Mansoura Journal of Computers and Information Sciences

Representation Learning Framework of Object Recognition via Feature Construction

Muhammad H. Zayyan
Faculty of computers and information systems , C.S dep.
Mansoura University, Egypt,
mhaggag@mans.edu.eg

Samir Elmougy
Faculty of computers and information systems , C.S dep.
Mansoura University, Egypt
mougy@mans.edu.eg

Mohammed F. AlRahmawy
Faculty of computers and information systems , C.S dep.
Mansoura University, Egypt
mrahmawy@mans.edu.eg

ABSTRACT
In this paper, we recognize objects within images by collecting information from a large number of random-size patches of the image. The different backgrounds accompany the foreground object demand to have a learning system to identify each patch as belonging to the object category or to the background category. We strengthen a recent method called Evolution-COnstructed (ECO), which is based on the ensemble learning approach which combines several weak classifier. The improvement is relying on decreasing the overfitting problem. Two different improving ideas are proposed: 1) Pooling operation, which is applied to the weak classifiers data, 2) Random Forest algorithm, which combines the weak classifiers outcomes. Experimental results are reported for classification of 9 categories of Caltech-101 data sets and proved that our modifications boost the performance over the base method and other existing methods.

Keywords
Object recognition, ECO features, Adaboost, Random Forest, Pooling, Genetic Algorithm.

1. INTRODUCTION
The choice of data representation or features has a great impact on the accuracy of machine learning methods. So, many efforts had been used in designing preprocessing pipelines and data transformations that resulted in a representation of the data to support effective machine learning. Feature engineering is the process of using the domain knowledge of the data to create features. This process is important but it is labor intensive and shows a weakness of the current learning algorithms, as they lack the ability to mine the discriminative information from the data.
Building learning algorithms that are less dependent on feature engineering is highly desired to expand the usage of machine learning. Artificial Intelligence (AI) methods are used to learn representation of data, as they can identify and sort out the suppressed explanatory factors unseen in the observed low-level data. Adequate representation of the data makes it easier to extract useful information when building supervised classifiers.
Our proposed recognition approach revisits a method called ECO [1]. It's identifying the remarkable patches of the objects in the images. For each patch, its data representation is controlled by a Genetic Algorithm (GA) [2] which selects the optimal sequence of transformations (filters). Each patch's discriminative ability is assessed by a weak classifier. A number of candidate patches are collected and combined with Adaboost [3] algorithm to build strong classifier.
Overfitting is one of the biggest causes for poor performance of machine learning algorithms. So, this paper handles the issue of overfitting of the learning algorithm through proposing two potential places for enhancements of ECO. First, we enhance the predictive performance of the weak classifier by injecting a pooling operation (which is resizing the original image to half size) after applying each filter in the sequence obtained by GA. Second, we replace the boosting algorithm (Adaboost) in [1] by a more powerful bagging ensemble learning algorithm called Random Forest.

The paper is structured as follows: the next section introduces some related works. Section 3 describes ECO [1, 4]. Section 4 discusses the importance of the enhancements. Section 5 provides the experimental results obtained by applying the proposed framework on the Caltech 101 datasets. Finally, conclusion and future work are provided in Section 6.

2. RELATED WORKS
Most of the current recognition systems are based on specific procedures for extracting the relevant features. As a result; these domain-specific systems are not working well with all real classification tasks. Building a general method for recognition requires learning how to pick up the useful features from the raw images as the training images. Bulitko et al. [5] created automated image interpretation to recognize forest inventorization areas using Markov processes; the same technique is used by Draper et al. [6] to identify common household objects. Lin et al. [7] synthesized effective composite features in recognizing objects in radar modality images using coevolutionary genetic programming. Krawiec and Bhanu [8-13] used an evolutionary algorithm to find series of transformations that are used in composing the features. New features are deduced via utilizing operators such as (add, sub, max) on extracted features such as various moments and other pixel statistics of the image.

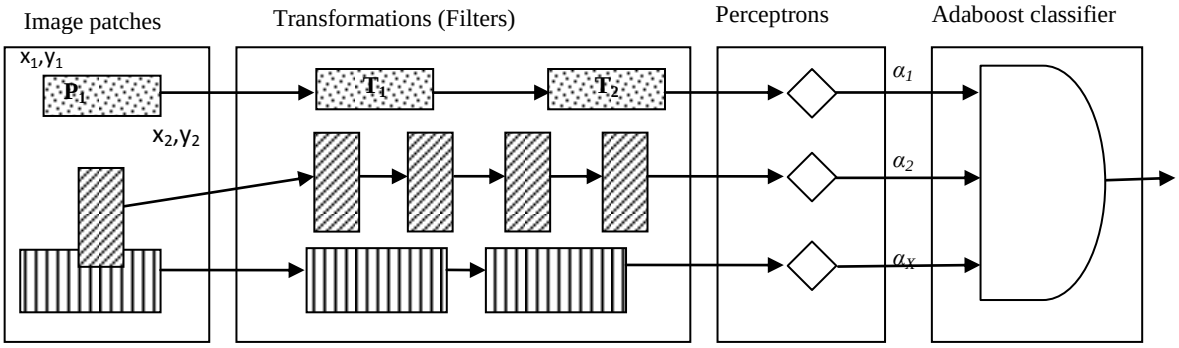


Figure 1: ECO features and their perceptrons are combined by AdaBoost to classify an image [1]

3. RECOGNITION ALGORITHM OVERVIEW

This section gives a summarized overview of the ECO algorithm. ECO method consists of two stages. First stage is used to generate candidate features in which each feature has its own classifier. These classifiers are joined together in the second stage using Adaboost algorithm to construct a strong classifier.

Figure 1 shows an example of three ECO features. Every patch is selected randomly from the image, which is defined by the coordinates x_1, y_1, x_2, y_2. Patches can be overlapped to each other. Selective sequence of transformations is picked by a customizing GA and is applied on the patch. Each feature has its own classifier, which is considered as a weak classifier represented by a single layer neural network called perceptron. The final result of the classification is achieved by selecting the relevant features and combining their perceptron classifiers α using Adaboost model.

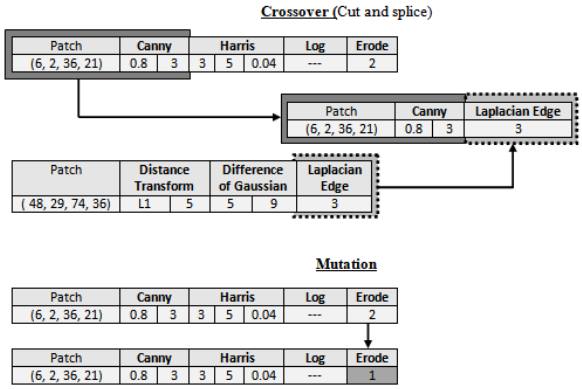


Figure 2: Examples of crossover and mutation of ECO feature

In the customizing GA, a GA chromosome refers to a feature. At initialization, each chromosome has a random patch of the image and can have between 2 and 8 random transformations. The training of the feature consists of applying sequentially its transformations to an image region of interest that is defined by the feature's patch in all training images. The output of last transformation is used in training the feature's perceptron. Once all features have been processed, genetic operator's crossover and mutations are performed to determine the next generation as depicted in Figure 2.Tournament selection is used to determine the features for crossover operator.

Each ECO feature is considered as a weak classifier. The quality of this classifier is computed with the fitness function after training the feature and its perceptron is tested with the testing set of images. The fitness score is in the range of [0,1000] as shown in Equation 1 based on the following terms:

Variables kept in the perceptron class are as follows:

- t_p : number of positive images classified correctly
- f_n : number of positive images classified incorrectly
- t_n : number of negative images classified correctly
- f_p : number of negative images classified incorrectly

$$Fitness\ score = \frac{t_p \times 500}{t_p + f_n} + \frac{t_n \times 500}{t_n + f_p} \quad (1)$$

In Adaboost, the output of the weak classifiers is combined into a weighted sum that represents the final output of the boosted strong classifier.

4. THE PROPOSED MODIFICATIONS of ECO

ECO [1] works accurately on the datasets where the images are cropped around the object of interest. This guarantees, the same patch raw data to be nearly consistent across all images, therefore simplifies the classifier task. On the contrary, object's shape variation in the dataset images makes the same patch inconsistent. As a result, ECO performance degrades when relies on these unremarkable patches as features.

The power of the ensemble strong model depends on the weak classifiers. Therefore, enhancing the quality of the weak classifier increases the final prediction by the model. The proposed additions focus on improving the following classifiers:

4.1 Weak Classifier

ECO [1] employs a weak linear classifier called perceptron which is a single-layer neural network. Perceptron only classifies linearly separable sets of vectors. This means that, if the vectors are not linearly separable, then the classifier fails to discriminate them correctly.

Learning the classifier with an input vector of much data makes it prone to overfitting. To address this issue, a simple approach is used to aggregate statistics of these raw data at various locations. That gains a spatial robustness to object translation. For example, the maximum, the minimum or the average value is computed over a patch of the image. These summary statistics are much lower in dimension (compared to the raw data input vector) and can also improve results (less overfitting). This operation is called "pooling".

The pooling operation is injected after each transformation and hence it downsizes the patch size for the next transformation in the sequence.

4.2 Strong Classifier

ECO [1] utilizes Adaboost ensemble algorithm. AdaBoost works by building a highly accurate strong classifier by combining many relatively weak and inaccurate classifiers. It works iteratively on weak classifiers and gives more weight to

the misclassified classifiers in next iteration. The final classification is done after combining prediction of each classifier. Adaboost works perfectly on weak classifiers that have high bias and low variance. Adaboost can be sensitive to noisy data and outliers which degrade it's predictive performance [14].
We utilize an alternative ensemble learning algorithm called Random Forest [15]. It's based on generating many trees and the final decision based on the majority of votes of each tree. Random Forest works perfectly on overfitting models that have low bias and high variance.

5. THE EXPERIMENTS AND THEIR RESULTS

The proposed framework is implemented within Microsoft visual studio C#. GA and Adaboost are customized coded, while Random Forest, perceptron and images transformations are called from EmguCV v2.4 [16], which is a .NET wrapper of OpenCV. The proposed system runs on an Intel i7 2.20GHz Processor with 8 GB RAM.

5.1 Evaluation Settings

Table 1 lists the parameters of GA used in our recognition framework. The number of GA individuals is set to 100 while the algorithm keeps creating candidate individuals (features) along 10 generations. The length of each individual is in the range of [2 to 8]. Crossover operator type is '*cut and splice*' which results in different lengths of the offspring individuals. To overcome a premature convergence of GA, the lower fitness-value individuals are given a chance to contribute as parents to generate the new offsprings. Individual selection is fulfilled using tournament selection by taking the individual with higher fitness-value of randomly two individuals (size of tournament pool).

Table 1. Configuration setting of the parameters of the GA

Generations count	10
Population size	100
Chromosome's genes length	8
Selection method	Tournament selection
Size of tournament pool	2
Operator type of crossover	Cut and splice
Crossover ratio	90%
Mutation ratio	0.05%

Table 2. List of image transformations with their number of parameters in column named Θ

Image transformation	Θ	Image transformation	Θ
Adaptive Threshold	3	Harris Corner	3
Canny Edge	3	Histogram Equalization	0
Census Transformation	1	Integral Transformation	0
Contrast Limited Adaptive Histogram	2	Laplacian Edge	1
Distance Transformation	2	Log	0
Dilate	1	Median Blur	1
Difference of Gaussian	2	Rank Transformation	1
Erode	1	Sobel	3
Gaussian Blur	1	Square Root	0
Gradient	1	--	--

In our evaluation, we use 19 image transformations in generating ECO features. For every transformation, the number of parameters is specified as stated in Table 2; for example, "Adaptive Threshold" transformation has three parameters: 1) Adaptive type: GAUSSIAN or MEAN. 2) Threshold type: BINARY or BINARY_INV. 3) Block size.

5.2 Evaluation Metrics

Several measures are used in evaluating the efficiency of the recognition system. The following metrics are used:

$$\text{Recall} = \frac{TP}{TP+FN} \quad (2)$$

$$\text{Specificity} = \frac{TN}{TN+FP} \quad (3)$$

$$\text{Precision} = \frac{TP}{TP+FP} \quad (4)$$

$$\text{NegativePredictiveValue} = \frac{TN}{TN+FN} \quad (5)$$

$$\text{FalsePositiveRate} = \frac{FP}{FP+TN} \quad (6)$$

$$\text{FalseNegativeRate} = \frac{FN}{TP+FN} \quad (7)$$

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN} \quad (8)$$

The results of the above measurements are expressed as percentages. These metrics calculations depend on the basic terms: TP, TN, FP, and FN, which are true positive, true negative, false positive and false negative respectively.

5.3 The Experiments Composition

Our framework has two parts: composing features classifiers and combining features classifiers. For each part, there are alternative components. To conduct an experiment, we select a component from each stage. All combinations of different component from each stage are assessed.
Figure 3 illustrates the various configurations of all the experiments where the path with dotted edges represents the only experiment that are applied by ECO [1], while other paths are our proposed solutions.

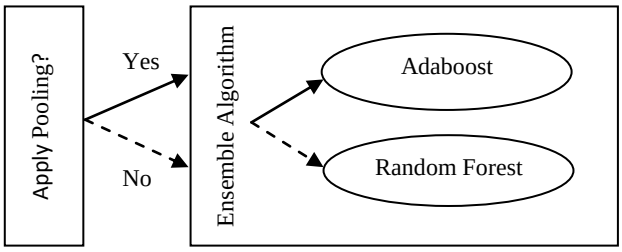


Figure 3: Different configurations of the experiments

With each of the four different configurations, we randomly choose 30 positive and 30 negative images as training data, while the rest of the data is used as testing data. Negative images are chosen from the background category. Each experiment is carried out 5 times with different randomly chosen training and testing images. The results are presented in average accuracies with 95% confidence intervals.
Adaboost and Random Forest are applied to the same features that are generated from GA stage. Moreover, all the experiments are performed using the same seed of randomization.

5.4 Caltech-101 Dataset

We evaluate the performance of our proposed method on 9 different object categories existing in Caltech-101 [17] dataset.

5.5 Evaluating the Framework using Caltech Datasets and Comparing with other Methods

Table 3. Accuracies (%) on binary classification tasks for 9 categories from Caltech-101

Dataset	Other Methods			The Proposed Methods		
	Adaboost (ECO) [1]	SIFT+SVM [18]	Sparse Code [18]	Adaboost	Random Forest	
	without Pooling			Pooling	without Pooling	Pooling
Cellphone	85.5 ±6.9	68.7 ±5.1	**87.9 ±4.2**	93.5 ±4.6	92.8 ±4.6	**95.2 ±4.1**
Dragonfly	82.3 ±5.6	66.0 ±4.0	**87.0 ±4.1**	89.0 ±4.0	87.3 ±4.5	**91.0 ±3.1**
Guitar	76.7 ±8.4	70.0 ±2.4	**80.9 ±5.1**	81.3 ±8.6	83.3 ±8.3	**88.0 ±6.1**
Ibis	76.0 ±4.3	67.8 ±6.0	**83.0 ±1.9**	79.7 ±6.6	87.0 ±3.7	**87.0 ±8.1**
Ketch	**91.7 ±3.9**	82.2 ±0.8	89.2 ±2.4	91.0 ±3.8	95.3 ±2.7	**96.3 ±1.7**
Lamp	78.0 ±6.9	61.5 ±4.5	**81.7 ±3.7**	83.0 ±7.8	82.0 ±9.4	**88.7 ±5.2**
Laptop	81.0 ±7.1	73.5 ±5.3	**87.9 ±2.2**	91.7 ±4.9	84.7 ±8.2	**96.0 ±3.5**
Sunflower	**93.0 ±5.7**	76.0 ±2.5	92.9 ±2.5	98.0 ±0.9	95.3 ±4.0	**99.3 ±1.1**
Watch	80.7 ±8.0	90.1 ±1.0	**91.3 ±2.0**	88.0 ±6.5	87.7 ±4.8	**92.3 ±4.3**

Figure 4 shows sample of the positive images that contain the object for each category. The negative images that are used against the positive images are picked from the *Background* category. All training and testing images are converted to gray color and resized to a resolution of 128 x 80 pixels.

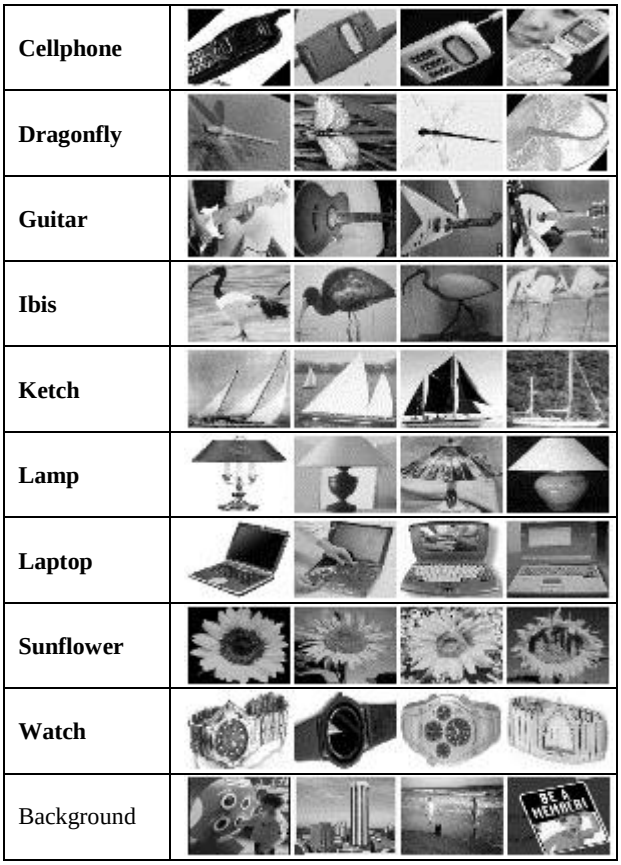


Figure 4: Grayscale images of Caltech datasets

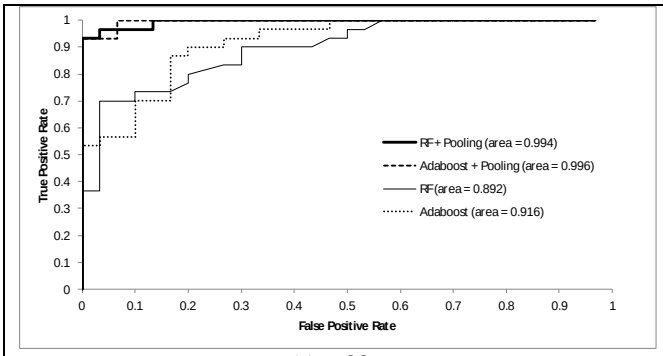
We implemented Adaboost algorithm as used in [1] which creating a weighted sum strong model using several weak classifiers. Random Forest is implemented using a class "*RTrees*" in EmguCV wrapper [16]. There are two parameters that affect Random Forest learning: *maxDepth* that specifies the maximum possible depth of the tree, and *maxIteration* that specifies the maximum number of generated trees. In pooling we used the average operation.

Preparing input vectors for Random Forest learning

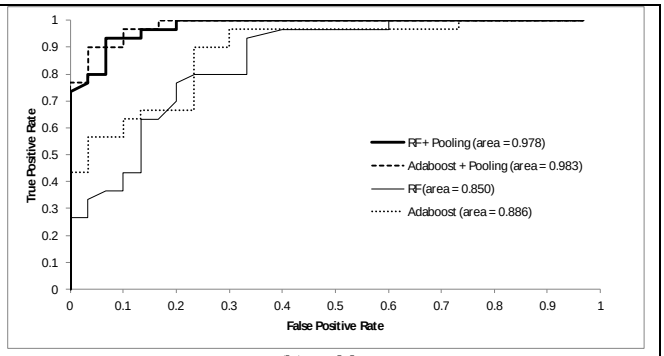
Each training/testing image is converted into input vector that contains 0's and 1's. All the candidate features weak classifiers that are found in GA stage classify each image and the responses of the weak classifiers are concatenated to form an input vector for each image. For positive image, when the weak classifier classifies it, then the response is 1, otherwise is 0. The opposite for negative image where the response is 0 if it's being classified, otherwise is 1.

In Table 3, the left side reports the results that are obtained by contemporary methods. Original ECO [1] and two published methods results in [18] are shown. The right side shows the proposed three different methods configurations as illustrated in Figure 3. Integrating the pooling operation with Adaboost leads to heighten all the Adaboost results for all the datasets except for *Ketch* dataset.

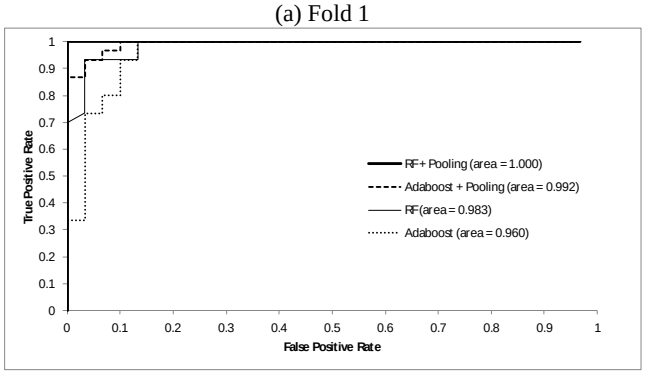
On the other hand, using Random Forest instead of Adaboost returns high results in comparing to results of ECO [1]. The maximum results are obtained when using pooling along with the Random Forest which outperforms all the other methods.



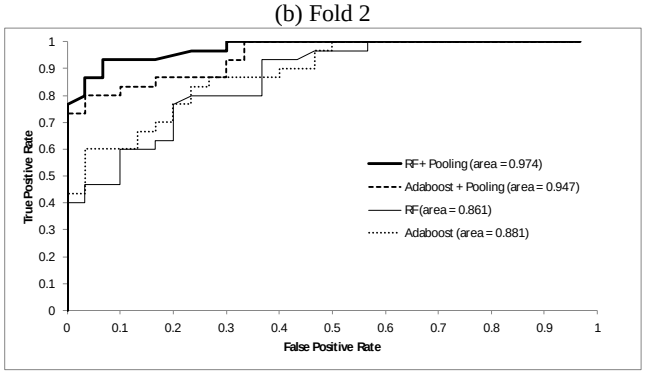
(a) Fold 1



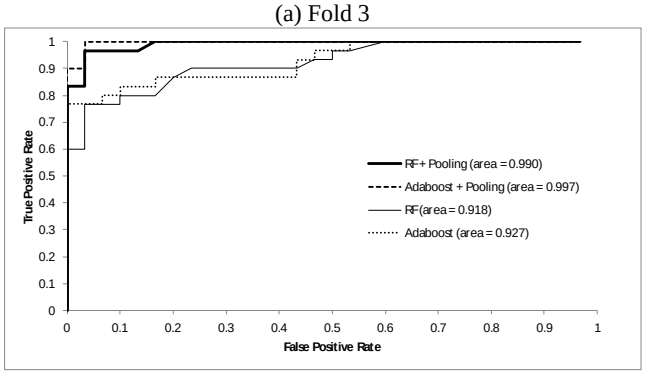
(b) Fold 2



(a) Fold 3



(d) Fold 4



(e) Fold 5

	RF+Pooling	Ada+Pooling	RF	Ada
Fold 1	96.7	95.0	83.3	83.3
Fold 2	93.3	91.7	80.0	73.3
Fold 3	100.0	93.3	95.0	86.7
Fold 4	93.3	85.0	78.3	76.7
Fold 5	96.7	93.3	86.7	85.0
Average	96.0 ±3.5	91.7 ±4.9	84.7 ±8.2	81.0 ±7.1

(f) Accuracies for all folds/classifiers

Figure 5: Effect of using pooling with Adaboost and Random Forest on classification performance using ROC curves - tested on "Laptop" dataset

Figure 5, depicts the impact of our proposed modifications, pooling and Random Forest on the classifier predictive performance using Receiver Operating Characteristic (ROC) curve. We applied four experiments with different configurations as illustrated in Figure 3 on "Laptop" dataset. Each experiment is carried out 5 times (folds). In each fold, true positive rates (TPR) are plotted against false positive rates (FPR) for all the four experiments (classifiers). The advantage of ROC curve is determining the optimal cut off (threshold) value via analysis of the ROC curve itself.
The improvement in performance illustrates by raising the classifier curve, which results in increasing the area under the curve (AUC). AUC is computed for each classifier's curve. The higher value of AUC, the more flexibility of the classifier to tradeoff between TPR and FPR at different thresholds.

In Figure 5f, it shows the accuracy in each fold for all the classifiers. Based on these results, Random Forest with pooling outperforms Adaboost with pooling. On the other hand, the later AUC curve is slightly greater than the former AUC curve in Figure 5a, 5b ,5e. This implies that Adaboost can reach Random Forest highest results but at different 'Threshold' level which differs from a testing set to another. In our evaluation, we use the default threshold level "0" for Adaboost and "0.5" for Random Forest for all experiments.

The accuracy of Random Forest when uses the default cut-off is robust to different testing sets rather than Adaboost.
Each reported accuracy value in Table 3 is computed by running the experiment five times then computing their

average. For example, evaluating the configuration of pooling and Random Forest against "Laptop" dataset returns **96.0±3.5**, where the accuracies percentages of the five repeated experiments are **96.7**, **93.3**, **100**, **93.4** and **96.7** respectively.

Table 4 results are computed based on the accuracy performance metric. Accuracy metric assigns an equal cost to

Table 4: Performance measurements at various thresholds of pooling and Random Forest on "Laptop" dataset

Threshold	True Positive	False Negative	True Negative	False Positive	Recall	Specificity	Precision	Negative Predictive Value	False Positive Rate	False Negative Rate	Accuracy
0.61	23	7	30	0	0.77	1.00	1.00	0.81	0.00	0.23	0.883
0.57	25	5	30	0	0.83	1.00	1.00	0.86	0.00	0.17	0.917
0.54	26	4	30	0	0.87	1.00	1.00	0.88	0.00	0.13	0.933
0.50	**28**	**2**	**30**	**0**	**0.93**	**1.00**	**1.00**	**0.94**	**0.00**	**0.07**	**0.967**
0.45	28	2	29	1	0.93	0.97	0.97	0.94	0.03	0.07	0.950
0.36	30	0	26	4	1.00	0.87	0.88	1.00	0.13	0.00	0.933

the rates of false positives and false negatives, these rates formulate a trade-off. Based on the problem needs; for example, if the requirement is not missing any true positive cases regardless false positives, then we should use Recall metric. Changing the 'Threshold' of the ensemble model can affect all the metrics, so we can apply a user-defined threshold based on the need.

Table 4 shows the computed performance metrics for one of the repeated five times of an experiment with a configuration of pooling and Random Forest. The experiment is applied on the "Laptop" dataset. Threshold level (the majority of votes) is computed as the ratio of the number of trees that votes for the positive class and the total trees in the Random Forest model. The highest accuracy is 96.7% obtained at threshold level of 0.5 with 93% recall, while if we need to have 100% recall, then we should use a threshold level of 0.36.

6. Conclusion

In this paper, we proposed a framework that provides many alternative designs of the evolutionary constructed features in [1] by presenting two enhancements. First, we applied a pooling operation on the data that is fed to the weak classifiers to improve their predictive performance. Second, we utilized Random Forest ensemble learning, which it is more powerful than Adaboost. The highest accuracy is achieved with the use of pooling with Random Forest. In addition, pooling operation reduces the size of the processing data which dramatically decreases the required time for building and evaluating the final model in comparing to ECO [1]. Although we replaced the genetic algorithm with simulated annealing which provides no improvement, we plan to investigate other optimization algorithms as a future work.

7. REFERENCES

[1] Lillywhite, K., et al., *A feature construction method for general object recognition.* Pattern Recognition, 2013. **46**(12): p. 3300-3314.

[2] Mitchell, M., *An introduction to genetic algorithmsThe MIT Press.* Cambridge, MA, 1996.

[3] Freund, Y. and R.E. Schapire. *Experiments with a new boosting algorithm*. in *ICML*. 1996.

[4] Lillywhite, K., B. Tippetts, and D.-J. Lee, *Self-tuned Evolution-COnstructed features for general object recognition.* Pattern Recognition, 2012. **45**(1): p. 241-251.

[5] Bulitko, V., et al. *Adaptive image interpretation: A spectrum of machine learning problems*. in *Proceedings of the ICML Workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*. 2003. Citeseer.

[6] Draper, B.A., U. Ahlrichs, and D. Paulus. *Adapting object recognition across domains: A demonstration*. in *International Conference on Computer Vision Systems*. 2001. Springer.

[7] Lin, Y. and B. Bhanu, *Evolutionary feature synthesis for object recognition.* IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2005. **35**(2): p. 156-171.

[8] Krawiec, K. and B. Bhanu. *Coevolution and linear genetic programming for visual learning*. in *Genetic and Evolutionary Computation—GECCO 2003*. 2003. Springer.

[9] Krawiec, K. and B. Bhanu. *Coevolutionary computation for synthesis of recognition systems*. in *Computer Vision and Pattern Recognition Workshop, 2003. CVPRW'03. Conference on*. 2003. IEEE.

[10] Krawiec, K. and B. Bhanu, *Visual learning by coevolutionary feature synthesis.* Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 2005. **35**(3): p. 409-425.

[11] Krawiec, K. and B. Bhanu, *Visual learning by evolutionary and coevolutionary feature synthesis.* Evolutionary Computation, IEEE Transactions on, 2007. **11**(5): p. 635-650.

[12] Krawiec, K. and B. Bhanu, *Visual Learning by Evolutionary Feature Synthesis.* IEEE Transactions on Evolutionary Computation, 2007. **11**(5): p. 635 - 650

[13] Lin, Y. and B. Bhanu. *Learning features for object recognition*. in *Genetic and Evolutionary Computation—GECCO 2003*. 2003. Springer.

[14] Schapire, R.E., *Explaining adaboost*, in *Empirical inference*. 2013, Springer. p. 37-52.

[15] Breiman, L., *Random forests.* Machine learning, 2001. **45**(1): p. 5-32.

[16] Emgu. *Emgu CV is a cross platform .Net wrapper to the OpenCV*. April 2016; Available from: http://www.emgu.com/wiki/index.php/Main_Page.

[17] Fei-Fei, L., R. Fergus, and P. Perona, *Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories.* Computer vision and Image understanding, 2007. **106**(1): p. 59-70.

[18] Hong, Y., et al., *Unsupervised learning of compositional sparse code for natural image representation.* Quarterly of Applied Mathematics, 2013. **72**: p. 373-406.