# Part-based Object Detection using RealAdaBoost and ANN

*Manuel Montoya Catalá* [1]

[1] Master in Multimedia and Communications

mmontoya@ing.uc3m.es

## Abstract

In this paper, a part-based system for object detection is proposed. The system uses randomly sampled patches from the groundtruth of sample images as parts of the object-model and arranges them in a star shape. Each feature is obtained from a triplet composed by a filter, a patch image and the relative location of the patch to the center of the model. The cross-correlation similarity measure between the patch and the input images is used to obtain the features. The classifier for the object detection is trained using the RealAdaBoost ensemble method along with an Artificial Neural Network trained with a variation of the Extreme Learning Machine algorithm as a weak learner. The properties and results of this system are compared with that one using GentleBoost and Decision-Stumps as classifier.

**Index Terms**: Object Detection, Part-based models, Artificial Neural Networks, Extreme Learning Machine, Boosting, car,

## 1. Introduction

Object recognition is one of the fundamental challenges in computer vision. In this paper we consider the problem of detecting and localizing generic objects belonging to the categories of cars in static images. This is a difficult problem because objects in such categories can vary greatly in appearance not only from changes in illumination and viewpoint, but also due to intra-class variability in shape and other visual properties. We describe an object detection system that represents variable objects using a part-based model.

This project is based on the Lab Session 8 – Object Detection of the subject Computer Vision [1]. The classifier used by this code is the GentleBoost ensemble method using a decision Stumps as weak learners. This paper proposes a more flexible a powerful classifier, a RealAdaBoost ensemble using an Artificial Neural Network trained with a variation of the Extreme Learning Machine algorithm for boosting.

This paper will start describing what a part-based object detection model is and how it is implemented in the proposed system. Since this paper is focused on the classifier, the boosting ensemble will be revisited next, followed by basic introduction to the ANNs and the ELM algorithm. The variation of the ELM algorithm for boosting has been personally derived since it couldn't be found in the literature.

Once all the theoretical background is explained, the experimental setup is described. The dataset will be seen in more detail, viewing how the parts of the model are obtained and how the training and testing images are treated to obtain their features. The parameters of the classifier will be discussed along with some properties of the dataset that are valuable for the tuning and validation of these parameters. Finally, the evaluation of the results, conclusions and future work will be exposed.

## 2. Part based object detection

A part-based model is a model, typically of an object category, that represents the appearance of "parts" and how they relate to each other. In order to implement this model, the first task is to select those object parts that will be used to represent the model. A part is any element of an object (or scene, …) that can be reliably detected using only local image evidence.

Any process can be used to select this parts, i.e. by hand, random samples from the groundtruth or discriminative clustering algorithms. Since the objective of this paper is to improve the results of the detection algorithm, the rest of the components should remain constant; no changes have been made over this part. The parts are selected from the edges of the groundtruth of several images.

Once the different parts of the model have been selected, we have to define their spatial relationship. These relations characterize the mutual independence assumptions we want to make about relative part locations, and it directly affects the number of parameters needed to fully specify the resulting model, as well as the complexity of performing inference using this model. The following figure shows some of these relational models.

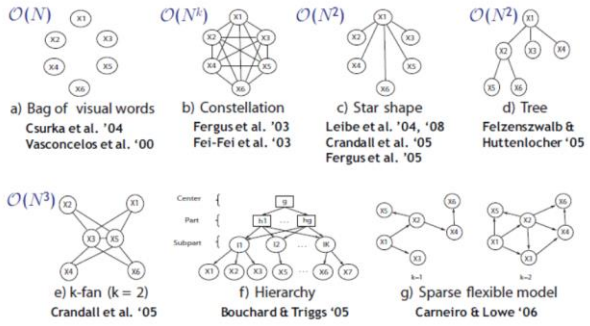


Figure 1 Different Relational Models.

A compromise is to combine the parts in a Star Model (Fig. 7.2(c)), where each part is only connected to a central reference part and is independent of all other part locations given an estimate for this reference part.

### 2.1. Part based model used

The part based model used in this paper uses the Star Model; it contains 640 model parts obtained from the edges of the groundtruth of a set of 8 images. These images will not be used in for training or testing. Each one of these features is composed of 3 elements:

- Filter: Texture filter used over the image. In this paper the filters are selected at random along with the parts. The filters used are the "original image", "x derivate", "y derivate" and "laplacian".
- Part patch: Pre-filtered image part of the model. They are obtained from a subset of the training images groundtruth. 640 parts are obtained from 8 images. Sizes vary over 9-25 pixels square images.
- Location: Relative location of the part with respect to the center of the object.

The following image shows an example of these triplets that shape the features of the system:

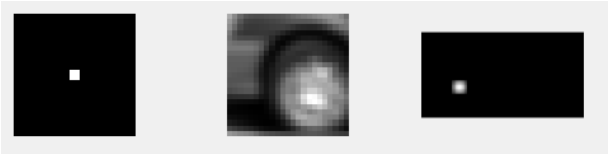

Figure 2 Feature triplet composed by filter, image and location.

The process to obtain the feature corresponding to a given part for an input image is performed by the function *convCrossConv.m* and is as follows: First, the image is filtered with the texture filter of the triplet, then the cross-correlation with the part image is computes for every possible point, finally, the output image is convoluted with the location in order to assign the scores to the center of the object:

$$h_i(I, x, y) = [|I * f_i| \otimes P_i] * g_i$$

Notice that due to the relative position and the size of the patch, the feature cannot be obtained from pixels in the boundaries of the image. The following figure shows this process in a graphical manner.

Figure 3 Illustrated example on how to obtain the feature of all pixels of an image.

Each pixel value of the resulting image is the value of the feature associated to that triplet in that pixel position. The feature space is composed by 640 of these values.

## 3. Boosting

Boosting is an ensemble technique used for creating a strong classifier as a linear combination of weak classifiers. Given a set of T weak classifiers $h_t(\cdot)$, $t = 1,2 \dots T$, the output of the whole system is obtained as:

$$H(\bar{x}_i) = sign\big(f(\bar{x}_i)\big) = sign\left(\sum_{t=1}^{T} \alpha_t \, h_t(\bar{x}_i)\right)$$

These weak classifiers $h_t(\cdot)$ are trained in a sequential manner, each weak learner focuses on a different region of the input space, this is achieved using a weight vector $\bar{D}$ during the training phase. This $\bar{D}$ is a discrete distribution vector over the training samples space that tells the weak learner how much importance it has to give to every training sample. Samples that are poorly classified will have big weight values so the weak learner will focus on them.

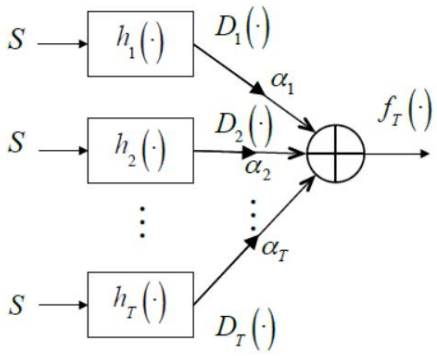


Figure 4 Structure of a Boosting Ensemble.

The emphasis $\bar{D}$ is updated at every weak learner $h_t(\cdot)$ so that the next weak learner $h_{t+1}(\cdot)$ focuses in the samples that have been poorly classified previously. $\bar{D}$ is only used for training, affecting to the cost function of the weak learner, not its actual output.

The general implementation of boosting follows the next definition. Given a training set $\{\bar{x}_i, y_i\}^N$ where $\bar{x}_i \in \bar{X}$ and $y_i \in \{-1,1\}$. Initialize the weight distribution uniformly $D_0(i) = 1/M$. For every weak learner $t = 1, \dots, T$:

1) Train Weak learner using distribution $\bar{D}_t$
2) Get output of weak learner for every sample in the training set $h_t: \bar{X} \to R$.
3) Choose the linear combination constant $\alpha_t$ for this learner.
4) Update the Samples Distribution using the following equation:

$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t \, h_t(\bar{x}_i) y_i}}{Z_t}$$

Where $Z_t$ is a normalization factor.

Boosting's goal is to minimize the exponential loss function, which is minimized when the margin is also minimized

$$E = \sum_{i=1}^{M} e^{-y_i f(\bar{x}_i)} = \sum_{i=1}^{M} e^{-y_i \sum_{t=1}^{T} \alpha_t \, h_t(\bar{x}_i)}$$

Due to the Update rule of the weight vector $\bar{D}$, the training error probability is upper bounded by:

$$P_e \leq \prod_{t=1}^{T} Z_t$$

So the main objective of every weak learner should be minimizing its own particular and independent $Z_t$. Note that $Z_t$ depends on the weight vector $\bar{D}_t$ and on the output of the current weak learner $h_t(\bar{x}_i)$

$$Z_t = \sum_{i=1}^{M} D_t(i) e^{-\alpha_t h_t(\bar{x}_i) y_i}$$

There are many implementations of boosting, this paper uses 2 approaches: RealAdaBoost and GentleBoost.

### 3.1. RealAdaBoost

This implementation chooses the de-emphasis $\alpha_t$ that minimizes an approximation of the normalization constant $Z_t$ at every iteration. No matter what the output of the weak learner is $h_t(\bar{x}_i)$.

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1+r}{1-r}\right)$$

Being $r$ the expected margin over the distribution $\bar{D}$.

$$r = E\{\text{margin}\}_{\bar{D}_t} = E\{ h_t(\bar{x}_i) y_i \}_{\bar{D}_t} = \sum_{i=1}^{M} D_t(i)\, h_t(\bar{x}_i) y_i$$

The upper bound approximation of $Z_t$ used to derive this value of $\alpha_t$ imposes that $h_t(\bar{x}_i) \in \{-1,1\}$ but it finds a good $\alpha_t$ no matter the weak learner $h_t(\bar{x}_i)$.

### 3.2. GentleBoost

Instead of trying to minimize $Z_t$ for a given $h_t(\bar{x}_i)$ using $\alpha_t$, its weak learners directly try to minimize the Taylor approximation of $Z_t$:

$$Z_t = \sum_{i=1}^{M} D_t(i) e^{-\alpha_t h_t(\bar{x}_i) y_i} \propto \sum_{i=1}^{M} D_t(i)\, (h_t(\bar{x}_i) - y_i)^2$$

The cost function of the weak learner must be:

$$C = \sum_{i=1}^{M} D_t(i)\, (h_t(\bar{x}_i) - y_i)^2$$

The advantages of this method is that we don't have to calculate $\alpha_t$, it is implicit in the weak learner $h_t(\bar{x}_i)$, also, $h_t(\bar{x}_i)$ can have any value, it is not bounded by $h_t(\bar{x}_i) \in \{-1,1\}$. On the other hand, this technique does not ensure a good linear combination of the weak learners; a bad weak learner will hurt the overall combination. Weak learners are forced to use the MSE cost function.

## 4. Artificial Neural Networks

An Artificial Neural Network is a machine learning system inspired by the architecture of the mammals' brain. They are composed by a mesh of interconnected artificial neurons (perceptron), simple systems that map an input vector $\bar{x}$ to an output $o$. An ANN is characterizes over 3 main properties: the Neuron Model, the Architecture, and the Learning Algorithm.

The Neuron Model is the model that defines each of the neurons of the ANN. The most used model is a system that outputs a transformed linear combination of its input:

$$o = f_A(Z) = f_A(\bar{X} \cdot \bar{W} + b) = f_A\left(\sum_{i=1}^{N} (w_i \cdot x_i) + b\right)$$

The linear combination of the input Z is called the activation value of the neuron; each neuron also has a input bias that does not depend on any other input. The transformation function $f_A$ is called the activation function; it is usually a sigmoid function. The image below show a graphical representation of the system:

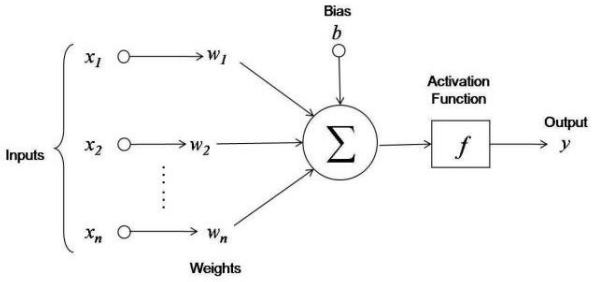


Figure 5 Neuron Model

The Architecture of the ANN defines the number of neurons it has and the interconnections among them. Usually, neurons are clustered into groups called layers; all the neurons in a layer have the same neuron model and interconnection geometry. We can differentiate 3 main kinds of layers; the input layer, in which the input of its neurons is the input of the system, the output layer which neurons have the output of the system as output and the hidden layer, which inputs and output are only connected to other neurons.

In this paper a Single Layer FeedForward Network is used, these layers only have one hidden layer and there are no loops in the neurons interconnections. The image below shows an example of this structure

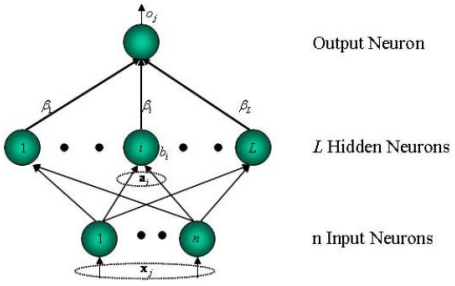


Figure 6 Single Layer FeedForward Network.

The SLFN architecture has the parameters $\{\bar{\bar{W}}, \bar{b}, \bar{\beta}\}$. Given a set of input samples $\bar{\bar{X}}$, the output $o_{ij}$ of every hidden neuron to every input sample can be expressed into a matrix H, called the hidden output matrix:

$$H(\bar{\bar{X}}, \bar{\bar{W}}, \bar{b}) = \begin{bmatrix} o_{1,1} & \cdots & o_{1,\mathrm{L}} \\ \cdots & \cdots & \cdots \\ o_{N,1} & \cdots & o_{N,\mathrm{L}} \end{bmatrix}_{NxL}$$

Every row represents the output of the hidden neurons for a given input sample, $o_{ij}$ is the output of the $j-th$ hidden neuron for the $i-th$ input sample. The total output of the system $\bar{O}$ can be obtained as:

$$\bar{\bar{H}}_{NxL}\, \bar{\beta}_{Lx1} = \bar{O}_{Nx1}$$

The Learning Algorithm is the algorithm that tunes the parameters of the model $\{\bar{\bar{\mathrm{W}}}, \bar{\mathrm{b}}, \bar{\beta}\}$ so that the output of the system $\bar{\mathrm{O}}_{\mathrm{Nx1}}$ is the best possible estimation of the target output $\bar{\mathrm{T}}_{\mathrm{Nx1}}$. The learning algorithm is defined by a cost function that usually depends on the difference between $\bar{\mathrm{O}}_{\mathrm{Nx1}}$ and $\bar{\mathrm{T}}_{\mathrm{Nx1}}$ and an algorithm that tunes the ANN parameters in order to decrease the cost function. The error metric used for most algorithms is the Mean Square Error whose cost function is:

$$E_{MSE} = \sum_{i=1}^{M} \varepsilon_i^2 = \sum_{i=1}^{M} (o_i - t_i)^2$$

We can write this error in a vector form as:

$$E_{MSE} = \bar{\varepsilon}^t \cdot \bar{\varepsilon} = (\bar{O} - \bar{T})^t (\bar{O} - \bar{T})$$

### 4.1. Extreme Learning Machine Algorithm

The ELM algorithm is a fast learning algorithm for SLFNs, basically it chooses the input weight parameters $\{\bar{\bar{\mathrm{W}}}, \bar{\mathrm{b}}\}$ at random and then computes the output weights vector $\{\bar{\beta}\}$ analytically as the Least Square Solution of the system:

$$\bar{\bar{\mathrm{H}}}\bar{\beta} = \bar{\mathrm{T}} \quad \text{with} \quad \bar{\bar{\mathrm{H}}} = \mathrm{f_A}(\bar{\bar{\mathrm{X}}} \cdot \bar{\bar{\mathrm{W}}})$$

We have to find a $\hat{\beta}$ so that:

$$\left\|\mathrm{H}\hat{\beta} - \mathrm{T}\right\| = \min_{\beta} \|\mathrm{H}\beta - \mathrm{T}\|$$

Every neuron can be seen as a hyperplane and its output, the projection of the samples over it. So this algorithm can be interpreted as a generation of random orthogonal hyperplanes, transformed by a sigmoid function and linearly combined by $\bar{\beta}$ as the least square solution of the system with respect to the desired output $\bar{\mathrm{T}}$. The analytical equation for $\bar{\beta}$ is:

$$\hat{\beta} = \mathrm{H}^{\dagger}\mathrm{T} \quad \text{where} \quad H^{\dagger} = \left(\bar{\bar{H}}^t \bar{\bar{H}}\right)^{-1} \bar{\bar{H}}$$

The derivation of the analytical equation of $\bar{\beta}$ is as follows: Given a training set $\{\bar{\mathrm{x}}_\mathrm{i}, \mathrm{t}_\mathrm{i}\}^{\mathrm{N}}$ where $\bar{\mathrm{x}}_\mathrm{i} \in \mathrm{X}^{\mathrm{d}}$ and $\mathrm{t}_\mathrm{i} \in \{-1,1\}$. We have $\bar{\mathrm{T}}$ the vector of desired outputs of the training set, $\bar{\mathrm{O}}$ the vector of outputs of the system and $\bar{\varepsilon}$ , the MSE error between the previous two.

$$\bar{\varepsilon} = \begin{bmatrix} \varepsilon_0 \\ \vdots \\ \varepsilon_{\mathrm{M}} \end{bmatrix} \quad \bar{\mathrm{O}} = \begin{bmatrix} \mathrm{o}_0 \\ \vdots \\ \mathrm{o}_{\mathrm{M}} \end{bmatrix} \quad \bar{\mathrm{T}} = \begin{bmatrix} \mathrm{t}_0 \\ \vdots \\ \mathrm{t}_{\mathrm{M}} \end{bmatrix}$$

Since $\bar{\mathrm{O}} = \bar{\bar{\mathrm{H}}}\bar{\mathrm{W}}_{\mathrm{o}}$ we have the square error to be:

$$\mathrm{E_{MSE}} = \left(\bar{\bar{\mathrm{H}}}\bar{\mathrm{W}}_{\mathrm{o}} - \bar{\mathrm{T}}\right)^{\mathrm{t}} \left(\bar{\bar{\mathrm{H}}}\bar{\mathrm{W}}_{\mathrm{o}} - \bar{\mathrm{T}}\right)$$

$$\mathrm{E_{MSE}} = \left(\bar{\bar{\mathrm{H}}}\bar{\mathrm{W}}_{\mathrm{o}}\right)^{\mathrm{t}} \left(\bar{\bar{\mathrm{H}}}\bar{\mathrm{W}}_{\mathrm{o}}\right) - 2 \cdot \left(\bar{\bar{\mathrm{H}}}\bar{\mathrm{W}}_{\mathrm{o}}\right)^{\mathrm{t}} \bar{\mathrm{T}} + \bar{\mathrm{T}}^{\mathrm{t}} \bar{\mathrm{T}}$$

This error is a convex function of $\bar{\mathrm{W}}_{\mathrm{o}}$ so we take derivative with respect to $\bar{\mathrm{W}}_{\mathrm{o}}$ and equal to 0:

$$\frac{\partial \mathrm{E_{MSE}}}{\partial \bar{\mathrm{W}}_{\mathrm{o}}} = \left(\bar{\bar{\mathrm{H}}}\right)^{\mathrm{t}} \left(\bar{\bar{\mathrm{H}}}\bar{\mathrm{W}}_{\mathrm{o}}\right) - 2 \cdot \left(\bar{\bar{\mathrm{H}}}\right)^{\mathrm{t}} \bar{\mathrm{T}} = 0$$

Resolving the equation we obtain:

$$\left(\bar{\bar{\mathrm{H}}}^{\mathrm{t}} \bar{\bar{\mathrm{H}}}\right) \bar{\mathrm{W}}_{\mathrm{o}} = \bar{\bar{\mathrm{H}}}^{\mathrm{t}} \bar{\mathrm{T}}$$

$$\bar{\mathrm{W}}_{\mathrm{o}} = \left(\bar{\bar{\mathrm{H}}}^{\mathrm{t}} \bar{\bar{\mathrm{H}}}\right)^{-1} \bar{\bar{\mathrm{H}}} \cdot \bar{\mathrm{T}} = \bar{\bar{\mathrm{H}}}^{+} \bar{\mathrm{T}}$$

With $\bar{\bar{\mathrm{H}}}^{\dagger}$ the Moore-Penrouse inverse of $\bar{\bar{\mathrm{H}}}$.

### 4.2. ELM for boosting.

Since we are using boosting, we have to modify the ELM algorithm in order to include the Sample Distribution $\mathrm{D_t}$ so that the error cost function is:

$$\mathrm{E_{MSE}} = \sum_{\mathrm{i}=1}^{\mathrm{M}} \mathrm{D_i} \cdot \varepsilon_{\mathrm{i}}^2 = \bar{\mathrm{D}} \circ (\bar{\varepsilon}^{\mathrm{t}} \cdot \bar{\varepsilon})$$

Where the symbol $\circ$ means element-wise multiplication. Since we don't know the matrix properties of this operation, we express $\bar{\mathrm{D}}$ as a diagonal matrix $\Lambda$ that has D as diagonal:

$$\bar{\bar{\Lambda}} = \begin{pmatrix} \mathrm{D}_0 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathrm{D}_{\mathrm{M}} \end{pmatrix} \quad \bar{\mathrm{D}} = \mathrm{diag}\left(\bar{\bar{\Lambda}}\right)$$

Therefore we can express the error as:

$$\mathrm{E_{MSE}} = \bar{\varepsilon}^{\mathrm{t}} \cdot \bar{\bar{\Lambda}} \cdot \bar{\varepsilon}$$

Moreover, the LS solution for $\bar{\mathrm{W}}_{\mathrm{o}}$ is derived as follows:

$$\mathrm{E_{MSE}} = \left(\bar{\bar{\mathrm{H}}}\bar{\mathrm{W}}_{\mathrm{o}} - \bar{\mathrm{T}}\right)^{\mathrm{t}} \bar{\bar{\Lambda}} \left(\bar{\bar{\mathrm{H}}}\bar{\mathrm{W}}_{\mathrm{o}} - \bar{\mathrm{T}}\right)$$

$$\mathrm{E_{MSE}} = \left(\bar{\bar{\mathrm{H}}}\bar{\mathrm{W}}_{\mathrm{o}}\right)^{\mathrm{t}} \bar{\bar{\Lambda}} \left(\bar{\bar{\mathrm{H}}}\bar{\mathrm{W}}_{\mathrm{o}}\right) - 2 \cdot \left(\bar{\bar{\mathrm{H}}}\bar{\mathrm{W}}_{\mathrm{o}}\right)^{\mathrm{t}} \bar{\bar{\Lambda}}\, \bar{\mathrm{T}} + \bar{\mathrm{T}}^{\mathrm{t}} \bar{\bar{\Lambda}}\, \bar{\mathrm{T}}$$

This error is a convex function of $\bar{\mathrm{W}}_{\mathrm{o}}$ so we take derivative with respect to $\bar{\mathrm{W}}_{\mathrm{o}}$ and equal to 0:

$$\frac{\partial \mathrm{E_{MSE}}}{\partial \bar{\mathrm{W}}_{\mathrm{o}}} = \left(\bar{\bar{\mathrm{H}}}\right)^{\mathrm{t}} \bar{\bar{\Lambda}} \left(\bar{\bar{\mathrm{H}}}\bar{\mathrm{W}}_{\mathrm{o}}\right) - 2 \cdot \left(\bar{\bar{\mathrm{H}}}\right)^{\mathrm{t}} \bar{\bar{\Lambda}} \bar{\mathrm{T}} = 0$$

Resolving we obtain:

$$\left(\bar{\bar{\mathrm{H}}}^{\mathrm{t}} \bar{\bar{\Lambda}} \bar{\bar{\mathrm{H}}}\right) \bar{\mathrm{W}}_{\mathrm{o}} = \bar{\bar{\mathrm{H}}}^{\mathrm{t}} \bar{\bar{\Lambda}} \bar{\mathrm{T}}$$

$$\bar{\mathrm{W}}_{\mathrm{o}} = \left(\bar{\bar{\mathrm{H}}}^{\mathrm{t}} \bar{\bar{\Lambda}} \bar{\bar{\mathrm{H}}}\right)^{-1} \bar{\bar{\mathrm{H}}} \bar{\bar{\Lambda}} \cdot \bar{\mathrm{T}} = \bar{\bar{\mathrm{H}}}^{+} \cdot \bar{\mathrm{T}}$$

As we can see, the equation $\bar{\mathrm{W}}_{\mathrm{o}}$ for the Boosting ELM is quite similar to the normal ELM but it needs to be derived properly.

# 5. Experimental Setup

The system proposed performs object recognition over the cars repository of the LabelMe Dataset [X]. The dataset is composed of 776 images that contain cars in different scenarios; these cars appear from different points of view including partially occluded positions. The dataset is divided into 3 components:

- Patch Selections: These are the images from which we obtain the parts of the model. The parts are selected at random from 20 edges of the groundtruth of 8 images applying each of the 4 filters to them. Thus totaling 640 features of the system. This is computed by the function *createDictionary.m*.
- Training set: These are 20 images from which the training patches are obtained. From each of these images we obtain 30 random patches from the background labeled as negative and a total of 231 positive patches from the groundtruth. The file *computeFeatures.m* performs this selection of positive and negative patches. It also pre-computes the features of these 6321 training patches, so the training patches itself are not stored.
- Testing set: Set of 156 images from which we will obtain the needed feature values for every pixel appling the triplets.

## 5.1. Features obtaining

As previously stated, the features are obtained applying the triplets to the images by means of the function *convCrossConv.m*. The features are obtained from whole images or patches and not from just an independent pixel since the features need to be obtained from a patch region making use of the surrounding pixels.

The features of the training set are obtained over specific patches from the groundtruth (labeled as 1) and from the background (labeled as -1). For the testing images, the cross-convolution of the triplet is computed over the whole image. For an usual 256x256 pixel image, this means we have 65536 samples, one per pixel, although some of them like the ones at the border or the displaced ones will not count since there is not enough information (surrounding pixels) to make a reliable measure.

Since there are 640 features, every image contains 41.943.040 feature values, if they are stored as a 32 bit float number, then a single image occupies 167.772.160 bytes. This means 23.5 images occupy 4 GBytes, this makes pre-computation of the test images infeasible. Also, the time required to obtain a feature for a whole image is relatively high, in the order of seconds.

## 5.2. Training

The training features are already computed and ready to be used in the structure data. If a new set of training images were to be used, we need to rerun the file *computeFeatures.m*

We are left with a dataset made of 6231 samples including 6000 negatives samples and 231 positive samples. This is a highly unbalanced training dataset; therefore it needs techniques that are able to focus on the minority class. A Boosting ensemble method is used with a neural network as a weak learner.

The parameters of the learning algorithm are:

- Nh: Number of hidden neurons of the weak learner.
- T: Number of weak learners.

When the classifier is trained, it is deb

Every weak classifier is defined by the model parameters of every weak learner:

- W: Set of input weights and biases of the hidden neurons.
- b: Set of weights of the hidden neurons.
- beta: Input weights of the output neuron.
- alpha: Linear combination constant of the weak learner output.

The training algorithm has been previously checked with the Abalone NIST dataset to ensure no bugs were made. Cross Validation of the model parameters Nh and T was hard to perform since the error measure here used is not that representative.

## 5.3. Validation

Validation of the parameters of the model was quite challenging, there were two parameters to validate, the number of hidden neurons and the number of weak learners. It usually happened that, as the classifier was better and better trained, the car objects were more big and reliable but some small non-car object start appearing, these small selections are treated as cars thus bringing down the score of the classifier, a simple filter could be used to remove these small detections so that they are not identified as cars.

It was also observed that some of the images where wrongly labeled, their groundtruth does not contemplate some real cars in the images, maybe because they were not expected to be detected. Nevertheless this plays against detectors that are really able to detect these cars.

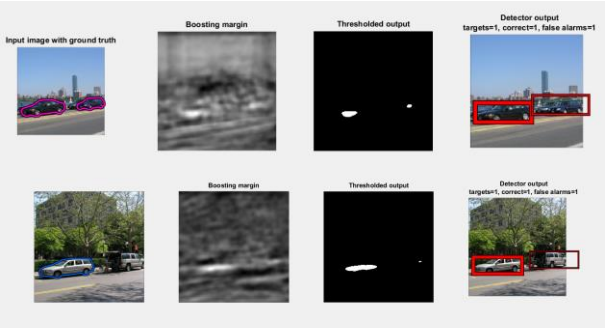


Figure 7 Wrongly labeled detection examples

For a given number of hidden neurons, the more weak learners that were used, the bigger the detections were, but some small points in the background taken as cars started to arise. There was a thin tradeoff between precision and recall that is very affected by these small detections that could be improved if these were eliminated.

For a given number of weak learners, the more hidden neurons they had, the better the weak learner were. One weak learner of around 300 neurons was good enough on its own to build a system with decent performance.

The main temporal bottle-neck of the classifier is the number of learners, not the number of neurons. The compromise solution adopted in this paper is 150 hidden neurons and 6 weak learners since the weak learners are able to perform basic detection and a small combination of them will make detections more robust without increasing the number of false alarms.

## 6. Evaluation

In order to evaluate the performance of the proposed system, we have performed precision-recall graph over 100 testing images using and compare it with the one generated using 150 hidden neurons and 6 learners. The following 2 figures show the precision-recall curves for the ANN weak classifier and for the Decision-Stump classifier.

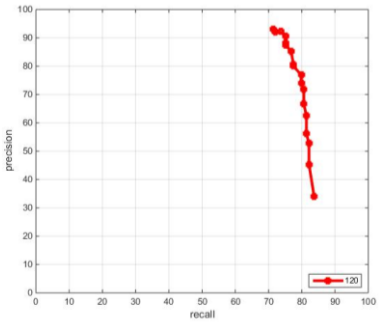


Figure 8 Precision Recall for the Decision Stump

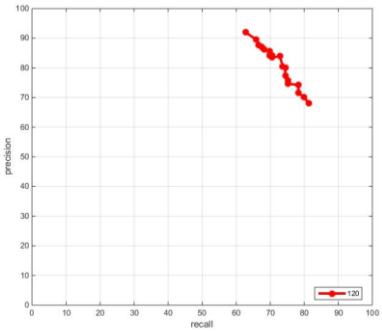


Figure 9 Precision Recall for the

It can be seen that both weak classifiers have a good performance, being the ANN more stable. Other aspects worth mention is that the ELM algorithm takes about 5 seconds to be trained whereas the decision Stump took much longer. On the other hand, the ANN takes more time during testing, one of the causes of this is that the ANN uses all 640 features whereas Decision Stumps only use 120 maximum.

## 7. Conclusions and Future Work

Object recognition using part-based models have been proven to be a reliable and efficient mechanism. The combination of fast weak learners with boosting leads to a fast and adaptable system. Large ANN might be too slow for real-time testing but their parallelizable nature can overcome this problem. Although boosting is needed to avoid over-fitting, there is no need for deep boosting implementations when using an ANN trained with ELM since a single ANN is good enough for performing a basic classification. Having the number of hidden neurons as a parameter increases the adaptability of the classifier, therefore potentially improving the results.

There are many future works to be done with this project concerning all of their stages:

- Each ANN weak learner will only be trained over a random subset of the features instead of being trained among all of them. This will reduce the over fitting and computational cost of the algorithm. It will increase diversity which will benefit boosting. Adding the selection of features as parameter of the weak learner can really boost the classifier.
- Implement the pre-detection filter that removes small false alarms.
- Perform multi-scale detection.

## 8. References


[1] Lab Session 8 – Object Detection from C4.278.12995-1 COMPUTER VISION 14/15-S2.

[2] Antonio Torralba and Bryan Russell , “LabelMe Toolbox: MATLAB Toolbox for the LabelMe Image Database”, 2008 MIT, Computer Science and Artificial Intelligence Laboratory. http://people.csail.mit.edu/torralba/LabelMeToolbox/

[3] Object Detection with Discriminatively Trained Part Based Models. Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester and Deva Ramanan

[4] Chapter 7: Part-Based Category Models from the course Visual Recognition by Kristen Grauman and Bastian Leibe.

[5] ICCV 2013 Tutorial on Part-based Models for Recognition. Sydney, Australia.

[6] Extreme learning machine: Theory and applications. Guang-Bin Huang, Qin-Yu Zhu, Chee-Kheong Siew.

[7] Improved Boosting Algorithms Using Confidence-rated Predictions. Robert E. Schapire Yoran Singer.