generate the output of the last convolutional layer, the feature map, and is later classified by the fully connected layer.

The Fast R-CNN [11] is a network with the same structure as SPPnet but substitutes the spatial pyramid pooling with a RoI (Region of Interest) pooling layer. The RoI pooling layer is a simplified version of the spatial pyramid pooling, where instead of using a grid with multiple sizes, only one size (tipically 7×7) is used. Fast R-CNN also introduces a more effective method for training the CNN and adds a bounding box regressor. The network is trained using multiple regions per image instead of using only one as it is done in SPPnet. The bounding box regressor is a layer that outputs the locations of bounding boxes where objects of interest might be located.

In our work, we use a variant of the Fast R-CNN known as the "Faster R-CNN" which combines the Fast R-CNN with a RPN (Region Proposal Network). The RPN is a neural network that uses the output of the last convolutional layer of the CNN,

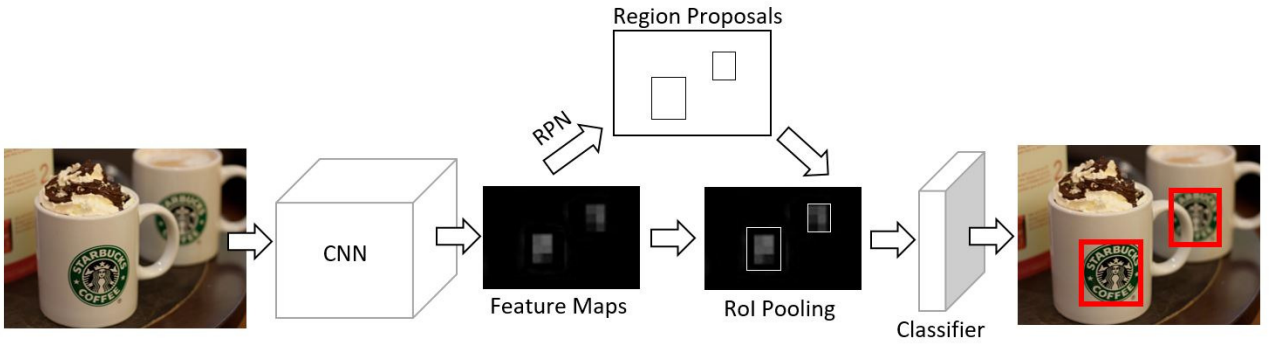

Figure 1. *Faster R-CNN is a network that combines a Convolutional Neural Network, a Region Proposal Network, a Region of Interest Pooling layer, and a classifier*

the feature map, to generate regions of interest. The RPN consist of a 3×3 sliding window that outputs a set of 9 bounding boxes containing regions of interest. Each bounding box has a different size and a different aspect ratio. A fully connected layer assigns a binary class (foreground or background) to each bounding box. Following the steps in the Fast R-CNN, each region of interest is applied to the RoI pooling layer and is later classified by a fully connected layer. In the classification step, a confidence score is assigned to each bounding box. The confidence value ranges from 0 to 1 where a confidence of 1 represents that the network is almost certain that the class assigned is correct. A threshold is usually set to 0.7 in a deployment stage and all the bounding boxes below the threshold are discarded. The network can be trained end-to-end and provides an almost real time performance. With the addition of RPN, there is no need to use external region proposals methods. Figure 1 shows the structure of the Faster R-CNN.

The Faster R-CNN is the basis of several 1st-place entries in the ImageNet and MS COCO competitions [2]. It is also used in commercial systems such as Pinterest [23].

Other methods for image detection such as You Only Look Once (YOLO) [24] provide real-time performance by compromising accuracy. Recent methods such as Single Shot Multibox Detector (SSD) [25] provide real time performance and good accuracy but seem to perform poorly detecting small objects since it resizes the input images to a size of 300×300 pixels and resolution is lost. Both methods divide the image into a fixed number regions and predicts bounding boxes and probabilities for each region using fully connected layers.

Several methods have been proposed for logo detection and recognition using both hand crafted visual features [26] and deep learning [1, 27]. The work presented in [1] makes use of CNNs for logo classification and the Fast R-CNN for logo detection with and without localization. They report a mean average precision of 74.4% using Fast R-CNN with the VGG16 [19] architecture and selective search as region proposal method. In this paper, we show that the Faster R-CNN combined with data augmentation produces significant improvements in logo detection with localization.

Data augmentation has been used for object detection using hand crafted visual features [26, 28] and for deep learning [4]. Typical data augmentation techniques used in deep learning include image cropping, flipping and color changes [9] to create the augmented or synthesized images. More complex techniques can include noise addition, geometric transformations, or image compression. The method presented in [28] combines multiple transformations to the training set. After the data augmentation process an accuracy increase of 3.5% in 2010 ImageNet competition was reported.

Synthesized images have been used for training neural networks for self-driving vehicles [29] and text recognition applications [30] and have demonstrated encouraging results. The method presented in [29] makes use of 3D virtual worlds to train a neural network. The use of virtual worlds has proved to be effective when using reinforced learning [31].

Other work has used neural networks to generate new data. The work presented in [30] uses a neural network to estimate the depth of images used as background images. Text is then added to uniform regions of the background images using the depth information. Methods such as [32] use Recurrent Neural Networks (RNN) to generate new training samples using information extracted from a training dataset.

Our Proposed Approach

In this section we describe our method for synthesizing images. Our approach is based on the work described in [4, 28] which blends images of objects with real-world background images. The images of objects and logos undergo several transformations as described in the following sections. Images of objects are essential to data augmentation. For logo detection, the images are extracted from the FlickrLogos-32 [26] dataset. There are several public datasets available containing labeled images with logos, these include FlickrLogos-32[26], FlickrLogos-27 [33], BelgaLogos [34] and MICC-Logos. We selected FlickrLogos-32 for our training and evaluation purposes because it contains the largest number of labeled images and is the one commonly used in previous works of logos in the wild detection.

FlickrLogos-32 consists of 32 different brands (classes) each with various versions. The dataset contains 8240 images mined from Flickr [35]. Figure 2 shows samples from the FlickrLogos-32 dataset. The dataset is divided into training and testing parts. Training data is comprised of 1280 images (40 per class) con-

taining genuine logos and 3000 images with no logo content. The 3000 images are used as background images (distractors). Testing data includes 960 (30 per class) images containing genuine logos and 3000 background images. Each image with a genuine logo is provided with the true class and a binary segmentation mask indicating the logo location.

In the case of object detection, high quality images are captured using a high-definition camera contained in the HP Sprout [1]. The HP Sprout is a desktop computer released in November 2014 that also has a projector, a HD camera, a 3D camera, a touch mat and a LED desk lamp [36].

Logo Extraction

Figure 2. *Image samples from FlickrLogos-32. The logos are for Adidas (left), Corona (center) and Starbucks (right)*

Logo images are extracted from the FlickrLogos-32 training set using the binary segmentation masks and labels provided. Each image may include more than one logo. In total, we obtain 60-80 images per brand. Images smaller than 20×20 pixels are discarded since they are very difficult to detect after data augmentation. Figure 3 shows examples of logos from the FlickrLogos-32 dataset.

Figure 3. *Six different logos from FlickrLogos-32*

The same brand can have different versions of logos. Figure 4 shows the inter-class variation of three different brands. We do not make distinction between logo versions and we assign one class per brand.

Object Capture

Total of 20 high quality images are captured for every object in a different pose. The HP Sprout is used in the image acquisition process. The images are captured using the top HD camera with white light projected to the touch mat and the LED desk lamp turned off. In this paper, no 3D information is captured or used.

Various poses are used, Figure 5 shows six examples of the same object. As presented in following sections, the number of poses used in the data augmentation process will affect the detection and precision performance. Intuitively, if more poses are

[1]HP Sprout, HP Inc ®

Figure 4. *Different logo versions for Fedex (left), Apple (center) and Google (right)*

available for training, the network will be more resistant to rotations and change of poses. In this paper, a set of 15 different toys was used. Figure 6 shows examples of different toys. Some figures have minor differences between them (set of small red and black toys). Despite that, the network is able to differentiate them as presented in next sections.

Figure 5. *Captures of different poses*

Figure 6. *Six different toys*

Synthetic Data Generation

We want to generate real world looking images containing the objects of interest (logos or toys). To accomplish this, we start by randomly selecting a background image from the MIT-Places dataset [37]. The MIT-Places dataset contains 205 scene categories and a total of 2.5 million images recorded at various locations around the world. We utilize the testing data within this dataset for the process of synthetic images generation. The testing data contains 41,000 images. Figure 7 shows samples from the MIT-Places dataset. We assume that the background image does not contain any object that we are trying to detect. This is reasonable assumption since the MIT-Places dataset is focused on real world or natural scenes.

Then, several object or logo images are randomly selected (1 to 9 images). For each object or logo image, a set of transformations and deformations are used as described below.

Figure 7. *Examples of MIT-Places dataset*

Geometric Transformations

First, the images are randomly rotated with a degree selected uniformly between -40 and 40. Then, a random homographic projection is used matrix 1. Where the parameters h_{11} and h_{12} are randomly selected between -0.001 and 0.001. Next, the image is randomly resized such that the new size is 0.1 to 0.25 times the size of the background image. The parameters are manually selected in order to make the synthesized images look as much real as possible. Therefore, extreme resizes and highly deforming homographic transformations are unwanted.

$$\mathrm{H} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & h_{11} & h_{12} \end{pmatrix} \tag{1}$$

Geometric transformations aim to model different poses of objects and logos. By rotating and resizing the training images, the network is able to be scale and rotation invariant. By using perspective projections combined with capturing multiple poses in the object capture step, the network is able to be pose change invariant.

Color Transformation

After geometric transformations, a small color variation is done to the images. Following the steps in [9, 28] we compute the eigenvalues and eigenvectors of the RGB values of the image. Each of the three eigenvectors is a 3D vector. We then find a randomly chosen weight (uniform distribution between -0.1 and 0.1) for each eigenvector and calculate the weighted sum. The weighted sum is a 3D vector and is added to the RGB vector of each pixel.

Color transformations aims to model small color variations that objects or logos may present in the real world caused by different lighting conditions.

Blurring And Noise Addition

In this step, we use Gaussian blurring with a variance randomly selected between 0.001 and 0.1 and kernel size of 3 x 3. Then, we randomly select a noise model between Gaussian, Salt & Pepper, Poisson and Speckle noise. A small amount of noise is added in the image.

Gaussian noise is commonly generated by capture devices during image acquisition. In order to model Gaussian noise, we add a different random RGB value to each pixel in the image. The random values are extracted from a random variable with normal density function with mean 0 and a variance selected randomly between 1.2 and 2.4. Variance range is selected empirically to introduce a reasonable amount of noise.

Salt & Pepper can originate as analog-to-digital converter errors or transmissions errors. We model Salt & Pepper noise by changing the value of each pixel of the image with probability 0.03. The pixel will be changed either to white, (255, 255, 255) in RGB value, or to black, (0, 0 ,0) in RGB value, both cases with a probability of 0.015.

Poisson noise, or also known as shot noise, can be modeled by a Poisson process. In order to generate Poisson noise, a random variable is created for each pixel. This random variable has a Poisson distribution (Equation 2) with mean λ, where λ is equivalent to the value of the pixel. A random sample is extracted from every random variable. The sample is used to replace each original pixel. After this process, each pixel of the image has been replaced by a random value from a Poisson distribution. This process modifies the image in a non linear way. Because the variance of the Poisson distribution is equal to its mean λ, the darker pixels will not suffer much change while the brighter pixels will have more variation. Finally, we make a weighted average with the original image and the distorted image with weights 0.8 and 0.2 respectively. This average weight aims to avoid images too distorted.

$$P(x) = \frac{e^{-\lambda}\lambda^{x}}{x!} \tag{2}$$

Speckle noise is a granular multiplicative noise. We generate Speckle noise by multiplying each pixel of the image by a random value. The random values are extracted from a random variable with normal density function with mean 1 and a variance of 0.2.

After blurring and noise addition, the noisy images are clipped to range between 0 to 255 before they are added to background images as presented in the next section.

Image Blending

Finally, the object images are blended into the background in a random position ensuring that there is no overlap between various objects. The blending process consist of substituting the pixels of the background image with the pixels of the foreground image. More complex blending techniques, such as Poisson Image Editing [38] , were discarded for simplicity and because they can produce undesired artifacts or distortions to the foreground image. Figure 8 shows examples of generated images. Two synthetic datasets are generated using the process described above. The first dataset contains 16,000 images with logos extracted from FlickrLogos-32 and the second one contains 25,000 images with 15 different toys.

Figure 8. *Examples of generated images using Logos (left) and toys (right).*

Experiments

We describe several experiments here where we train the Faster R-CNN using ZF [18] and VGG16 [19] models, presented in previous sections, with FlickrLogos-32 dataset and synthetic

data. In all the experiments the Mean Average Precision (mAP) is computed using the Pascal VOC 2010 [15] procedure. mAP is defined later. In the Pascal VOC 2010 procedure, for every image each predicted bounding box is compared with all the ground truth bounding boxes of the same class. If the Intersection over Union (IoU) overlap (Equation 3) between the predicted bounding box B_p and some ground truth bounding box B_{gt} is 50% or larger, the prediction is considered as a True Positive (TP), if not, is consider as a False Positive (FP).

$$IoUOverlap = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \tag{3}$$

To compute the mAP, the Precision (Equation 4) and Recall (Equation 5) are required [39, 40]. Precision is the ratio between the True Positives (TP) and the sum of True Positives (TP) and False Positives (FP). Recall is the ratio between True Positives (TP) and the number of ground truth bounding boxes (N_{bbox}).

$$Precision = \frac{TP}{TP+FP} \tag{4}$$

$$Recall = \frac{TP}{N_{bbox}} \tag{5}$$

For each class, a Precision/Recall curve is obtained by varying the threshold parameter from 0 to 1. The Average Precision (AP) is defined as the area under the curve. The Mean Average Precision (mAP) is computed by averaging the AP value for all classes.

The previous process is repeated obtaining the AP for each class. The Mean Average Precision (mAP) is the average of the AP.

In the following experiments, we start the training process using pre-trained models with MS COCO for VGG16 and ImageNet for ZF.

Logo Recognition

In the first experiment, we train a Faster R-CNN with the VGG16 model. The network is trained using various combinations of synthetic images and FlickrLogos-32 images: using only synthetic images, using only FlickrLogos-32 images, and using both synthetic and FlickrLogos-32 images. The Faster R-CNN is trained for 100,000 iterations and we evaluate it every 10,000 iterations using the testing set from FlickrLogos-32. In Table 1 we present the best results for each combination of training data.

Our use of the Faster R-CNN instead of the Fast R-CNN appears to provide a significant improvement. The use of synthetic data together with the original data (FlickrLogos-32) provides an increase of 1.3% respect to only using original data. In Figure 9 we can see some examples of detected logos. The use of synthetic data without any original image has poor performance. We believe this is caused by the loss of information of the background while synthesizing images (i.e. a Corona logo is more likely to be found in a bottle or a Starbucks logo is more likely to be found in a cup of coffee).

Figure 9. *Examples of logos detected in the wild*

Object Recognition

While deploying applications using the HP Sprout the GPU memory can be a limiting factor. In this set of experiments, we introduce the ZF model since it has smaller size than VGG16 and can be used with the HP Sprout GPU. Since we only have synthetic data for toys, a small test dataset was manually labeled using LabelMe [41] for evaluation purposes. The dataset contains 35 labeled images containing up to 15 different toys. The images are captured using HP Sprout camera with good lighting conditions. Figure 10 shows some examples of testing images.

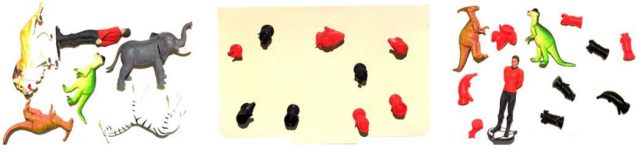

Figure 10. *Examples of toys testing set*

In the first experiment, we train the Faster R-CNN with VGG16 and ZF using the synthetic dataset containing objects (toys). We train several networks using a different number of images from synthetic toys dataset for each one: 25,000 images (100% of synthetic dataset), 12,500 images (50% of synthetic dataset) and 6,250 images (25% of synthetic dataset). Following the previous experiment, the network is trained over 100,000 iterations and it is evaluated every 10,000 iterations. We present the best results using ZF and VGG16.

Note that VGG16 model has a larger mAP than ZF. VGG16 is formed by more layers and it allows the network to learn more complex features. Using only 12,500 images in the training process seems to provide a better performance. The use of a large number of images for training might cause some overfitting and therefore a decrease of performance. In Figure 11 some detection results are presented. Notice that the network is able to differentiate each of the individual red and black toys despite having minor differences between them.

In the last experiment, we analyze the effect of the number of images used in the synthetic generation process. We generate two extra toys datasets with 25,000 images using 10 and 5 clean images per class. We train ZF with the new datasets using different number of synthetic images (100%, 50% and 25% of the

Table 1 Mean Average Precision (mAP) of different methods for logo recognition

Training data	FL32	Synthetic	FL32 + Synthetic	Previous Work [1]
mAP	84.11%	65.55%	85.40%	74.40%

Table 2 Performance (mAP %) using different amount of synthetic data using 20 different poses

Model	6,250 images	12,500 images	25,000 images
VGG16	94.32%	97.42%	96.36%
ZF	92.91%	92.41%	93.41%

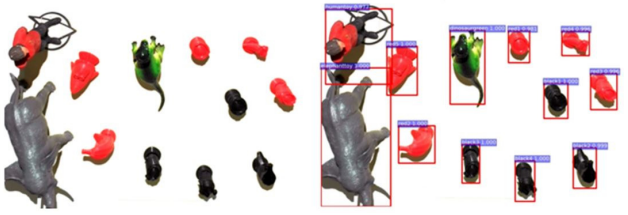

Figure 11. *Original images (left) and objects detected (right) using ZF model*

dataset) and we compare it with the original dataset made using 20 object images per class.

The results presented in Table 3 indicate that the number of original object images used for data synthesis is related with the amount of data required to train the network to achieve a good performance. If a low number of images is used in the synthesis process, the synthesized images will contain less information and less images will be required in the training process. In the example of toys recognition, if more points of view (more images) are used for image synthesis, the synthetic images will contain more information and the average precision will increase. In Figure12 we present the evolution of the mean average precision over the iterations in the training process. We observe that stops increasing between 10,000 and 30,000 iterations and it varies up to 100,000 iterations.

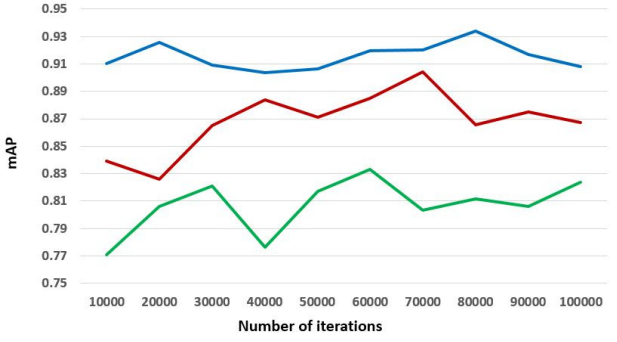

Figure 12. *Evolution of mAP as a function of number of iterations using ZF model trained with 25,000 synthetic images of 20 poses (Blue), 12,500 synthetic images of 10 poses (Red) and 6,500 synthetic images of 5 poses (Green)*

Network Visualization

For a better understanding of how the Faster R-CNN can detects objects in images, we use the visualization toolbox described in [42]. This tool can represent the activations maps generated when an image is processed through the network. We visualize the activation maps of the convolutional layers of the Faster R-CNN using ZF network trained with synthetic toys dataset. In Figure 13 we present different activation maps of the first convolutional layer. The input image (top left) passes through the first convolutional layer and activates different filters. We select some representative filters from the first convolutional layer of the ZF. The white pixels of the activation map represent the regions of the image that activate the filter. We can observe that some filters in the first layer learn features such as green (bottom left) or red colors (bottom right) while other filters detect edges (top right).

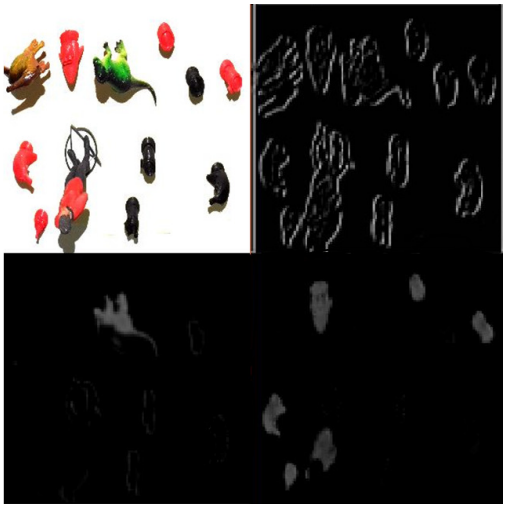

Figure 13. *Activation map of filters from conv1 layer in ZF model*

Deeper layers can learn more complex features. In Figure 14 representative activations in the second, third and fourth convolutional layers are presented. We can observe that black objects are detected (bottom left) in the second convolutional layer. The brown toy is detected in the third convolutional layer (top left) and objects with banana shape are detected in the fourth convolutional layer (bottom right). Each convolutional layer contains some filters that never get activated. This indicates that the network could be reduced in size and obtain the same accuracy for object detection.

Table 3 Performance (mAP %) of ZF model using different amount of images for data synthesis and training

Number of pose images per object	6,250 images	12,500 images	25,000 images
20	92.91%	92.41%	93.41%
10	87.54%	90.43%	88.08%
5	83.32%	80.83%	80.87%

Figure 14. *Activation map of filters from conv2, conv3 and conv4 layers in ZF model*

Conclusions

In this paper, we showed that the Faster R-CNN is able to detect objects with fewer training images. Data augmentation techniques allows us to generate a larger number of images and for satisfactory training. We obtained near real time object recognition using the HP Sprout system with excellent accuracy when test images have clean background and good illumination. While detecting logos, we obtain better results than previous methods. The network is resistant to scale changes, rotations, and small occlusions.

In the future, we want to explore the use of the Single Shot Multibox Detector [25] since it might improve speed and accuracy. We also want to incorporate depth information in the process of image synthesis as it might produce more realistic images [30].

Acknowledgment

This work was supported by HP Labs. Address all correspondence to Edward J. Delp, ace@ecn.purdue.edu

References

[1] F. N. Iandola, A. Shen, P. Gao, and K. Keutzer, "Deeplogo: Hitting logo recognition with the deep neural network hammer," *arXiv:1510.02131*, 2015.

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *Proceedings of the Advances in Neural Information Processing Systems*, pp. 91–99, December 2015, Montreal, Canada.

[3] I. Goodfellow, Y. Bengio, and A. Courville, "Introduction," *Deep Learning*. Cambridge, MA: MIT Press, 2016, vol. 1, p. 20.

[4] A. Rozantsev, V. Lepetit, and P. Fua, "On rendering synthetic images for training an object detector," *Computer Vision and Image Understanding*, vol. 137, pp. 24–37, August 2015.

[5] D. G. Lowe, "Object recognition from local scale-invariant features," *Proceedings of the International Conference on Computer Vision*, pp. 1150–1159, September 1999, Kerkya, Greece.

[6] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-3, no. 6, pp. 610–621, November 1973.

[7] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[8] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Proceedings of the Advances in Neural Information Processing Systems*, pp. 1097–1105, December 2012, Stateline, NV.

[10] H. Kaiming, Z. Xiangyu, R. Shaoqing, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *Proceedings of the European Conference on Computer Vision*, pp. 346–361, September 2014, Zurich, Switzerland.

[11] R. Girshick, "Fast R-CNN," *Proceedings of the International Conference on Computer Vision*, pp. 1440–1448, December 2015, Santiago, Chile.

[12] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, August 2013.

[13] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[14] L. Zhang, L. Zhang, and B. Du, "Deep learning for remote sensing data: A technical tutorial on the state of the art," *IEEE Geoscience and Remote Sensing Magazine*, vol. 4, no. 2, pp. 22–40, June 2016.

[15] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, January 2015.

[16] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," *Proceedings of the European Conference on Computer Vision*, pp. 740–755, September 2014, Zürich, Switzerland.

[17] C.-C. J. Kuo, "Understanding convolutional neural networks with a mathematical model," *Visual Communication and Image Representation*, vol. 41, pp. 406–413, November 2016.

[18] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *Proceedings of the European Conference on Computer Vision*, pp. 818–833, September 2014, Zürich, Switzerland.

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Proceedings of the International*

Conference on Learning Representations (arXiv:1409.1556), May 2015, San Diego, CA.

[20] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587, 2014, Columbus, OH.

[21] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *Visual Communication and Image Representation*, vol. 104, pp. 154–171, September 2013.

[22] P. D. Larry Zitnick, "Edge boxes: Locating object proposals from edges," *Proceedings of the European Conference on Computer Vision*, pp. 391–405, September 2014, Zurich, Switzerland.

[23] "Pinterest," URL: https://www.pinterest.com/.

[24] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proceedings of th IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788, June 2016, Las Vegas, NV.

[25] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," *Proceedings of the European Conference on Computer Vision*, pp. 21–37, October 2016, Amsterdam, Netherlands.

[26] S. Romberg, L. G. Pueyo, R. Lienhart, and R. van Zwol, "Scalable logo recognition in real-world images," *Proceedings of the ACM International Conference on Multimedia Retrieval*, pp. 251–258, April 2011, Trento, Italy.

[27] S. C. Hoi, X. Wu, H. Liu, Y. Wu, H. Wang, H. Xue, and Q. Wu, "Large-scale deep logo detection and brand recognition with deep region-based convolutional networks," *arXiv:1511.02462*, 2015.

[28] M. Paulin, J. Revaud, Z. Harchaoui, F. Perronnin, and C. Schmid, "Transformation pursuit for image classification," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3646–3653, June 2014, Columbus, OH.

[29] J. Xu, D. Vazquez, A. Lopez, J. Marin, and D. Ponsa, "Learning a part-based pedestrian detector in virtual world," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2121–2131, October 2014.

[30] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Reading text in the wild with convolutional neural networks," *International Journal of Computer Vision*, vol. 116, no. 1, pp. 1–20, January 2016.

[31] C. Beattie, J. Z. Leibo, D. Teplyashin, T. Ward, M. Wainwright, H. Kttler, A. Lefrancq, S. Green, V. Valds, A. Sadik, J. Schrittwieser, K. Anderson, S. York, M. Cant, A. Cain, A. Bolton, S. Gaffney, H. King, D. Hassabis, S. Legg, and S. Petersen, "Deepmind lab," *arXiv:1612.03801*, 2015.

[32] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, "Draw: A recurrent neural network for image generation," *Proceedings of the International Conference on Machine Learning*, pp. 1462–1471, July 2015, Lille, France.

[33] Y. Kalantidis, L. Pueyo, M. Trevisiol, R. van Zwol, and Y. Avrithis, "Scalable triangulation-based logo recognition," *Proceedings of the ACM International Conference on Multimedia Retrieval*, pp. 1–7, April 2011, Trento, Italy.

[34] A. Joly and O. Buisson, "Logo retrieval with a contrario visual query expansion," *Proceedings of the ACM International Conference on Multimedia Retrieval*, pp. 581–584, October 2009, Beijing, China.

[35] "Flickr," URL: https://www.flickr.com/.

[36] "HP Sprout," URL: http://www8.hp.com/us/en/sprout/home.html.

[37] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," *Proceedings of the Advances in Neural Information Processing Systems*, pp. 487–495, December 2014, Montreal, Canada.

[38] P. Prez, M. Gangnet, and A. Blake, "Poisson image editing," *Proceedings of the International Conference On Computer Graphics And Interactive Techniques*, pp. 313–318, July 2003, San Diego, CA.

[39] D. M. W. Powers, "Evaluation: From precision, recall and f-factor to roc, informedness, markedness and correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, December 2011.

[40] T. Fawcett, "An introduction to roc analysis," *Pattern Recognition Letters*, vol. 27, no. 8, p. 861874, June 2006.

[41] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "Labelme: A database and web-based tool for image annotation," *International Journal of Computer Vision*, vol. 77, no. 1-3, pp. 157–173, May 2008.

[42] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," *Proceedings of the International Conference on Machine Learning (arXiv:1506.06579)*, July 2015, Lille, France.

Author Biography

Daniel Mas Montserrat graduated from Polytechnic University of Catalonia in 2015. He is currently attending graduate school at Purdue University under the supervision of Professor Edward J. Delp. He is working as a research assistant in a project funded by HP Labs under the supervision of Professor Edward J. Delp, Professor Jan P. Allebach and Qian Lin. His main areas of research are Deep learning and signal and image processing.

Dr. Qian Lin is a distinguished technologist working on computer vision and deep learning research in HP Labs. Dr. Lin joined the Hewlett-Packard Company in 1992. She received her BS from Xi'an Jiaotong University in China, her MSEE from Purdue University, and her Ph.D. in Electrical Engineering from Stanford University. Dr. Lin is inventor/co-inventor for 44 issued patents. She was awarded Fellowship by the Society of Imaging Science and Technology (IS&T) in 2012, and Outstanding Electrical Engineer by the School of Electrical and Computer Engineering of Purdue University in 2013.

Jan P. Allebach is Hewlett-Packard Distinguished Professor of Electrical and Computer Engineering at Purdue University. Allebach is a Fellow of the IEEE, the National Academy of Inventors, the Society for Imaging Science and Technology (IS&T), and SPIE. He was named Electronic Imaging Scientist of the Year by IS&T and SPIE, and was named Honorary Member of IS&T, the highest award that IS&T bestows. He has received the IEEE Daniel E. Noble Award, the IS&T/OSA Edwin Land Medal, and is a member of the National Academy of Engineering. He currently serves as an IEEE Signal Processing Society Distinguished Lecturer (2016-2017).

Edward J. Delp was born in Cincinnati, Ohio. He is currently The Charles William Harrison Distinguished Professor of Electrical and Computer Engineering and Professor of Biomedical Engineering at Purdue University. In 2004 he received the Technical Achievement Award from

the IEEE Signal Processing Society, in 2008 the Society Award from IEEE Signal Processing Society, and in 2017 the SPIE Technology Achievement Award. In 2015 he was named Electronic Imaging Scientist of the Year by IS&T and SPIE. Dr. Delp is a Life Fellow of the IEEE, a Fellow of the SPIE, and a Fellow of IS&T and a Fellow of the American Institute of Medical and Biological Engineering.