https://doi.org/10.2352/EI.2022.34.10.IPAS-346

Training Decision Trees to Guide Feature Selection for Infrared Image Pre-Screening Algorithms

Dawne M. Deaver, Nader Namazi, Department of Electrical Engineering and Computer Science, The Catholic University of America, Washington, D.C., USA

Abstract

This research explores a fresh approach to the selection and weighting of classical image features for infrared object detection and target-like clutter rejection. Traditional statistical techniques are used to calculate individual features, while modern supervised machine learning techniques are used to rank-order the predictive-value of each feature. This paper describes the use of Decision Trees to determine which features have the highest value in prediction of the correct binary target/non-target class. This work is unique in that it is focused on infrared imagery and exploits interpretable machine learning techniques for the selection of hand-crafted features integrated into a pre-screening algorithm.

Background

The US Army uses infrared (IR) sensors to detect and classify enemy threats and has long sought to field algorithms which are effective at aiding Soldiers in target acquisition. Algorithm-assisted target acquisition tasks are accomplished in multiple stages: (1) a pre-screener to locate regions of interest using the sensor's low resolution, wide Field of View (FOV) and (2) a classifier to further discriminate objects using the sensor's high resolution, narrow FOV. This research focuses on improved methods of choosing image features for a pre-screener that do not require processing with a neural network.

Pre-screening algorithms operating in midwave and longwave infrared (IR) imaging wavebands enable day/night computer-aided target acquisition by rapidly separating wide area scenes into "target" and "non-target" regions. Early detection algorithms, such as the "Scale-Insensitive Detection Algorithm for FLIR Imagery" published by Der, et al. of the Army Research Laboratory (ARL) [1], use a double-window sub-image for calculating traditional image-based features to locate targets in IR imagery. ARL researchers led by Chan later published a multi-stage IR target detector [2], which added an Eigenspace Multi-Layer Perceptron (EIGMLP) to separate clutter from real targets, and finally the "Evidence Integrator" which combined the output of EIGMLP with intermediate feature values from the initial detector. Young, et al., [3] proposed the use of Eigenspace Separation Transforms (EST) and Principal Components Analysis (PCA) applied to gradient vectors to detect differences in the structural information between low-contrast, short-to-medium range targets and non-targets. Mehmood and Nasrabadi [4] proposed the "wavelet-RX" algorithm, which uses two-dimensional wavelet transforms to decompose the image into a number of sub-bands, followed by the multi-variate RX Constant False Alarm Rate (CFAR) algorithm, adopted from hyperspectral image processing.

A parallel research track grew out of studies of human search performance using visual saliency models to locate objects of interest in imagery. The benchmark work by Itti and Koch [5] established an algorithm to model human visual attention, or saliency, given a standard 3-color image. Their model defined multi-scaled features based upon color, intensity, and orientation, and linearly combined these to produce a saliency map. In 2013, Chen, et al. [6] proposed an adaptation of the Itti and Koch saliency model for use with static IR imagery. This technique produces a saliency map as a weighted sum of spatial frequency, orientation, and shape-based features. Borji, et al [7] provide an excellent summary of the past two decades of salient object detection techniques for color imagery.

The focus of this research is to combine previously defined features from anomaly detection, image pre-screeners, and saliency models and to establish a generalized methodology for feature down-selection. This will result in an improved algorithm to pre-screen infrared (IR) imagery for ground-based objects of interest, while maximizing target detections and minimizing the time required to cover the search sector. The hypothesis is that performance of traditional automated pre-screening algorithms can be improved by incorporating visual saliency and a select subset of possible IR image-based features.

Methodology

A database of publicly available infrared imagery with labeled vehicles is used as input to a baseline Stationary Target Indicator (STI) algorithm to produce a set of target and target-like Regions of Interest (ROI). For each ROI, a set of pre-defined image feature values are computed: 20 features from the baseline STI algorithm and 60 features from the saliency (SAL) model of Itti and Koch. Binary classification Decision Trees (DT) are trained and optimized using 10-fold cross validation to fine-tune training parameters for minimum misclassification error rates. The process is repeated with three different subsets of features to determine the predictive value of each feature, and provide a relative rank-ordering of feature importance within the trained DTs. The analysis provides an approach using supervised machine learning to down-select and combine a set of available features to use for improving a traditional pre-screening algorithm.

Feature Database Development

Infrared Imagery

The data for this analysis is a set of infrared imagery collected by the U.S. Army is available for public distribution for use in automated target recognition algorithm development [8]. Commonly referred to as the "DSIAC database", it includes video sequences of ten foreign military and civilian vehicles, as well as ground truth providing frame-by-frame bounding box locations of all vehicles in the scene. The present analysis uses all videos with vehicles being slowly driven in a circle centered at ten discrete ranges from 500 m to 5000 m. In order to reduce highly correlated frame-to-frame target signature information available in the 30-Hz source video, the video sequences are temporally down-sampled to 0.5-Hz prior to further processing. By doing so, each 1800-frame video of a vehicle traversing the circular path is reduced to 30

frames, resulting in a snapshot of each vehicle at roughly 12 degree aspect angle increments. For further description of the infrared sensor and imagery, the reader is referred to the referenced User Guide.[9]

Train/Test Set Partitioning

The vehicle data was split into training and testing sets by assigning eight of the range bins to the training set, and the remaining two to the test set. This partitioning scheme, shown in Table 1, was deliberately chosen instead of a random partition in order to reduce the correlation of the scene background between training and training sets, given that all of the imagery is from a common site.

Table 1 Training/Testing Set Partitioning

	Training Set	Testing Set
Video Sequences	123 videos	36 videos
Image Frames	3690 frames	1080 frames
Range Bins (meters)	500, 1000, 1500, 2500, 3000, 3500, 4500, 5000	2000, 4000

Region of Interest Selection

For this analysis, a set of 80 engineered features (described in the next section) are computed for regions in each image which contain targets or target-like signatures. The ROIs are generated in two ways. First, the baseline ARL pre-screener [1] , hereafter generically denoted STI, is configured to report the top thirty (30) most likely target ROIs per image frame, based on a rank order of the declaration confidence score calculated by the algorithm. Second, the resulting ROIs are compared to ground truth and augmented to include bounding boxes for any true targets missed by the STI. The ROI database is designed to contain all of the actual targets along with a large number of the most challenging target-like regions. A sample infrared image frame in **Figure 1** is overlaid with bounding boxes of the 30 most target-like regions output from the STI algorithm.

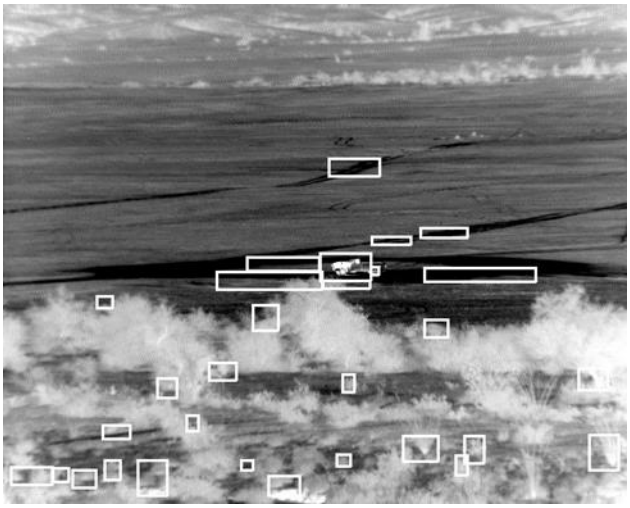

Figure 1 *Example infrared image frame overlaid with 30 ROIs from the baseline STI prescreening algorithm. Pickup truck is near center of scene.*

Table 2 summarizes the ROI count for targets (Class 1) and non-targets (Class 0), along with the *a priori* probability of each class being randomly selected in the training and testing sets. While the vehicle type sample count within Class 1 is fairly uniform (μ_{trn}=414, σ_{trn}=40, μ_{tst}=132, σ_{tst}=11), there is a large class imbalance in Class 0 and Class 1 samples – roughly 23:1 ratio between non-targets and targets.

Table 2 Region of interest number of samples and class priors for training and testing sets

	Training Set ROIs	Testing Set ROIs
Class 0 (Non-Target)	N = 107,551 Pr(C0) = 0.9629	N = 30,588 Pr(C0) = 0.9629
Class 1 (Target)	N = 4,140 Pr(C1) = 0.0371	N = 1,318 Pr(C1) = 0.0371

Feature Descriptions

This analysis uses engineered feature types drawn from two different algorithms, both of which internally produce pixel level Feature Maps. The "STI Features" are derived from a tailored version (code revision ao18) of the ARL STI algorithm. The STI uses a sliding double-window scheme in which the inner window is sized to an expected target at an expected range, and the outer window represents local background pixels. The feature types include Local Extreme, two types of Contrast, Local Variance, and Average Gradient. Each of the five feature types is calculated for inner windows sized for two different target aspects (side-view and front/rear view) at a single range gate, resulting in 10 pixel level STI feature maps. The "SAL Features" are derived from the orientation and intensity feature maps described in the Itti and Koch saliency model. (Color feature maps were omitted due to irrelevance to IR imagery.) When computed at six different spatial scales, the four orientation maps and one intensity map result in 30 pixel level SAL feature maps. To compile the master feature database, the average and maximum pixel values within the area of each ROI are computed from the 40 pixel-level feature maps, resulting in a total of 80 feature values for each candidate ROI.

Binary Decision Tree Classification Model

A classification decision tree (DT) is a supervised machine learning method that splits the training data into subsets using the available features in order to separate Class 1 from Class 0 samples. The data is split as many times as needed (up to N-1 nodes, where N is the number of samples) in order to predict the true class of the input data. For this analysis, the decision tree predictors are real, continuous-valued features, and the objective is to assign a discrete binary class (target or non-target) to each sample, where any civilian or military vehicle in the ROI is considered a target belonging to Class 1, and all other ROIs belong to Class 0. The objective of creating the binary decision trees is to analyze the relative strength of each feature for an improved pre-screening algorithm.

Decision Trees Trained on Feature Subsets – Default Parameters

In the first experiment, three independent DT classifier models were grown to compare the performance of each tree using three feature subsets: STI, SAL, and ALL, where ALL includes both STI and SAL feature sets. Each feature vector was scaled to a

zero-mean, unit variance distribution for all of the samples in the training set.

The MATLAB Statistics and Machine Learning Toolbox [10] function *fitctree* was used to build each DT Classifier. The split criterion used Gini's diversity index, with a maximum number of splits to be one less than the total training samples.

The confusion matrices for each of the three DT classifiers, when used to predict the classes for data in the Training Set and Testing Set are shown in **Figure 2** and **Figure 3,** respectively. The percentages shown along the diagonals are correct predictions, while the percentages shown on the off diagonals are incorrect predictions. The first observation is that each DT classifier performs significantly better on the Training Set than the Testing Set, which is to be expected, since no explicit attempt was made in the initial training process to avoid overtraining or to increase generalization of the DT. A second observation is that the DT with default training parameter settings performs best when using only the STI features, and worst when using only the SAL features. Third, all DT models are better at correctly predicting Class 0, the non-target class then they are at predicting Class 1, the targets.

In addition to comparing confusion matrices, calculation of the misclassification error is helpful in evaluating overall classifier performance.

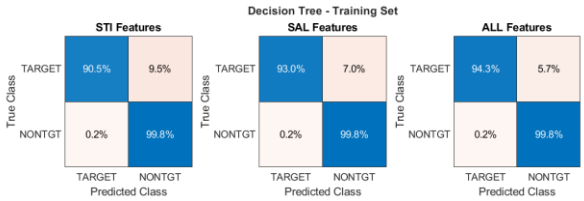

Figure 2 *Training set confusion matrices for decision trees trained using default parameter settings and three different subsets of features.*

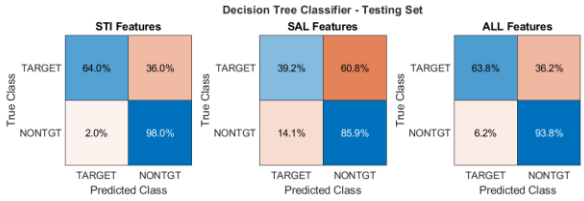

Figure 3 *Testing set confusion matrices for decision trees trained using default parameter settings and three different subsets of features.*

DT Performance Metric: Misclassification Error

The misclassification error, L, given by **Equation (1)**, measures the inaccuracy of the predicted class, $\hat{y}_j$, based on feature data input, compared to the actual class, y_j. Lower values correspond to a better predictive model.

$$L = \sum_{j=1}^{n} w_j I\{\hat{y}_j \neq y_j\}, \quad (1)$$

where, $\hat{y}_j$ is the predicted class of sample j. The sample weights, w_j, are computed to ensure the contributions from each class correspond to the prior class probability, and such that the sum over all sample weights are normalized to 1. The operator I produces a value of 1 for misclassified samples and a value of 0 for correctly classified samples.

Table 3 shows the misclassification error for the three un-optimized DT Classifier Models when computed with the Training Set ("in-sample [IS] error") and Testing Set ("out-of-sample [OOS] error"). A low IS error does not imply good generalization of the decision tree on the unseen test data, and therefore is not the best metric to compare model performance. The values in the table represent the decimal rate at which misclassifications (i.e. errors) occur. For example, for the DT trained with STI features, the IS misclassification rate is 0.6%, while the OOS misclassification rate is 3.2%.

The class priors shown in **Table 2** may be used as a benchmark for the computed loss rates. Consider a classifier that always predicts the majority class (in this case, Class 0) regardless of feature value input. The misclassification rate for such a majority-class predictor is equal to the Class 1 prior, or 0.0371. With respect to the OOS misclassification error, only the STI-feature-trained DT performs better than a majority-class predictor.

Table 3 IS and OOS misclassification error for decision trees trained with 3 feature subsets using default training settings.

	Samples, N	STI	SAL	ALL
Training Set	111,691	L_{IS} = 0.006	L_{IS} = 0.005	L_{IS} = 0.004
Testing Set	31,767	L_{OOS}= 0.032	L_{OOS}= 0.159	L_{OOS}= 0.073

The following section will discuss the application of k-fold cross validation to estimate the generalizability of the DT model against unseen data without explicitly using the Testing Set.

K-Fold Cross-Validation

To perform cross-validation, the training set is randomly partitioned into 10 folds. The MATLAB *crossval* function was used to create 10 new model partitions, with each partition successively trained on data in 9 out of the 10 folds. The data used for training each partition are considered "in-fold" (IF) observations, while the remaining fold is used to validate classification performance with of "out-of-fold" (OOF) observations. The predictive accuracy of the trained partitions was then estimated using the *kfoldloss* function to predict the response and misclassification error on the out-of-fold data.

Table 4 shows the IF and OOF misclassification errors for the un-optimized DT trained on three feature subsets. Unsurprisingly, the OOF misclassification errors resulting from the k-fold cross-validation method are considerably worse than the IF misclassification errors calculated when all of the data was used for training a single DT. The cross-validation method provides a better estimate of the generalizability of the DT model to unobserved data.

Table 4 IF and OOF misclassification error for decision trees trained with 3 feature subsets using default training settings.

	Samples, N	STI	SAL	ALL
Training Set "In-fold"	111,691	L_{IF} = 0.006	L_{IF} = 0.005	L_{IF} = 0.004
Training Set "Out-of-fold"	Each Partition: Trn= 100,522 Tst= 11,169	L_{OOF} = 0.0275	L_{OOF} = 0.0290	L_{OOF} = 0.0213

Again, we can consider a comparison of the OOF misclassification error with that of a majority-class predictor. Unlike the performance against the Test Set (L_{OOS}), the OOF misclassification error (L_{OOF}) for each of the three DT models trained with 10-fold cross validation is slightly better than a majority-class predictor, although still an order of magnitude worse than the L_{IF} metric calculated using the Training Set.

Note that by convention, data in the Testing Set is never used in actual training of any of the Decision Tree models, whereas the results of k-Fold Cross Validation are a surrogate used to estimate the performance trends when the final DT is tested against unseen data (i.e. the Testing Set.) The next step is to vary certain parameters while training the DTs and use K-Fold Cross Validation to evaluate how the parameter settings affect the OOF misclassification rates. By doing so, optimal training parameters can be found and used to train a final DT classifier.

Decision Tree Depth Parameter Optimization

Three training parameters control the depth and complexity of the trained DT. These are *MinLeafSize*, *MaxNumSplits*, and *MinParentSize*. Each of these parameters are individually varied over range of values while performing k-fold cross validation to evaluate the OOF misclassification rate. The parameter settings found to yield the lowest misclassification rates for the three feature sets will be used to retrain each tree for better generalizability with the test set.

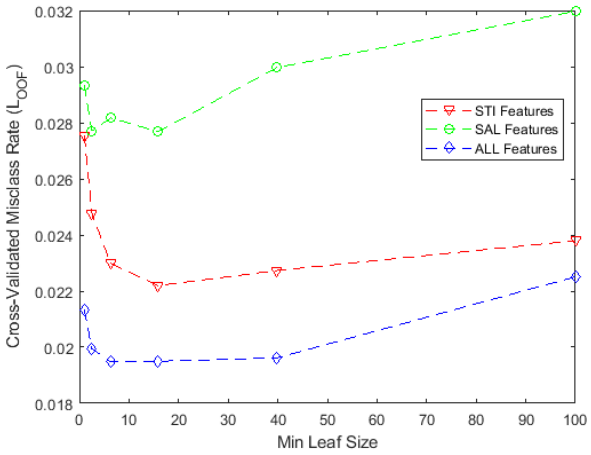

Figure 4 *Out-of-fold misclassification rate for different values of the minimum leaf size tree depth parameter.*

The baseline (unoptimized) DT was trained using a default minimum leaf size of 1, which means that the decision nodes can continue to split until there is only a single sample at the lowest level of the tree (i. e. the leaf.) As shown in **Figure 4**, the minimum leaf size was sampled logarithmically between values of 1 and 100 and yielded the best overall misclassification rates at *MinLeafSize*=16 for each of the three feature sets. This means that with this data there is better predictive power in a decision tree that requires at least 16 training observations at the lowest layer of the tree, and implies that a tree that splits down to a single observation at each leaf becomes overtrained for the test set, and does not generalize as well to unseen data.

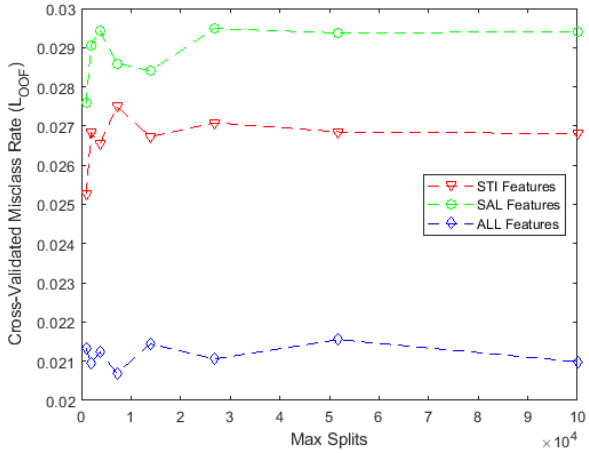

Figure 5 *Out-of-fold misclassification rate for different values of the maximum number of split tree depth parameter.*

The maximum number of splits was sampled logarithmically between 1000 and 100000, with the default being (Nsamples-1) (111,690). As shown in **Figure 5**, a *MaxNumSplit* value of 1000 yields the best OOF misclassification error (L_{OOF}) for two of the three cross-validated models (i.e. STI and SAL). For the third model (ALL), there is very little difference (delta = 0.0006) between the value at 1000 and the minimum sampled. For the optimized tree, the *MaxNumSplit* value will be set to 1000 for all three re-trained DTs.

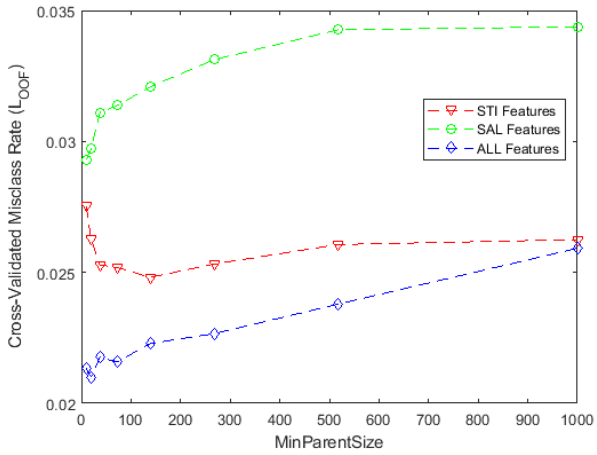

Figure 6 *Out-of-fold misclassification rate for different values of the minimum parent size tree depth parameter.*

The default setting for *MinParentSize* is 10. This means that any decision node must have at least 10 training observations to continue to split into classes. Out-of-fold misclassification rates for *MinParentSize* sampled logarithmically at values between 10 and 1000 are plotted in **Figure 6**. Above a value of 100, all of the OOF misclassification rates increased monotonically; however there is no single optimal setting below 100 that minimizes the L_{OOF} for all three models. Note that when training a DT, specifying both the *MinLeafSize* and *MinParentSize* could result in a condition of over-specifying parameters (because a parent node has two leaf nodes). The lowest value of *MinParentSize* is twice the *MinLeafSize* specified. With a *MinLeafSize* of 16, the *MinParentSize* used in retraining the DTs is 32.

Retrained DT with Optimal Parameter Set

Finally, using the set of optimal parameters for controlling the tree depth, the three DTs are retrained twice more: (1) using all available training data in order to compute the in-sample misclassification rate, and (2) using 10-fold cross validation in order to compute the out-of-fold misclassification rate. Table 5 shows key training parameters before and after the tree depth parameter optimization.

Table 5 Decision Tree training settings before and after optimization.

Training Parameter	Non-Optimized	Optimized
MaxSplits	111690	1000
MinLeafSize	1	16
MinParentSize	10	32

Evaluating the misclassification errors indicates that using the optimal parameter set was somewhat detrimental to the in-sample misclassification rate as listed in **Table 6**, as compared to using the default parameter values (**Table 4**.) The L_{IS} values are shown only for completeness, since it is the L_{OOF} calculated from the cross validated models that provide insight as to the expected performance on unseen data. Using the tailored parameter set very minimally reduced the out-of-fold misclassification rate for all three feature sets, as compared to using the default parameter settings (**Table 4**.) The reductions, ranging from one-tenth to one-half of a percent of misclassifications, are achieved with less complex DT models.

Table 6 IS, OOF, and OOS misclassification error for decision trees trained with 3 feature subsets using optimized training settings.

	Samples, N	STI	SAL	ALL
Training Set	111,691	L_{Is}= 0.005	L_{Is}= 0.019	L_{Is}= 0.013
10-fold Cross Validation	Per Partition: Trn=100,522 Tst= 11,169	L_{OOF}= 0.023	L_{OOF}= 0.028	L_{OOF}= 0.019
Testing Set	31,767	L_{OOS}= 0.023	L_{OOS}= 0.070	L_{OOS}= 0.025

The Testing Set confusion matrices for each of the three DT classifiers retrained with the optimal parameters are shown in **Figure 7**Error! Reference source not found.. Compared to results shown in **Figure 3**, using the optimized parameters to train the DT was detrimental to performance against Class 1 (targets) while beneficial to performance against Class 0 (non-targets) for all three feature subsets. This is not a desirable result for the military use-case, in which it is critical to find a high percentage of the regions containing targets. Although the L_{OOF} and L_{OOS} metrics were improved by using k-fold cross-validation to find optimal tree depth training parameters, the misclassification error alone is insufficient in determining whether the resulting model is better suited to a specific task.

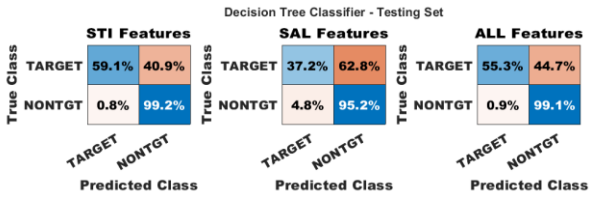

Figure 7 Testing set confusion matrices for decision trees trained using optimized parameter settings and three different subsets of features.

Feature Ranking and Selection

Using each of the trained DTs, the predictor importance values are estimated by summing changes in the risk due to splits on every predictor and dividing the sum by the number of branch nodes. Note that the predictors used in each DT correspond to the STI, SAL, and ALL feature subsets that have been described.

Figure 8 shows the predictor importance values for the DT trained using only the 20 STI features. The feature names are listed down the y-axis, and identify the range gate (always gate 1), the target box aspect (1 or 2), the feature number (1-5), and whether the maximum or average inside the ROI was calculated. The predictor importance values are shown for the DT trained with the default tree depth parameter settings, as well as those that were optimized. The specific features are ranked by predictor importance using the default settings. It is notable that the highest ranking STI feature (F_STI_gate01_aspect02_feat04_ave) is so dominant on a logarithmic scale over all other STI features.

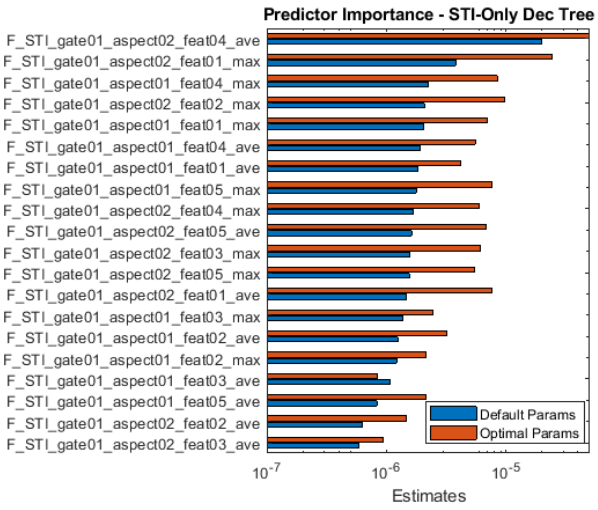

Figure 8 Predictor importance for DT trained using STI features. Features are rank ordered using the results of the default training settings. All twenty of the STI features are shown.

Figure 9 shows the predictor importance values for the DTs trained using only the 60 SAL features. The feature names for saliency indicate whether it is intensity-based (IFM) or orientation-based (OFM). The orientation features are further coded by the angle of orientation (000, 045, 090, and 135), which correspond to vertical, horizontal, and two diagonal angles. Due to space considerations, only the top 20 SAL features are shown in the plot.

Figure 10 shows the predictor importance values for the DTs trained using all 80 available features, with the top 20 ranked features shown on the plot. It is interesting to note that the ranking of the combined feature set does trend with the rankings of the STI-only and SAL-only decision trees, suggesting that, if one

wanted to calculate a fixed number of features, there may be value in swapping some of the current STI features with specific saliency features to create an improved pre-screening algorithm.

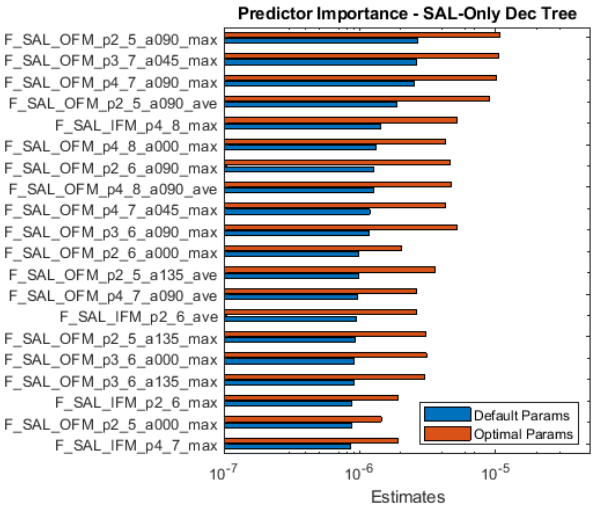

Figure 9 *Predictor importance for DT trained using SAL features. Features are rank ordered using the results of the default training settings. Only the top twenty saliency features are shown.*

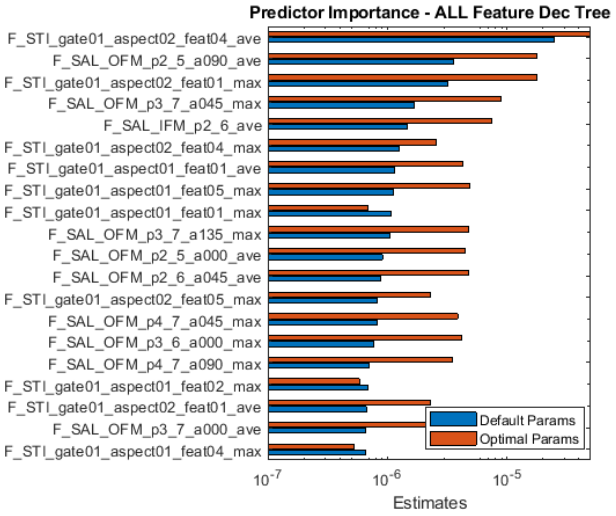

Figure 10 *Predictor importance for DT trained using ALL features. Features are rank ordered using the results of the default training settings. Only the top twenty amongst all the STI and SAL features are shown.*

Conclusions

Using a significantly imbalanced training set to optimize the training parameters for a decision tree caused the resulting classification model to be tuned for high performance on the non-target majority class, while sacrificing performance on the minority target class, which is not desirable for use as a pre-screener for a military target acquisition system. Future efforts will explore methods of sampling the master database of observations such that the number of target and non-target samples are balanced. Methods could include randomly sampling the non-target samples, or using the declaration confidence score during the initial ROI selection to keep a smaller number of high-ranking non-target samples. In either case, it is expected that using a database with balanced classes while optimizing the decision tree training parameters would have the desired effect of improving performance on both classes, not just the majority class.

The feature ranking and selection process described in this paper is one step in an overall approach to using supervised learning techniques to improve the performance of more traditional statistically-based pre-screening algorithms. Future efforts include directly incorporating the most highly ranked features from the STI and SAL feature sets into a new version of the baseline ARL STI pre-screening algorithm. Performance of the improved pre-screener will be evaluated against the baseline using standard binary classifier metrics, including true and false positive rates, as well as any changes in the detection confidence reported by the algorithm.

References

[1] Der, S., Dwan, C., Chan, A., Kwon, H., Nasrabadi, N. (2001). Scale-Insensitive Detection Algorithm for FLIR Imagery. *ARL-TN-175, Army Research Laboratory*.

[2] Chan, A. L. (2003). Multistage infrared target detection. *Optical Engineering*, *42*(9), 2746. https://doi.org/10.1117/1.1593038.

[3] Young, S. S. (2004). Adaptive target detection in forward-looking infrared imagery using the eigenspace separation transform and principal component analysis. *Optical Engineering, 43(8)*, 1767. https://doi.org/10.1117/1.1768534.

[4] Mehmood, A. and Nasrabadi, N. M. (2010). Anomaly detection in wavelet domain for long-wave FLIR imagery. *Proc. of SPIE Vol. 7696*, 76960S. https://doi.org/10.1117/12.850211.

[5] Itti, L., & Koch, C. (2000). A saliency-based search mechanism for overt and covert shifts of visual attention. Vision Research, 40(10–12), 1489–1506. https://doi.org/10.1016/S0042-6989(99)00163-7.

[6] Chen, Y., Sang, N., & Dan, Z. (2013). A saliency-based approach to detection of infrared target (T. Zhang & N. Sang, Eds.; p. 89180S). https://doi.org/10.1117/12.2031336.

[7] Borji, A., Chen, MM., Hou, Q., Jiang, H., Li, J. (2019). Salient Object Detection: A Survey. *Computational Visual Media*. 5, 117-150. https://doi.org/10.1007/s41095-019-0149-9.

[8] Defense Systems Information Analysis Center, https://dsiac.org/databases/atr-algorithm-development-image-database/, Retrieved on 27 Dec 2021.

[9] "ATR Algorithm Development Image Database – Database Overview and User Reference." Version 1.49d, 9 July 2014.

[10] MATLAB. (2021). version 9.10, (R2021a), Natick, Massachusetts: The MathWorks, Inc.

Author Biography

Dawne M. Deaver received a BS in physics from Loyola College (1994), an MS in optics from University of Rochester (1996), and is working toward a PhD at The Catholic University of America. Ms. Deaver is a civilian employee at the U.S. Army Combat Capabilities Development Command, specializing in the development and exploitation of electro-optic/infrared imaging sensors and algorithms to assist Soldiers in the detection and classification of battlefield threats