

Computational Intelligence Methods in Forward-Looking Explosive Hazard Detection

Timothy C. Havens, Derek T. Anderson, Kevin Stone, John Becker
and Anthony J. Pinar

Abstract This chapter discusses several methods for *forward-looking* (FL) *explosive hazard detection* (EHD) using FL *infrared* (FLIR) and FL *ground penetrating radar* (FLGPR). The challenge in detecting explosive hazards with FL sensors is that there are multiple types of targets buried at different depths in a highly-cluttered environment. A wide array of target and clutter signatures exist, which makes detection algorithm design difficult. Recent work in this application has focused on fusion methods, including fusion of multiple modalities of sensors (e.g., GPR and IR), fusion of multiple frequency sub-band images in FLGPR, and feature-level fusion using multiple kernel and iECO learning. For this chapter, we will demonstrate several types of EHD techniques, including kernel methods such as *support vector machines* (SVMs), *multiple kernel learning* MKL, and feature learning methods, including deep learners and iECO learning. We demonstrate the performance of several algorithms using FLGPR and FLIR data collected at a US Army test site. The summary of this work is that deep belief networks and evolutionary approaches to feature learning were shown to be very effective both for FLGPR and FLIR based EHD.

Keywords Sensor fusion • Explosive hazard detection • Aggregation • Multiple kernel learning • Deep learning • Fuzzy integral

T.C. Havens (✉) • J. Becker • A.J. Pinar

Department of Electrical and Computer Engineering/Department of Computer Science,
Michigan Technological University, Houghton, MI 49931, USA
e-mail: thavens@mtu.edu

J. Becker

e-mail: jtbecker@mtu.edu

D.T. Anderson

Department of Electrical and Computer Engineering, Mississippi State University,
Mississippi State, MS, USA
e-mail: anderson@ece.msstate.edu

K. Stone

Department of Electrical and Computer Engineering, University of Missouri,
Columbia, MO, USA
e-mail: kes25c@mail.missouri.edu

© Springer International Publishing Switzerland 2016

R. Abielmona et al. (eds.), *Recent Advances in Computational Intelligence
in Defense and Security*, Studies in Computational Intelligence 621,
DOI 10.1007/978-3-319-26450-9_2

1 Introduction

An important goal for the U.S. Army is remediating the threats of explosive hazards as these devices cause uncountable deaths and injuries to both Civilians and Soldiers throughout the world. Since 2008, explosive hazard attacks in Afghanistan have wounded or killed nearly 10,000 U.S. Soldiers; worldwide, explosive devices on average cause 310 deaths and 833 wounded per month [25]. Systems that detect these threats have included *ground-penetrating-radar* (GPR), *infrared* (IR) and visible-spectrum cameras, and acoustic technologies [9, 10, 37]. Past research has examined both handheld and vehicle-mounted systems and much progress has been made in increasing detection capabilities [7, 14]. *Forward-looking* (FL) systems are an especially attractive technology because of their ability to detect hazards before they are encountered; standoff distances can range from a few to tens of meters. A drawback of forward-looking systems is that they are not only sensitive to explosive devices, *unexploded ordnance* (UXO), and landmines, but also to other objects, both above and below the ground. Because these sensors are standoff sensors, the area being examined for targets is much larger than with downward-looking systems. Thus, clutter is a serious concern. Furthermore, the explosive hazard threat is very diverse—they are made from many different materials, including wood, plastic, and metal, and come in many different shapes and sizes—and this threat continues to evolve. This means that it is nearly impossible to detect explosive hazards solely by a modeling-based approach, and, hence, *computational intelligence* (CI) methods are very appropriate. Previous work has shown that if *forward-looking infrared* (FLIR) or visible-spectrum imagery is combined with L-band FLGPR, *false alarm* (FA) rates can be reduced significantly [2, 16, 18, 19, 44, 45]. Hence, we focus on CI methods for sensor-fused forward-looking detection of explosive threats, comparing CI to other machine learning approaches.

The structure of the remainder of this study is as follows. Section 2.2 describes the preprocessing of the sensor data into a format that is ready for prescreening and feature extraction. The prescreener algorithms are described in Sect. 2.3, and the feature extraction is detailed in Sect. 2.4. In Sect. 3 we describe kernel learning methods, including *support vector machine* (SVM)-based methods, *multiple kernel* (MK) methods, and a fuzzy integral-based MK learner. Methods that learn the features implicitly, such as *deep belief networks* (DBNs), *convolutional neural networks* (CNNs), and iECO feature learning, are described in Sect. 4. Results for the various learning algorithms will be presented in the respective parts of Sects. 3 and 4. We summarize in Sect. 5. Table 1 contains the acronyms used in this chapter. Next, we describe the sensing technologies used to demonstrate the various EHD algorithms in this chapter.

Table 1 Acronyms

UXO	Unexploded ordnance	EHD	Explosive hazard detection
GPR	Ground-penetrating radar	IR	Infrared
FL	forward looking	DL	Downward looking
LW	long-wave	MW	Mid-wave
UTM	Universal traverse mercator	CI	Computational intelligence
FA	False alarm	ROC	Receiver operating characteristic
MK	Multiple kernel	SK	Single kernel
MKLGL	MK learning-group lasso	SVM	Support vector machine
FIMKL	Fuzzy integral MKL	CNN	Convolutional neural network
RBM	Restricted Boltzmann machine	DBN	Deep belief network
CFAR	Constant false-alarm rate	NAUC	Normalized area under the curve
iECO	Improved evolution constructed	CLAHE	Contrast-limited adaptive histogram equalization
HOG	Histogram of oriented gradients	LBP	Local binary patterns
MSER	Maximally stable extramal regions	GMM	Gaussian mixture models
SIFT	Scale-invariant feature transform	AOI	Area of interest

2 Explosive Hazard Detection: Background Knowledge

2.1 Sensing Technologies for FLEHD

FLGPR GPR has long been an interest to the U.S. Army for EHD, and *downward-looking* (DL) systems have been shown to be very effective in operational scenarios. However, DL systems fail to provide a standoff range from the threat; the array is located directly above the threat upon detection. Hence, there has been much focus on improving standoff distances by using FL systems. FLGPR aims to improve standoff by aiming the GPR array forward, often with the center of the beam aimed 10–15 m in front of the vehicle. Since the angle of incidence at which the beam hits the ground surface is important for penetration—the more orthogonal the beam is to the surface, the better the ground penetration—the arrays are usually built on some type of boom above the vehicle. Still, due to the geometry of the FL problem, much array energy is lost to specular reflection from the ground surface. Hence, FLGPR *signal-to-noise ratios* (SNRs) are not nearly as good as with DLGPR systems. Furthermore, the index of refraction of the soil is significantly different than that of the air, which causes a refraction—or bending—of the radar beam at the ground surface, further complicating image formation. These, and other challenges, mean that FLGPR-based EHD is not as simple as looking for local regions of high intensity; more complex EHD strategies are necessary. We talk about several approaches in this chapter.

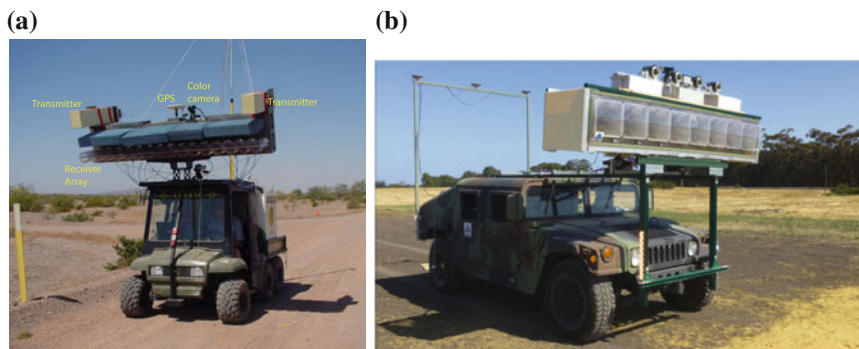


Fig. 1 FLGPRs under research and development for use in EHD. **a** ALARIC L-B and FLGPR. **b** L/X-Band FLGPR

Many FLGPR systems have been designed specifically for EHD, including the two shown in Fig. 1. View (a) shows the ALARIC system, which combines an L-band FLGPR and a visible spectrum imaging system, while (b) shows an FLGPR that combines L- and X-band radar arrays. The FLGPR results shown in this chapter will focus on data recorded with the L/X-band system shown in Fig. 1b. The government-furnished FLGPR data is composed of complex radar data as well as motion data of the vehicle from several lanes at an arid U.S. Army test site.

FLIR While numerous frequency ranges in the infrared portion of the electromagnetic spectrum have been investigated for EHD, e.g., *mid-wave* IR (MWIR) and combinations of IR bands for “disturbed earth” detection, we focus on recent advancements in anomaly detection in *long-wave* IR (LWIR). However, without loss of generality the vast majority of mathematics and algorithms discussed herein are naturally applicable to both MWIR and LWIR imagery with little-to-no change. LWIR or thermal imagers are passive (i.e., they do not require illuminators) and detect infrared radiation in approximately the 8–14 μm wavelength. Objects with a temperature above absolute zero emit infrared radiation in this range at their surface. The amount of emitted thermal radiation increases with temperature. The exact relationship between an object’s temperature and the amount of emitted thermal radiation depends on the emissivity, a quantity representing a material’s ability to emit thermal radiation that varies with wavelength. A thermal imager actually sees not only the emitted radiation of the object, but also transmitted radiation, i.e., radiation from an external source which passes through the object toward the imager, and/or reflected radiation, i.e., radiation from an external source which reflects off the object toward the imager. These factors complicate assigning absolute temperature values to objects. However, in EHD we can exploit the fact that buried objects will likely possess a different thermal conductivity, thermal capacity, or density than the surrounding soil, resulting in either a cooling or warming of the soil immediately surrounding the object. This most often leads to a change in temperature at the surface above the object and results in a measurable change in the amount of emitted

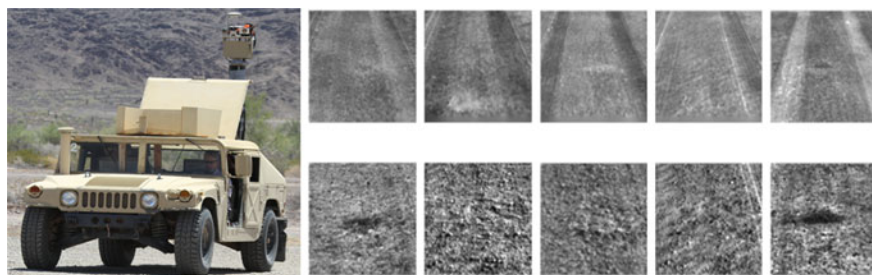


Fig. 2 Example of *thermal scarring* in FLIR with targets of varying difficulty at a fixed vehicle stand off distance. (left) NVESD FLEHD multi-sensor ground vehicle platform, (top row) LWIR and (bottom row) MWIR imagery. Columns are different (center aligned) targets co-registered in MWIR and LWIR. Note, the MWIR camera has a higher resolution (more pixels on target)

thermal radiation compared to areas of the ground free of such objects. Figure 2 shows this phenomenon, referred to in many circles as *thermal scarring*.

However, FLIR is not without flaw. One challenge is diurnal cross-over, the time-period during which the buried object comes to near thermal equilibrium with its surroundings making targets, for all intents, unidentifiable. Another factor is the difference in emitted radiance seen at the soil surface (even for the same soil composition and object) varies based on factors such as the amount of incident thermal radiation, which is dependent on time of day, time of year, and current weather conditions. These are just some of the factors that emphasize the need to include and fuse different sensing technologies to solve this extremely challenging real-world problem.

The FLIR data used in our experiments was collected from two cameras. The first camera, called DVE, was uncooled and used the DRS Infrared Technologies U6000 microbolometer detector which has a spectral response of 8–14 μm . The DVE camera captured 8-bit single channel imagery with a resolution of 640×480 , and horizontal and vertical fields of view of 40 and 30°, respectively. The second camera was a SELEX L20, which produces a 16 bit single channel image with resolution 640×512 . The SELEX camera had a spectral response of 8–10 μm , and horizontal and vertical fields of view of 15 and 12°, respectively. Both cameras were mounted on a mast at the back of the vehicle as shown in Fig. 2. The mast height was approximately 3.35 m and had a downward look angle of 6.3°. An inertial navigation system was mounted next to the cameras, and the time at which each image was captured was recorded. This allowed precise georeferencing using the dense 3D scene reconstruction technique described in [46].

The government-furnished data consists of numerous runs from three lanes at an arid U.S. Army test site. The number of targets per lane varied from 44 to 79, and the area of the lanes ranged from 3,600–4,200 square meters. Emplaced targets were buried between 1–6 in. deep, and varied in metal content (some had no metal).

2.2 Sensor Processing

FLGPR Preprocessing We use a backpropagation procedure to form the radar images (see [15] for a detailed description of the imaging algorithm). In brief, the radar images are formed by coherently summing successive backpropagation images, accounting for platform motion effects on phase and beam pattern effects. The images are formed on a 2.5 cm-spaced grid for each antenna polarization. We also apply a phase correction to the L-band FLGPR to account for vehicle motion during the swept-frequency transmission [4]. The end results of the FLGPR imaging and preprocessing are complex images for each of the L- and X-band polarizations on a rectangular grid coordinate system. In Sect. 2.3, we discuss how we take each FLGPR image $I_p(u, v)$ and indicate candidate detections.

FLIR Preprocessing Numerous algorithms have been applied to the government-furnished FLIR data for preprocessing. However, these algorithms are not the subject of investigation in this chapter as they are not focused on CI. The reader can refer to [3, 42, 43, 46] for more details. In general, these preprocessing algorithms are focused on deinterlacing, denoising, and global or local contrast enhancement. For the DVE images, preprocessing typically consists of deinterlacing, denoising, and *contrast limited adaptive histogram equalization* (CLAHE) [3]. For the SELEX, the 16-bit data was converted to 8-bit by contrast stretching, with saturation limits at 0.05 and 99.95 percent of the original pixel values, so the resulting values filled the entire 16-bit range. After contrast stretching the pixel values were divided by 256 and CLAHE was run. Next we describe how the initial hit locations are determined.

2.3 Prescreeners

Prescreener is a term used for a weak detection scheme by which candidate detections are found and passed on to stronger classification algorithms. The main ideas are to (i) reduce the computational load of the classification algorithms, and (ii) improve classification accuracy by only training on target-like candidate detections.

FLGPR Prescreener The result of the radar preprocessing method described in the Sect. 2.2 is a coherently integrated image $I_p(u, v)$, where (u, v) are the image coordinates: one image for each polarization of the L-band FLGPR (HH and VV polarizations) and one image of the X-band FLGPR (VV polarization). It is well known that penetration depth increases with wavelength; hence, the L-band will have a deeper penetration than the X-band radar. Thus, we use the L-band radar as the detection radar for the method proposed here; although, we will show results for X-band detection and classification too.

The prescreening detector is the first algorithm that indicates candidate detection locations—a block diagram is shown in Fig. 3a. In [15], we proposed two methods to indicate the presence of a target, both of which could be considered to be

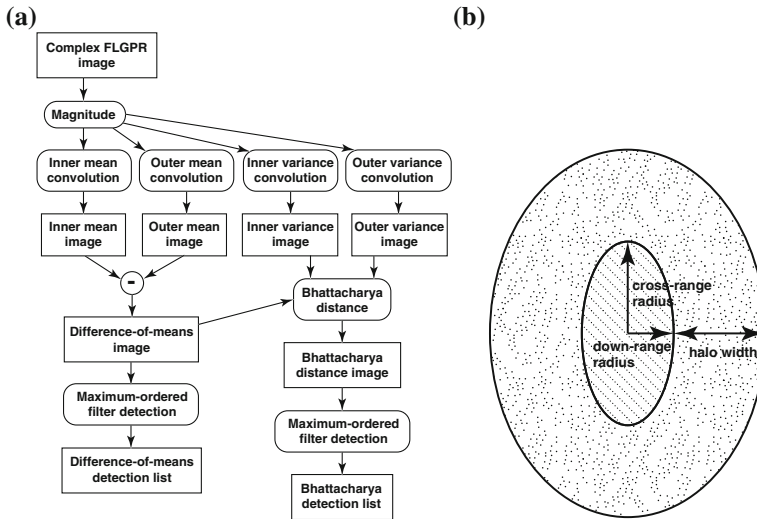


Fig. 3 **a** Block diagram of prescreener detection algorithm. **b** Elliptical convolution kernels used in prescreener. Detection is indicated by comparing the distribution of pixel intensities in inner ellipse to the distribution of pixel intensities in outer halo [15]

a *constant FA rate* (CFAR) detector. The first prescreener indicates a hit by taking the mean of the pixels in the inner ellipse and comparing that to the mean of the pixels in the outer halo (as shown in Fig. 3b). Essentially, the prescreener identifies regions that have values that are higher than the surrounding regions. The second prescreener uses a signed Bhattacharyya distance between the distributions of the pixel values in the center region and outer halo to indicate a hit. For a more detailed description of these prescreeners, see [15]. In our experiments, we have determined the following prescreener parameters to be good choices for this system: down-range radius = 0.25 m; cross-range radius = 0.5 m; and halo width = 0.75 m. These values are related to the impulse-response of the FLGPR system and to expected target sizes. Furthermore, for this chapter we will only present results for the difference-of-means prescreener, which has been shown to be more effective than the Bhattacharyya prescreener for FLGPR data [15].

One could simply threshold the output of the prescreener to indicate a detection; however, this can result in many detections in one local region. Hence, we use a maximum order-filter with a 3 m (cross-range) by 1 m (down-range) rectangular kernel to reduce the presence of closely grouped hits. The prescreeners are rough first-look algorithms for indicating candidate detections—they merely indicate if a region of pixels is different in intensity than the surrounding pixels. They do not, however, consider higher-level features, such as texture or shape, that might indicate better the difference between clutter and true detections. Hence, at each detection location, we then extract a set of shape- or texture-based features, described in Sect. 2.4.

FLIR Prescreener In [2], we outlined a FLIR prescreener for EHD which was later extended to FLGPR in [46]. This prescreener consists of an ensemble of trainable size-contrast (CFAR) filters, i.e., local dual sliding window detectors. Each size-contrast filter has seven parameters: the inner window height and width, the pad height and width (which determine the size of the outer window), a Bhattacharyya distance threshold, a squared difference between the mean values threshold, and three state parameters, referred to as DType (which determines whether the detector will trigger only on bright on dark regions, dark on bright regions, or both). At each pixel, the mean and variance of the inner and outer windows are computed, the Bhattacharyya distance and squared difference between the mean values is calculated and these two values are compared against their corresponding threshold. If both values are greater than their threshold, and the DType condition is met, then the corresponding detector fires. When a detector fires, it projects the inner window center pixel coordinate into UTM coordinates. Next, a clustering algorithm is run on all UTM coordinates generated from individual frames. Specifically, mean-shift, a mode seeking clustering algorithm, with an Epanechnikov kernel is used. Mean-shift was chosen as the application requires a fast clustering algorithm (in the offline training phase, the algorithm has to run hundreds of thousands of times on potentially large data sets: 10,000+ points) that also does not require the user to set the number of clusters. We have compare mean-shift results to the *basic sequential algorithmic scheme* and did not see a significant different in performance. Herein, this clustering step is referred to as spatial mean-shift, and it results in candidate hit locations. Next, mean-shift is run a second time on the hit locations from the combination of multiple frames (this is referred to as temporal mean-shift). Each mean-shift step requires two parameters: the kernel bandwidth and the minimum number of points around a peak in order to keep that cluster. Mean-shift works by performing gradient ascent on the kernel density estimator,

$$\hat{f}(x) = \sum_{i=1}^N K(x_i - x), \quad K(x_i - x) = k(\|x_i - x\|^2), \quad (1)$$

where K is the kernel function, N is the number of data points, and normalizing constants have been omitted for brevity. Taking the gradient of this function with respect to x and setting it to zero results in the following (well known) iterative update equation:

$$x_{t+1} = \frac{\sum_{i=1}^N k'(\|x_i - x_t\|^2) x_i}{\sum_{i=1}^N k'(\|x_i - x_t\|^2)}, \quad (2)$$

where, $k'(x)$ denotes the derivative of $k(x)$ with respect to x , and t denotes the iteration. For the Epanechnikov kernel with bandwidth parameter h , the update equation reduces to:

$$x_{t+1} = \frac{\sum_{x_i \in L} x_i}{|L|}, \quad k_{epn}(v) = \begin{cases} 1 - \frac{v}{h} & 0 \leq v \leq h \\ 0 & \text{else} \end{cases} \quad (3)$$

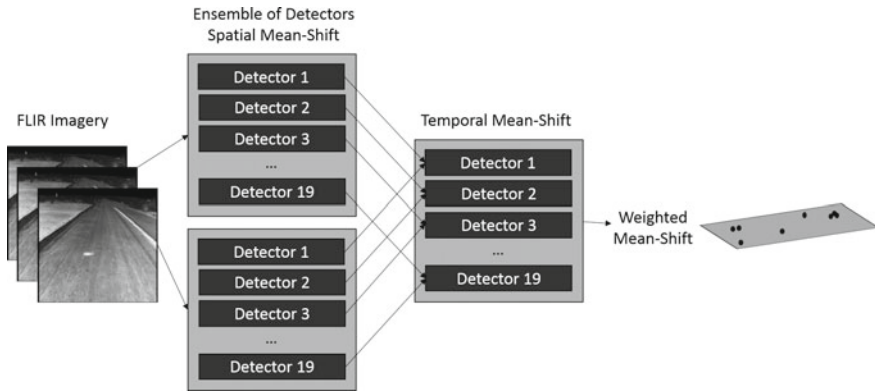


Fig. 4 Illustration of FLIR prescreener, which uses an ensemble of detectors (trained under different criteria) and spatial and temporal weighted mean-shift

where L is the set of all points for which k_{epn} is non-zero and $|\cdot|$ is cardinality. Mean-shift is initialized at every hit location, and the update procedure is run until convergence. For this application, convergence is defined as a change of less than 1 cm between updates (remember that the points are in UTM coordinates). Refer to [46] for additional algorithm speedups. Figure 4 illustrates the proposed FLIR prescreener.

A *genetic algorithm* (GA) is used to learn the detector parameters. To this end, we explored two methodologies. The first, referred to as *one-per-rate*, trains a single detector for each desired *detection rate*. The primary objective of the GA is to achieve the desired detection rate with the secondary objective of minimizing the *false alarm rate* (FAR). In [2], 19 detectors were trained at desired detection rates ranging from 0.05 to 0.95 in step sizes of 0.05. The idea behind training many detectors is that the resulting ROC curve after fusion should be better than if a single detector were trained and only its thresholds allowed to vary. The second method, referred to as *one-per-target*, trains a single detector for each ground truth encounter in the training data. The primary objective of this GA is to detect the specific target with the secondary objective of minimizing the FAR. For both cases, weighted mean-shift is used to fuse the detectors (each trained with a different objective function). A weight is learned for each detector using separable *covariance matrix adaptation evolution strategy* such that the *normalized area under the curve* (NAUC) is maximized on the training data. Reference [2] reports the learned detector parameters and aggregation weights for a prior experiment.

In [46], a few improvements to the above FLIR prescreener were outlined. The first improvement was allowing confidence information to be passed from the size-contrast filter to the spatial mean-shift step and from the spatial mean-shift step to the temporal mean-shift step. Previously, UTM coordinates resulting from a size-contrast filter triggering were treated identically during spatial mean-shift. However, this discards the Bhattacharyya distance and mean difference information which is

useful for locating the strongest response, which generally corresponds to the center of the object. Likewise, information about the peaks found during spatial mean-shift, such as the number of points surrounding each peak, could be useful for the temporal mean-shift step. To remedy this, mean-shift was replaced with weighted mean-shift in both steps, and two new parameters were added to each detector to control whether confidence information is passed on. This leaves it up to the GA to decide if the confidence information is useful. The second improvement was the introduction of a different grouping algorithm as an alternative to weighted mean-shift. The alternative method, also proposed in [46], is an ordered filter approach inspired by the MUFL FLGPR prescreeener introduced in [16]. Lastly, the separable CMA-ES optimization for finding weights for the weighted mean-shift step which combines detectors was eliminated as it tended to overfit the training data. Instead, three heuristics were used to generate weights, and the set of weights which performs best in terms of NAUC on the training data was chosen. The first method assigns equal weight to all detectors; the second method assigns weights based on detection rate and the third method assigns weights based on FAR.

2.4 Feature Extraction

While our FLIR and FLGPR prescreeners achieve relatively high positive detection rates, meaning they often do better than what an expert can identify visually, they still suffer from an unacceptable FAR (relative to U.S. Army requirements). In order to address this deficiency, we have explored, extended and created a number of new image space features and descriptors, including *convolutional neural networks* (CNNs) [43], *improved Evolution CO*nstructed (iECO) features [38], “soft” (importance map weighted) features [42], histogram of cell-structured Gabor energy filter and Shearlet filter bank responses [38, 46], *histogram of gradients* (HOG) [32] and *local binary pattern* (LBP) [15, 17, 35] and “soft” edge histogram descriptor features [2, 46]. In [2], additional anomaly evidence map features in FLIR were proposed, which include features from *maximally stable extremal regions* (MSERs) [33] and *Gaussian mixture models* (GMMs) [41] for change detection. Unlike a CFAR (or size-contrast) filter, which is often utilized as a local contrast feature, the above image space features focus on texture and shape. In addition, we do not use features “directly”, e.g., a single image gradient. Instead, high(er)-level image space descriptors are formed by “pooling” features within a given spatial *area of interest* (AOI), e.g., HOGs, LBPs, or edge descriptors. Furthermore, it is important to not just simply extract features and pool their values over a large spatial AOI as that often leads to ambiguous configurations of patterns. Instead, we preserve the spatial properties of image patterns by using a cell-structured (partially overlapping to allow patterns to drift some in translation across detections) grid for a given AOI. It is usually of great benefit to extract features at different scales in a given AOI, e.g., multi-scale HOG. Convention is to concatenate these multi-scale and multi-cell features together into a single long feature vector of high dimensionality and let a classifier (or fea-

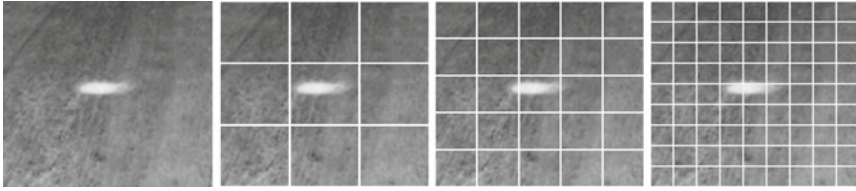


Fig. 5 Multiple cell-structured configurations for feature extraction at a single scale to preserve the spatial context of features. Note that cells are not shown as overlapping for visual simplicity

ture selection algorithm) learn which are most relevant to a particular task at hand. Figure 5 shows the use of multiple cells at a single scale.

The first feature introduced is the LBP. The LBP is a sort of texture or pattern feature and it is calculated at each pixel according to

$$LBP_n = \sum_{k=0}^n s(i_k - i_c) 2^k,$$

where LBP_n is the LBP code, i_c is the window center value, i_k is the value of the k th neighbor and function $s(x)$ is 1 if $x \geq 0$ and 0 otherwise. Ojala extended the LBP for neighborhoods of different shapes and sizes [35]. The circular (radius r) neighborhood version, $LBP_{n,r}$ includes bilinearly interpolating values at non-integer image coordinates. Ojala also observed that there is a limited number of transitions or discontinuities in the circular presentation of 3×3 texture patterns and that these uniform patterns, $LBP_{n,r}^u$, are fundamental properties of local image texture, meaning they provide the vast majority of all patterns (accounting for 90 % at $u = 2$). The u stands for no more than u 0–1 or 1–0 transitions, e.g., 00011110 has 2 transitions and 00101001 has 5 transitions. Last, the LBP is turned into a descriptor by binning the patterns into a histogram over an AOI. For example, for $u = 2$ there are only 59 patterns (thus histogram bins) for a neighborhood of size 8. In addition, Ojala put forth a rotation robust version that consists of shifting the binary patterns until there is a 1 in the first digit [35]. This reduces the number of patterns for a neighborhood size of 8 to only 9. Last, most normalize the resultant histogram by its ℓ_1 or ℓ_2 -norm.

Another feature is the famous HOG, popularized by David Lowe in the *scale invariant feature transform* (SIFT); however it was first explored by Edelman in the context of wet science and later popularized by Dalal-Triggs for HOG-based person detection [11]. It is important to note that SIFT technically consists of keypoint detection, a feature descriptor and detection. The HOG (the feature descriptor in SIFT) involves the extraction of a gradient vector per pixel in an image. For a given AOI, one computes the magnitude of each gradient, $\|(\partial I(x, y)/\partial x, \partial I(x, y)/\partial y)\|$, and its respective orientation. A histogram of B bins (a user defined or learned parameter) is specified and each pixel's gradient magnitude, per cell, is added to the bin with respect to its orientation. For example, for 360° and 8 bins each bin spans 45° and for a cell structured configuration of 4×4 we obtain a 128-length feature vector. Note,

convention involves bilinearly interpolating each gradient magnitude for the closest and next closest bin. Also, while SIFT identifies and then rotates the descriptor with respect to its major orientation bin(s), this is an optional step that the user must determine relative to the given detection task at hand. In our FLEHD investigations, we do not perform the rotation step.

In [46], we proposed a “soft” edge histogram descriptor feature. The edge histogram descriptor is inspired by the MPEG-7 edge histogram descriptor, which has five simple convolution operators that represent vertical, horizontal, diagonal, anti-diagonal and non-directional edge classes. The operators for the first four classes closely resemble the standard Sobel and Prewitt edge operators. At each pixel, the five operators are applied and the absolute value of the response to each is computed. The pixel is assigned to the class of the operator generating the largest response. In [46], we extended this feature to make it less sensitive to noise. We allow a pixel to contribute to all classes by creating a histogram at each pixel location and we accumulate the individual pixel histograms inside a window to form the final descriptor. A pixel’s histogram is constructed by computing the absolute value of the response to each of the edge convolution operators and then dividing each of those values by the sum, i.e. taking the l_1 norm. Linear interpolation is performed to distribute the pixel’s contribution between the edge classes and the non-edge class by comparing the sum of the absolute values of the operator responses to the edge threshold. If the sum is greater than or equal to the edge threshold then the non-edge class is assigned zero. Otherwise, the non-edge class is assigned one minus the fractional value of the sum divided by the edge threshold, and that fractional value is multiplied to the value of each of the edge classes in the histogram. We introduced a further change, the addition of two new edge masks; making the total descriptor length seven. We extract two edge histogram descriptors per cell using edge thresholds of 15 and 35. Therefore, edge histogram descriptor gives $7 \times 2 = 14$ features per cell.

In [40, 42], we created a softened version of the HOG, LBP, and edge histogram descriptor based on the extraction and utilization of an *importance map*. An importance map, one per each image, is simply a $[0, 1]$ -valued image that is the same size as the original image. Each pixel in an importance map informs us about the relevance or significance of that pixel for a given task at hand. The importance map is used to weight features, such as HOGs and LBPs, as they are added to a descriptor like a histogram. The motivation for importance maps is that current image space descriptors unfortunately extract both background (e.g., clutter, tire tracks, foliage, etc.) and foreground (target) information. In many cases, the number of encountered foreground features are extremely few relative to the background information and their presence in the descriptor can be dwarfed. Most researchers ignore this fact and pass the problem down the processing pipeline. That is, most extract all features in an AOI and leave it up to the classifier or feature selection to determine what is important. Instead, our goal is to extract feature-rich information in target areas and more-or-less ignore extraneous information in other parts of an AOI. In [40], Scott and Anderson used this philosophy and showed improvement in aircraft detection in remote satellite imagery across different parts of the world and times of the year based on importance-weighted multi-scale texture and shape descriptors. Their

importance maps were based on fuzzy integral-based fusion of differential morphological map profiles for soft object extraction. In [42], we extend this technique to FLEHD, introducing a new way to derive an importance map for FLIR. In FLIR, we are interested in detecting circular or elliptical (due to perspective deformation in FL imagery) shapes for anomaly detection. Hence, we exploited this information and created a frequency and orientation selective bank of Gabor energy filters, which we later reduced down to a single Shearlet filter, to build an importance map. The real-valued Gabor or Shearlet image is normalized between min and max across an AOI. It is then blurred with a Gaussian kernel to spread out the filter response, as many features reside at or around the edges of an object. The result is then re-normalized, according to its min and max, back into $[0, 1]$ (values that represent the relative worth of different pixels in the AOI relative to the task at hand). The soft HOG, LBP, and edge histogram descriptor features are calculated as before, however as these features are being added to their respective bins in the histogram they are multiplied by their corresponding per-pixel importance map weights $E(x, y)$. The features that we describe in this section can now be used to further reduce the number of FAs by training classifiers to indicate prescreeener hits as either FAs or true-positives. Next we discuss kernel methods that can accomplish this task.

3 Kernel Methods for EHD

Consider some non-linear mapping function $\phi : \mathbf{x} \rightarrow \phi(\mathbf{x}) \in \mathbb{R}^{D_K}$, where D_K is the dimensionality of the transformed feature vector \mathbf{x} . With kernel clustering, we do not need to explicitly transform \mathbf{x} , we simply need to represent the dot product $\phi(\mathbf{x}) \cdot \phi(\mathbf{x}) = \kappa(\mathbf{x}, \mathbf{x})$. The kernel function κ can take many forms, with the polynomial $\kappa(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + 1)^p$ and *radial-basis-function* (RBF) $\kappa(\mathbf{x}, \mathbf{y}) = \exp(\sigma \|\mathbf{x} - \mathbf{y}\|^2)$ being two of the most well known. Given a set of n objects X , we can thus construct an $n \times n$ kernel matrix $K = [K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)]^{n \times n}$. This kernel matrix K represents all pairwise dot products of the feature vectors associated with n objects in the transformed high-dimensional space—called the *Reproducing Kernel Hilbert Space*.

The main goal of kernel methods is to transform the feature vectors \mathbf{x} such that the new representations, $\phi(\mathbf{x})$, are advantageous to the classification problem. We present three methods for learning classifiers in kernel spaces, SVM, MKLGL, and FIMKL, which we now describe.

3.1 Single Kernel

One of the most popular kernel methods for classification is the SVM. The SVM attempts to find an optimal separating hyperplane between two classes of training data; for the case of EHD, we use it to find a hyperplane between features that describe FAs and those of true positives. For a detailed description of the SVM,

see [8]. The *single-kernel SVM* (SKSVM) is defined as

$$\max_{\alpha} \left\{ \mathbf{1}^T \alpha - \frac{1}{2} (\alpha \circ \mathbf{y})^T K (\alpha \circ \mathbf{y}) \right\}, \quad (4a)$$

subject to

$$0 \leq \alpha_i \leq C, \quad i = 1, \dots, n; \quad \alpha^T \mathbf{y} = 0, \quad (4b)$$

where \mathbf{y} is the vector of class labels, $\mathbf{1}$ is the n -length vectors of 1s, $K = [\kappa(\mathbf{x}_i, \mathbf{x}_j)] \in \mathbb{R}^{n \times n}$ is the kernel matrix, and \circ indicates the Hadamard product [5]. The value C determines how many errors are allowed in the training process [8]. Note that SKSVM reduces to the linear SVM for the kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$ (which is simply the Euclidean dot product).

One of the drawbacks of using the above SVM formulation is that it treats each datum equally; hence, when there is an imbalance between the number of datum in each class, then the SVM decision boundary is driven primarily by the data from the class with more data points. This is a problem in explosive hazards detection as there are typically many more FA detections than there are true positives—the true positives only comprise a small overall area of the lane. To attack this issue, we use a formulation of the SVM for imbalanced data which uses a different error cost for positive (C^+) and negative (C^-) classes. Specifically, we change the constraints of the kernel SVM formulation at (4) to

$$0 \leq \alpha_i \leq C^+, \forall i | y_i = +1; \quad 0 \leq \alpha_i \leq C^-, \forall i | y_i = -1; \quad \alpha^T \mathbf{y} = 0; \quad (5)$$

where C^+ is the error constant applied to the positive class and C^- is the error constant applied to the negative class. In our application, the positive class is true positives and the negative class is FAs. We set $C^+ = n^-/n^+$ and $C^- = 1$, where n^- is the number of objects in the negative class and n^+ is the number of objects in the positive class. This essentially allows for fewer errors in the true positive class.

We use LIBSVM to efficiently solve the SKSVM problem [6]. The output of LIBSVM is a classifier model that contains the vector α and the bias b . A measured feature vector \mathbf{x} can be classified by computing

$$y = \text{sgn} \left[\sum_{i=1}^n \alpha_i y_i \kappa(\mathbf{x}_i, \mathbf{x}) - b \right], \quad (6)$$

where sgn is the signum function. We now show the application of SKSVM to our FLEHD problem.

Application of SKSVM to FLGPR EHD Figure 6 shows selected results of training the SKSVM on FLGPR lanes A, B, and D, and testing on Lane C. The results are compared to random performance, which is the ROC achieved by uniform random selection of hit locations at given FA rates. View (a) shows the prescreeener ROC

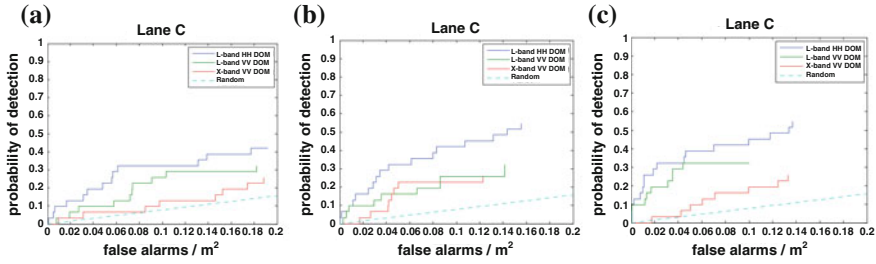


Fig. 6 ROC curves showing testing performance of (a) FLGPR prescreener, and SKSVM classifier with RBF kernel for (b) single HOG feature and (c) combination of HOG and LBP features. Percent NAUC improvements are shown for each of the L-band (HH and VV polarizations) and X-band FLGPRs. The performance of a uniform random detector is shown by the *dotted line*. **a** Prescreener. **b** L-HH: 21 %; L-VV: 32 %; X: 64 %. **c** L-HH: 20 %; L-VV: 36 %; X: 7 %

curve for Lane C for the three FLGPR sensors, while views (b) and (c) show the results of using the SKSVM classifier to reject FAs. The kernel used for this experiment is the RBF kernel, which is well-known to be effective for most data. View (b) shows the ROC curve using only the HOG feature, while view (c) shows the results when combining the HOG and LBP features. As the figure illustrates, the SKSVM is able to reduce the number of FAs significantly. Interestingly, the combination of features is detrimental to SKSVM performance for the X-band FLGPR. This is because the addition of the LBP feature to the SKSVM for the X-band radar results in over-training (the training or resubstitution results are nearly perfect), which negatively affects the test lane performance.

3.2 Multiple Kernel

MKL extends the idea of kernel classification by allowing the use of combinations of multiple kernels. The kernel combination can be computed in many ways, as long as the combination is a Mercer kernel [34]. In this chapter we assume that the kernel K is composed of a weighted combination of pre-computed kernel matrices, i.e.,

$$K = \sum_{k=1}^m \sigma_k K_k, \quad (7)$$

where there are m kernels and σ_k is the weight applied to the k th kernel. The composite kernel can then be used in the chosen classifier model; we will use the SVM. Thus, MKL SVM extends the SKSVM optimization at (4) by also optimizing over the weights σ_k .

$$\min_{\sigma \in \Delta} \max_{\alpha} \left\{ \mathbf{1}^T \alpha - \frac{1}{2} (\alpha \circ \mathbf{y})^T \left(\sum_{k=1}^m \sigma_k K_k \right) (\alpha \circ \mathbf{y}) \right\}, \quad (8a)$$

subject to (typically)

$$0 \leq \alpha_i \leq C, \quad i = 1, \dots, n; \quad \alpha^T \mathbf{y} = 0, \quad (8b)$$

where Δ is the domain of σ . Note that this is the *same* problem as SKSVM if the kernel weights are assumed constant [28]. This property has been used by many researchers to propose *alternating optimization* procedures for solving the min-max optimization problem. That is, solve the inner maximization for a constant kernel K , and then update the weights σ_k to solve the outer minimization, and repeat until convergence. We use the optimization procedure proposed by Xu et al. called MKL *group lasso* (MKLGL) [47]. This method is efficient as it uses a closed-form (i.e., non-iterative) solution for solving the outer minimization in (8a);

$$\sigma_k = \frac{f_k^{2/(1+p)}}{\left(\sum_{k=1}^m f_k^{2p/(1+p)} \right)^{1/p}}, \quad k = 1, \dots, m, \quad (9a)$$

$$f_k = \sigma_k^2 (\alpha \cdot \mathbf{y})^T K_k (\alpha \cdot \mathbf{y}), \quad (9b)$$

where p is the norm on the domain constraint, $\|\sigma\|_p = 1, p > 1$.

We further modify the MKLGL algorithm, as we did for SKSVM, to allow for unbalanced classes—i.e., we apply the constraints C^+ and C^- as shown at (5). The MKLGL training algorithm is outlined in Algorithm 1. The MKLGL is simple to implement and is efficient as the update equations for σ_k are closed-form. MKL can be thought of as a classifier fusion algorithm. It can find the optimal kernel among a set of candidates by automatically learning the weights on each kernel. The individual kernels can be computed in many ways—see our previous papers on this topic for more discussion on the formation of the kernel matrices [15, 17].

Algorithm 1: MKLGL Classifier Training [47]

Data: (\mathbf{x}_i, y_i) - feature vector and label pairs; K_k - kernel matrices

Result: α, σ_k - MKLGL classifier solution

Initialize $\sigma_k = 1/m, k = 1, \dots, m$ (equal kernel weights)

while not converged do

- Solve unbalanced SKSVM for kernel matrix $K = \sum_{k=1}^m \sigma_k K_k$
 - Update kernel weights by Eq. (9)
-

Application of MKLGL to FLGPR EHD The MKLGL algorithm is applied in the same way as the SKSVM—it acts to classify prescreeener hits as either FAs or true positives. Figure 7 shows results of the MKLGL classifier using an ensemble of RBF

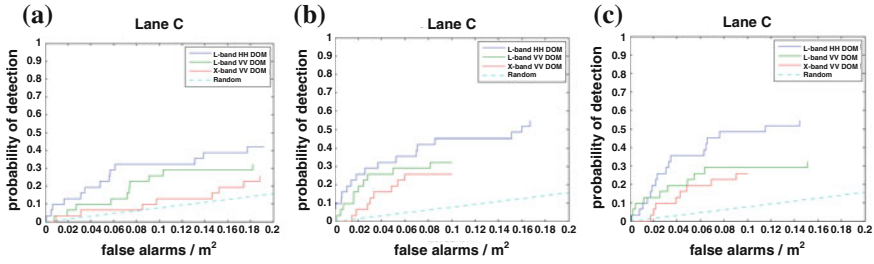


Fig. 7 ROC curves showing testing performance of (a) FLGPR prescreener, and MKLGL classifier for (b) single HOG feature and (c) combination of HOG and LBP features. Percent NAUC improvements are shown for each of the L-band (HH and VV polarizations) and X-band FLGPRs. The performance of a uniform random detector is shown by the dotted line. **a** Prescreener. **b** L-HH: 21 %; L-VV: 46 %; X: 67 %. **c** L-HH: 21 %; L-VV: 47 %; X: 67 %

kernels on the same training and testing lanes as shown for SKSVM in Fig. 6. The NAUC results show that the MKLGL is able to match and sometimes improve upon the results obtained using the SKSVM. The MKLGL improvement of the L-band VV NAUC is especially noteworthy.

3.3 Fuzzy Integral-Based Multiple Kernel (FIMKL)

The *Fuzzy Integral-based* MK (FIMKL) [22, 23] extends MKL by using a non-linear aggregation operator, the *fuzzy integral* (FI). The fusion of information using the Sugeno or Choquet FI has a rich history; for a recent review, see [1]. Depending on the problem domain, the input to the FI can be experts, sensors, features, similarities, pattern recognition algorithms, etc. The FI is defined with respect to the *fuzzy measure* (FM), a monotone and often normal capacity. With respect to a set of m information sources, $X = \{x_1, \dots, x_m\}$, the FM encodes the (often subjective) *worth* of each subset in 2^X . For a finite set of sources, X , the FM is a set-valued function $g : 2^X \rightarrow [0, 1]$ with the following conditions:

1. (Boundary condition) $g(\emptyset) = 0$,
2. (Monotonicity) If $A, B \subseteq X$ with $A \subseteq B$, then $g(A) \leq g(B)$.

Note, if X is an infinite set, there is a third condition guaranteeing continuity and we often assume $g(X) = 1$ (although it is not necessary in general). Numerous FI formulations have been proposed to date for generalizability, differentiability, and to address different types of uncertain data [1]. In [22, 23], we investigated the Sugeno and Choquet FIs for MKL. We proposed a solution based on sorting at the *matrix level*. Assume each kernel matrix K_k has a numeric “quality.” This can be computed, for example, by computing the classification accuracy of a base-learner that uses kernel K_k (or by a learning algorithm like a GA). Let $v_k \in [0, 1]$ be the k th kernel’s *quality*. These qualities can be sorted, $v_{(1)} \geq v_{(2)} \geq \dots \geq v_{(m)}$. Given m base Mercer

kernels, $\{\kappa_1, \dots, \kappa_m\}$, FM g , and a sorting $v_{(1)} \geq v_{(2)} \geq \dots \geq v_{(m)}$, the difference-in-measure Choquet FI is computed by

$$\mathcal{K}_{ij} = \sum_{k=1}^m (G_{\pi(k)} - G_{\pi(k-1)})(K_{\pi(k)})_{ij} = \sum_{k=1}^m \omega_k (K_{\pi(k)})_{ij}, \quad i, j \in \{1, \dots, n\}, \quad (10)$$

where $\omega_i = (G_{\pi(i)} - G_{\pi(i-1)})$, $G_{\pi(i)} = g(\{x_{\pi(1)}, \dots, x_{\pi(i)}\})$, $G_{\pi(0)} = 0$, and $\pi(i)$ is a sorting on X such that $h(x_{\pi(1)}) \geq \dots \geq h(x_{\pi(m)})$. The MK formulation at (10) produces a Mercer kernel as multiplication by positive scalar and addition are *positive semidefinite* (PSD) preserving operations. Since (10) involves per-matrix sorting, it can be compactly written in a simpler (linear algebra) form, i.e., $\mathcal{K} = \sum_{k=1}^m \omega_k K_{\pi(k)}$.

Prior works in MKL rely on the relatively *linear convex sum* (LCS) formulation. It is often desired due to its advantage in optimization, e.g., MKLGL. Both FIMK and LCS MK are of type convex sum, i.e., $w_k \in \mathfrak{R}_+^m$ and $\sum_{k=1}^m w_k = 1$. However, one is linear, the other is not, and the weights are derived from the FM. The Choquet FI is capable of representing a much larger class of aggregation operators. For example, it is well known that the Choquet FI can produce, based on the selection of FM, the maximum, minimum, *ordered weighted average* (OWA), order statistics, etc. However, the machine learning LCS form is simply m weights anchored to the individual inputs. The LCS is a subset (one of the aggregation operators) of the FI.

In [22, 23], we reported improved SVM accuracies and lower standard deviations over the state-of-the-art MKLGL on publicly available benchmark data. We proposed a GA, called FIGA, based on learning the densities for the Sugeno λ -FM. In that work we demonstrated that the GA approach is more effective than MKLGL, even in light of the fact that our GA approach used far fewer component kernels. In particular, the FIGA approach achieved a mean improvement of nearly 10 % over MKLGL on the Sonar data set. The performance of FIGA comes at a cost though, as MKLGL is much faster in terms of actual running time than FIGA. We also saw that FIGA using a combination of FM/FIs is somewhat more effective than the FIGA LCS form. These findings are not surprising as our intuition tells us that the nonlinear aggregation allowed by the FM/FI formulation is more flexible than just the LCS aggregation; hence, these results reinforce our expectation. Overall, these results are not surprising as different data sets require different solutions, and while an LCS may be sufficient for a given problem, it may not be appropriate for a different problem. Also, it should be noted that the FM/FI formulation includes LCS aggregation as a subset of its possible solutions; hence, when LCS is appropriate the FM/FI aggregation can mimic the LCS. In summary, the learner (GA vs GL) appears to be the most important improvement factor, followed by a slight improvement by using the nonlinear FM/FI aggregation versus LCS. While FIMKL has not been applied to date for EHD, this computational intelligence method is reviewed as it is an improvement to classical MKL and stands to be of relevance and benefit to EHD.

4 Deep Learners and Feature Learning for EHD

Deep learning architectures were initially designed to mimic the human brain, more specifically, the neocortex [36]. This part of the brain has been shown to have six layers and a forward-backward structure to classify image data collected by the eye [26]. In brief, deep learning architectures extend “shallow” neural networks by adding multiple hidden layers—these additional layers act as generalized feature detectors. Deep learning algorithms have been shown to perform very well on a variety of classification tasks, such as facial recognition [29], document classification [30], and speech recognition [39]. We will present results for two types of deep learning architectures: *deep belief networks* (DBNs) and *convolutional neural networks* (CNNs).

4.1 Deep Belief Networks

DBNs are a type of deep learning network formed by stacking *Restricted Boltzmann Machines* (RBMs) in successive layers to reduce dimensionality, making a compressed representation of the input. DBNs are trained layer by layer using greedy algorithms and information from the previous layer. In this subsection, we will first discuss RBMs and how to train them, then move on to training DBNs.

RBMs are simple binary learners that consist of two layers: one visible and one hidden. The visible layer is the input layer and typically consists of an n -length vector of normalized values. The hidden layer is the feature representation layer. The defining equation of the RBMs is the energy equation,

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{b}^T \mathbf{v} - \mathbf{c}^T \mathbf{h} - \mathbf{v}^T W \mathbf{h}, \quad (11)$$

where \mathbf{v} is the input vector, \mathbf{h} is the hidden feature vector, \mathbf{b} and \mathbf{c} are the visible and hidden layer biases, respectively, and W is the weight matrix that connects the layers. It should be noted that weights only exist between the hidden and visible layers, that is to say, that the nodes in either layer are not interconnected. \mathbf{v} is the input and used to train hidden layer \mathbf{h} as

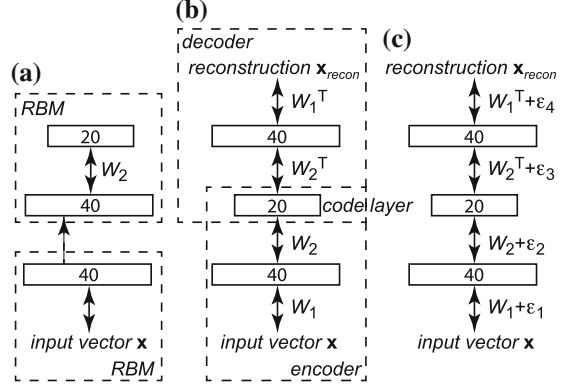
$$\mathbf{h} = \sigma(\mathbf{c} + W^T \mathbf{v}). \quad (12)$$

The hidden layer is then used to reconstruct the visible layer in the same manner,

$$\mathbf{v}_{recon} = \sigma(\mathbf{b} + W \mathbf{h}). \quad (13)$$

The reconstruction of the visible layer \mathbf{v}_{recon} is then used in (12) to form \mathbf{h}_{recon} and then the weight update is calculated as

Fig. 8 Illustration of DBN training: numbers in rectangles indicate the number of neurons in each layer. **a** Pretraining. **b** Unrolling. **c** Fine tuning



$$\Delta W = \epsilon ([\mathbf{v}\mathbf{h}^T]_{data} - [\mathbf{v}\mathbf{h}^T]_{recon}), \quad (14)$$

where ϵ is the learning rate. Iterated over several epochs, this weight update performs a type of gradient descent called contrastive divergence [36].

To form a DBN, layers of RBMs are stacked as shown in Fig. 8a, where the hidden layer of the lower RBM becomes the input/visible layer of the next RBM. Once the input RBM is trained, its reconstructed hidden layer \mathbf{h}_{recon} is used to create the visible layer of the next RBM by

$$\mathbf{v}_{n+1} = \sigma(\mathbf{c}_n + W_n^T \mathbf{h}_{recon,n}) \quad (15)$$

where n denotes the layer number. The $(n+1)$ th RBM is now trained and this cycle is repeated for the number of layers desired. After all layers have been trained, the DBN is typically then mirrored to make an encoder-decoder as shown in Fig. 8b [21]. An input to the encoder-decoder thus produces a reconstruction of itself, where

$$\text{encoder: } \mathbf{x}_{n+1} = W_n \mathbf{x}_n; \quad (16a)$$

$$\text{decoder: } \mathbf{x}_{recon,n-1} = W_{n-1}^T \mathbf{x}_{recon,n}; \quad (16b)$$

and $\mathbf{x}_1 \in \mathbb{R}^d$ is the input vector and $\mathbf{x}_{recon,1} = \mathbf{x}_{recon} \in \mathbb{R}^d$ is the reconstruction. Note that the final hidden layer in the encoder is the first layer in the decoder, $\mathbf{x}_{n+1} = \mathbf{x}_{recon,n+1}$, where n is the number of RBMs in the DBN. Fine-tuning of the weight matrices can be performed as shown in Fig. 8c. This fine-tuning is often done using stochastic gradient descent (backpropagation) or Hinton's *up-down* algorithm [20]. Note that this gives the DBN more flexibility as the weight matrices are adjusted for each of the encoder and decoder.

Application of DBNs to FLGPR EHD To apply DBNs to the FLEHD problem, we take the extracted features from each prescreener hit location in the training data and apply the DBN to learn the representation of the FAs; this is due to the imbalance between the number of FA and target examples in the training data. The reconstruction *root mean-square error* (RMSE),

$$RMSE = \sqrt{\sum_{i=1}^d (x_i - x_{recon,i})^2}, \quad (17)$$

of the DBN is thus a measure of how well an input feature vector matches to the learned representation of the FAs—true positives ideally have high RMSE and false positives ideally have low RMSE. Hence, the RMSE can be directly used as the confidence of a true positive in the ROC curve. The DBNs for the results here are trained on three lanes of data and then tested on a separate lane (in essence, 4-fold cross-validation).

Since DBNs are flexible in their construction, we tested many different architectures, learning rates, and epoch limits. The best DBN we found for overall EHD performance was a network that uses two hidden layers of sizes 40 and 20, giving a full encode-decode stack architecture of $[\mathbf{x} \ 40 \ 20 \ 40 \ \mathbf{x}_{recon}]$, where \mathbf{x} is the $d \times 1$ input feature vector and \mathbf{x}_{recon} is the $d \times 1$ reconstruction (see Fig. 8). The learning rate is 0.9, and 30 epochs of contrastive divergence was used for RBM training.

Several combinations of features were tested with the DBN classifier. Figure 9 illustrates selected results from our comprehensive evaluation of DBNs for FLGPR EHD. These ROC curves show the performance of the DBN classifier on Lane C (training on Lanes A, B, and D). The percent NAUC improvements clearly show that the DBN significantly improves NAUC, by up to 85% for the case of the X-band FLGPR using HOG & LBP features (note that the X-band FLGPR also has the most room for improvement in this case).

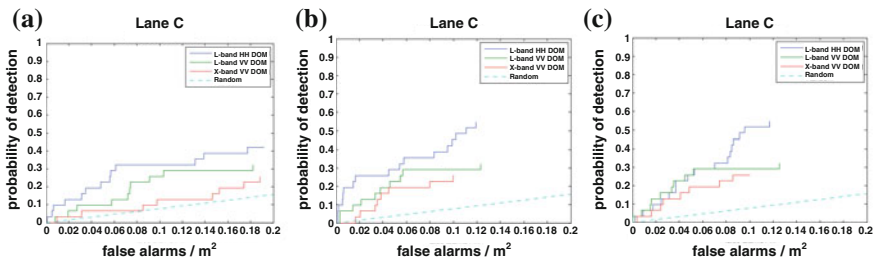


Fig. 9 ROC curves showing testing performance of **a** FLGPR prescreener, and DBN classifiers for **b** single HOG feature and **c** combination of HOG and LBP features. Percent NAUC improvements are shown for each of the L-band (HH and VV polarizations) and X-band FLGPRs. The performance of a uniform random detector is shown by the dotted line. **a** Prescreener. **b** L-HH: 28 %; L-VV: 52 %; X: 53 %. **c** L-HH: 23 %; L-VV: 52 %; X: 85 %

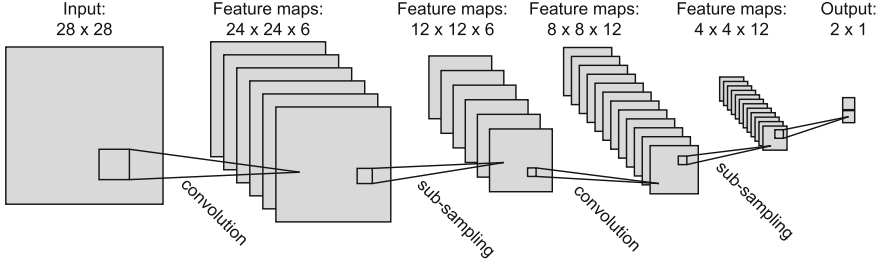


Fig. 10 Illustration of a convolutional neural network [31, 36]

4.2 Feature Learning

Convolutional Neural Networks CNNs are a type of neural network with a unique architecture. Inspired by the visual system, these networks consist of alternating convolutional and sub-sampling layers. The convolutional layers generate feature maps by convolving kernels over the data from the previous layers and then the sub-sampling layers downsample the feature maps [36]. CNNs work directly on the 2D data as opposed to most other forms of deep networks which reorganize the data into 1D feature vectors. Figure 10 illustrates a convolutional neural network.

The l th convolutional layer is generated from a j th feature map by

$$a_j^l = \sigma(b_j^l + \sum_{i \in M_j^l} a_j^{l-1} * k_{ij}^l), \quad (18)$$

where σ is the activation function, usually hyperbolic tangent or sigmoid, b_j^l is a scalar bias, M_j^l is an index vector of feature maps i in layer $l-1$, $*$ is the 2D convolution operator and k_{ij}^l is the kernel used on map i in layer $l-1$. A sub-sample layer l is generated from a feature map j by

$$a_j^l = \text{down}(a_j^{l-1}, N^l), \quad (19)$$

where down is a down-sampling function, such as mean-sampling, that down-samples by factor N^l [36]. The output layer is then generated by

$$o = f(\mathbf{b}^o + W^o \mathbf{x}_v), \quad (20)$$

where \mathbf{x}_v denotes a feature vector concatenated from the feature maps of the previous layer, \mathbf{b}^o is a bias vector, and W^o is a weight matrix. The parameters to be learned are thus k_{ij}^l , b_j^l , \mathbf{b}^o and W^o . Gradient descent is used to learn these parameters and this can be efficiently performed through the use of convolutional backpropagation [36].

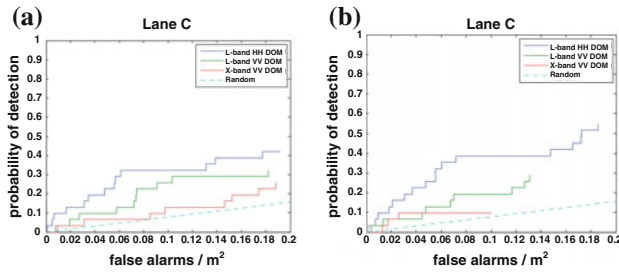


Fig. 11 ROC curves showing testing performance of **a** FLGPR prescreener, and **b** CNN classifier using the HOG feature. Percent NAUC improvements are shown for each of the L-band (HH and VV polarizations) and X-band FLGPRs. The performance of a uniform random detector is shown by the dotted line. **a** Prescreener. **b** L-HH: 13 %; L-VV: 1.4 %; X: 28 %

Application of CNNs to FLGPR EHD Unlike the SVM, MKLGL, and DBN, the CNN operates on 2D feature maps. Fortunately, the HOG, LBP, and FFST are all 2D features and thus can be used as input for the CNN; we also used the raw image data (imagelet) surrounding each prescreener hit as input to the CNN. The output of the CNN is a 2-element vector—one element to indicate FA and one to indicate true positive. As shown in Fig. 10, we use two convolutional layers and two subsampling layers. The learning rate was 0.9 and 350 epochs were used for training, which were shown to be good choice in a more comprehensive parameter study we performed. Figure 11 shows selected results from our comprehensive evaluation of CNNs for FLGPR EHD. These ROC curves show the performance of the CNN classifier on Lane C (training Lanes A, B, and D). As is evidenced by the percent NAUC improvement values, the CNN is the least effective of the classifiers that we have applied to the FLGPR EHD. Furthermore, many of the results (which we do not show) that we compiled using the CNN were very poor. Hence, we do not recommend the CNN at this time for FLGPR EHD.

Application of CNNs to FLIR EHD In [43], CNNs were evaluated for EHD in FLIR imagery. Image chips extracted at prescreener alarm locations were fed directly as input to a CNN. CNN classification results were compared to a baseline algorithm which extracted five hand-engineered feature sets and performed classification using a SVM. Due to the lack of training data for training a conventional deep CNN model, two alternative CNN approaches were explored.

The first approach used a deep CNN model pre-trained on the ImageNet dataset [12]. This model is referred to herein as DPT-CNN, and was made available through the open source python package DeCAF [13]. DPT-CNN uses the architecture proposed by Krizhevsky et al. in [27], which won the *ImageNet Large Scale Visual Recognition Challenge 2012* (ILSVRC2012). The architecture consists of five convolutional layers, some followed by *Rectified Linear Unit* (ReLU) activation, response normalization, and max pooling, followed by three fully connected layers. The last fully connected layer is fed to a 1000-way softmax function.

DPT-CNN was trained on the ILSVRC2012 training data, which consisted of more than 1.2 million training images from 1000 object classes. It was shown in [13] that values from intermediate layers of this pre-trained network work well as features for new vision tasks. Specifically, tasks with small amounts of training data, where a deep CNN trained directly performed poorly. For our tests, the alarm image chips were input to DPT-CNN, and intermediate values were saved at six stages. These intermediate values were used to train a SVM which was evaluated the same way as the baseline algorithm. The first two sets of intermediate values came from the second and first fully connected layers after ReLU activation. These are referred to as FC7-ReLU and FC6-ReLU, respectively. The ReLU activation takes the form $\phi(v) = \max(0, v)$. The next two intermediate values, POOL5 and CONV5, came from the last convolutional layer. CONV5 is before max-pooling, and POOL5 is after pooling. The last two sets, RNORM1 and POOL1, came from the first convolutional layer. RNORM1 is after pooling and response normalization. POOL1 is after pooling but before response normalization.

The fully connected layer outputs, which no longer convey spatial position, were not expected to be useful for this EHD task since position in the image chip is extremely important. The POOL5 and CONV5 features do retain some spatial information. The RNORM1 and POOL1 features retain more, but since they are not deep features they may not be as descriptive. Table 2 shows the NAUC results for the DPT-CNN features, as well as for the baseline algorithm and the best individual baseline feature, for a three lane leave-one-lane-out cross validation test using two FLIR cameras.

As expected, the fully connected layer features did not perform well. Performance improved significantly when moving to the POOL5 features, and again when moving to the CONV5 features. The CONV5 features compared well with the top performing hand-engineered image feature, multi-scale HOG, even outperforming it on Lane A. Surprisingly, the POOL1 features scored better overall than the CONV5 features on the DVE camera image chips, but show a pronounced drop on the SELEX image chips.

Table 2 DPT-CNN: DVE/Selex cameras: NAUC at 0.01 FA/m²

Feature	All lanes	Lane A	Lane B	Lane C
FC7-ReLU	0.435/0.451	0.321/0.355	0.420/0.418	0.556/0.573
FC6-ReLU	0.479/0.469	0.353/0.365	0.480/0.454	0.598/0.582
POOL5	0.557/0.501	0.404/0.386	0.600/0.500	0.658/0.609
CONV5	0.615/0.566	0.471/0.423	0.655/0.604	0.712/0.662
RNORM1	0.623/0.525	0.454/0.389	0.709/0.553	0.699/0.624
POOL1	0.624/0.519	0.458/0.386	0.710/0.545	0.695/0.617
BASE: HOG	0.645/0.584	0.453/0.421	0.722/0.615	0.753/0.708
BASE: ALL	0.677/0.610	0.496/0.445	0.766/0.652	0.762/ 0.727
CONV5 + BASE	0.676/0.607	0.508/0.449	0.748/0.649	0.764/0.714

*Bold indicates best result for each camera and lane

The DPT-CNN results indicated that a deep CNN model was not necessary to achieve good performance on this FLIR EHD task. This was not particularly surprising since the task requires little in the way of translation, scale, or orientation invariance. The primary difficulty is intra-class variation. Thus, a second CNN approach using a shallow CNN trained directly using the image chips was pursued. The shallow architecture consists of a single convolutional layer followed by an output layer containing a single neuron followed by the sigmoid activation function. The output of the sigmoid was used as the alarm confidence value. For all experiments, weights were learned using *stochastic gradient descent* (SGD) with momentum and the cross entropy error function. To address class imbalance, for each training pattern presentation an example was chosen randomly from either the true target class or the false alarm class with equal probability. Evaluation was performed using the same methodology as for DPT-CNN.

In [24], Jarrett et al. found that the single most important factor for recognition accuracy in a CNN model, considering architecture choices such as activation function, sub-sampling type, and response normalization, was the use of a rectifying non-linearity. While they used the *absolute value function* (AVF), the ReLU in Krizhevskys architecture performs a similar operation. Therefore, the first experiment evaluated performance when using either no non-linearity, ReLU, or AVF following the convolutional layer. These results are presented in Table 3. Both activations improved performance. AVF performed better than ReLU, and was chosen for further experiments.

We next investigated forcing the convolutional filters to have zero-mean and zero-phase. The intuition being that only the non-dc frequency characteristics are important, and that shifting of the output is meaningless for classification. To enforce these characteristics, transformation functions were inserted before the variables in question were used. The transformation functions modify their inputs to enforce the desired constraint. During SGD learning, derivatives are propagated through the transformations. For example, if the original convolutional layer is $OUTPUT = CONV(INPUT, X)$, to enforce zero-mean for the kernel X the expression becomes $OUTPUT = CONV(INPUT, G(X))$, where $G(X)$ modifies X to have the zero-mean characteristic. No significant performance improvement was seen from enforcing either constraint.

Table 3 Rectifying Nonlinearity: DVE Camera: NAUC at 0.01 FA/m²

Convolution filter radius	None, # filters			ReLU, # filters			AVF # filters		
	4	8	16	4	8	16	4	8	16
3	0.492	0.482	0.503	0.519	0.517	0.499	0.555	0.573	0.566
5	0.508	0.510	0.509	0.550	0.552	0.565	0.600	0.602	0.603
7	0.487	0.483	0.475	0.555	0.545	0.537	0.596	0.580	0.592

*Bold indicates best result for each filter radius

Table 4 Learning in freq domain: DVE camera: NAUC at 0.01 FA/m²

Convolution filter radius	Spatial—# of filters			Frequency—# of filters		
	4	8	16	4	8	16
3	0.555	0.573	0.566	0.636	0.636	0.640
5	0.600	0.602	0.603	0.617	0.619	0.618
7	0.596	0.580	0.592	0.614	0.613	0.619

*Bold indicates best result for each filter radius

We then experimented with learning the convolutional filters’ frequency domain representations instead of their spatial domain representations. This was done by using the inverse FFT as a transformation function. Table 4 shows the results for learning the convolutional filters in the frequency domain versus the spatial domain. Zero-mean and zero-phase were enforced in the frequency domain. A slight performance improvement was seen across all combinations.

Based on these results, shallow CNN networks with eight zero-mean, zero-phase filters learned in the frequency domain were scored on the DVE and SELEX data. Table 5 shows the per lane results for various kernel radii, as well as the DPT-CNN and baseline results for comparison. Overall, the shallow CNN results were very similar to those of DPT-CNN. The shallow CNN achieved a slightly better overall result on DVE, and a slightly worse overall result on SELEX when comparing to the CONV5 features of DPT-CNN. When comparing to the POOL1 and RNORM1 features, the shallow CNN SELEX result is much better. The baseline algorithm, which includes features that cannot be expressed via convolution, outperforms both CNN approaches.

iECO Feature Learning In [38], the algorithm *improved Evolutionary CO*nstructed (iECO) feature descriptors (referred to hereafter as simply iECO) was put forth for FLIR-based EHD. The iECO algorithm is a feature learning technique that looks to

Table 5 Shallow CNN: NAUC at 0.01 FA/m²

	DVE camera				SELEX camera			
	All lanes	Lane A	Lane B	Lane C	All lanes	Lane A	Lane B	Lane C
CNN radius 3	0.635	0.464	0.734	0.700	0.562	0.397	0.626	0.656
CNN radius 5	0.616	0.478	0.694	0.670	0.559	0.413	0.611	0.645
CNN radius 7	0.612	0.460	0.697	0.673	0.557	0.409	0.628	0.626
Pre- trained CNN	0.624	0.458	0.710	0.695	0.566	0.423	0.604	0.662
Baseline	0.677	0.496	0.766	0.762	0.610	0.445	0.652	0.727

*Bold indicates best result

exploit important cues in data that often elude non-learned (often referred to as “hand crafted”) features such as HOGs, LBPs and edge histogram descriptors. Each hand crafted feature is ultimately an attempt to more-or-less sculpt (force) a signal/image into some predisposed mathematical framework which may or may not reveal the information that a user/system needs. Instead of coming to the table with a limited set of tools and trying to make everything look like a nail, iECO learns the tool based on the task at hand.

While the field of deep learning has demonstrated state-of-the-art performance, the ECO (and iECO respectively) work of Lillywhite et al. has the advantage over CNNs of interpretability (it is not a black box) and it does not predispose the solution to that of convolution. At its core, ECO is the GA-based learning and (ensemble-based) use of a population of chromosomes that are compositions of functions (image processing transformations). Each chromosome is of variable length and the goal is to learn the image transformations and respective parameters relative to some task. An advantage of this approach, versus CNNs, is that it makes use of a relatively wide set of different heterogeneous image transformations to seek a new tailored solution. In [38] we used 19 different image transformations which range from a Harris corner detector to a square root, Hough circle, median blur, rank transform, LoG, mathematical morphology and Shearlet and Gabor spatial frequency domain filtering, just to name a few. In many cases, emergent behaviour arises and the chromosomes can be manually examined and studied, potentially revealing additional domain information such as what features or physics in IR or GPR are most important for a task like EHD. Figure 12 shows the iECO process (not learning, but application of iECO to a given AOI).

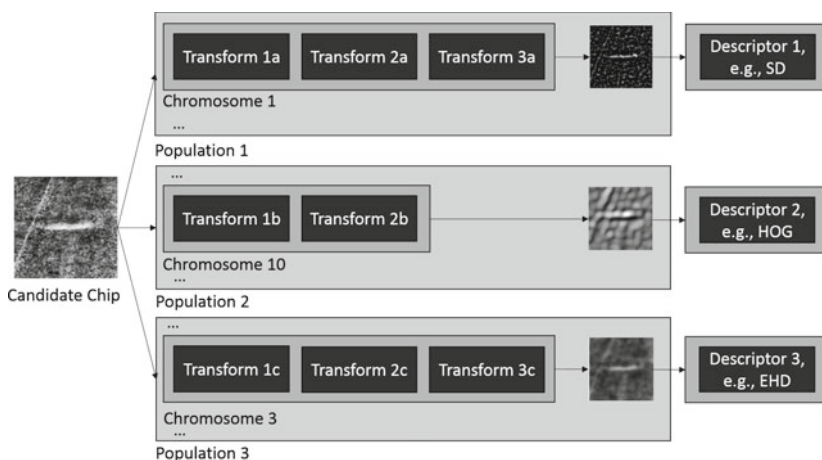


Fig. 12 iECO applied to a prescanner hit in FLIR. Learned iECO chromosomes, in different populations, are applied to the input image. Finally, different descriptors are extracted relative to each transformed image

In iECO, we address a shortcoming of the ECO features– the so-called “features” which are the unrolling of image pixels into a single vector. ECO suffers from the curse of dimensionality and the naive unrolling does not intelligently take into account various spatial and scale cues. In [38], we extract ECO features relative to different high-level descriptors and cell-structured configurations. Specifically, we explored the HOG, EHD and statistical features; which include the local mean, standard deviation, kurtosis, l_2 -norm, and the difference between the local values and their corresponding global values. A separate GA population is maintained and a separate search is conducted relative to each high-level descriptor. iECO, like CNN learning is not a computationally trivial task. As a result, we have not yet attempted to learn the different populations in a single simultaneous algorithm. Furthermore, in [38] we showed that each descriptor learns/prefers different chromosomes that have varying fitness values. With respect to classification, we experimented with taking a single best chromosome per descriptor (highest fitness), taking the top 50 % of chromosomes relative to each descriptor, and the identification of the top 5 most diverse chromosomes (which is currently a manual process). Our results indicate that the concatenation of multiple chromosome features leads to improved performance. Furthermore, we showed that if one pipeline is applied to a different descriptor than it was learned for, then the result is a significant drop in performance (fitness). This is interesting as it tells us that iECO appears to learn a tailored pre-processing of imagery relative to each descriptor in order to better highlight salient information. Figure 13 shows different learned iECO pipelines.

In [38], we introduced constraints on each individual’s chromosome to help promote population diversity and prevent infeasible solutions. This allows us to search for quality solutions faster and it typically results in shorter length chromosomes that are computationally simpler to realize (which is important for a real-time causal EHD system). In ECO, there are no direct mechanisms incorporated into the GA,

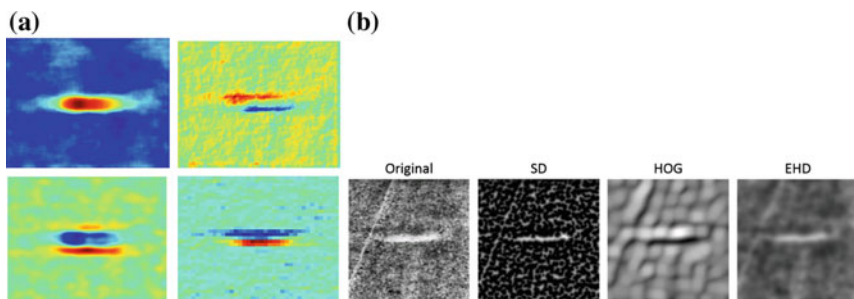


Fig. 13 iECO on FL LWIR. **a** Average iECO output of four chromosomes across 50 different buried targets. Each image is scaled to [0, 1] for visual display and they are shown in Matlab jetmap color coding, where *blue* is 0 and *red* is 1. These images show that diversity exists across chromosomes and different aspects of targets are learned, e.g., local contrast, orientation specific edge information, etc. **b** Output of highest fitness chromosomes for each descriptor for a single target. These images show that each descriptor prefers a different iECO pipeline. **a** Average iECO output. **b** Different iECO populations

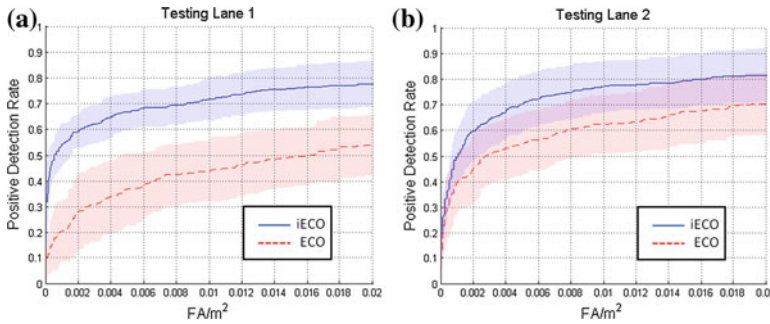


Fig. 14 Vertically averaged ROC curves with 95 % confidence intervals. **a** Testing lane 1. **b** Testing lane 2

outside of mutation, to promote diversity in the population. In [38], we introduced diversity promoting constraints that consider the uniqueness and complexity of the ECO's search space. We designed a set of diversity promoting constraints that define what percentage of the population is allowed overlapping genes at each layer of the individual's gene segment. Next, we addressed the issue of the occurrence of the same gene back-to-back. Such a scenario is undesirable, e.g., it does not typically make sense to perform a rank transform back-to-back. In addition, this increases the computational complexity of the system as a consequence of the unnecessary image transforms. We combat this by collapsing consecutive uses of the same gene type, i.e., if any gene occurs more than once consecutively then only the first occurrence is retained. Elitism is used in iECO.

In summary, in [38] we showed that the above diversity promoting constraints and the combination of high-level image descriptors leads to the discovery of significantly higher quality solutions for EHD. We showed that iECO continuously identifies higher performance solutions, i.e., an impressive drop in the FAR for a given PDR, populations are more diverse, which was verified manually, and the resultant chromosomes are significantly shorter and thus give rise to a simpler system (computationally and memory utilization-wise) to realize. Figure 14 is ROC results for iECO versus ECO features. iECO clearly outperforms ECO.

5 Conclusions

This chapter described the EHD problem and various methods for preprocessing, prescreening, and false rejection for FLGPR and FLIR. The methods discussed for FLGPR-based EHD were SKSVM, MKLGL, DBNs, and CNNs. The best overall detection and classification method for the FLGPR was the DBN using a combination of HOG and LBP features, showing up to 85 % improvement in NAUC. The weakest FLGPR-based method was the CNN. In the future, we are going to investigate more advanced CNN architectures and training methods for CNNs as applied to FLGPR

EHD. Several EHD methods were discussed for FLIR-based detection, including baseline SVM detection, CNNs, and iECO feature learning. Several CNN architectures were tested. While the CNN architectures showed promise, especially those that use frequency-domain AVF filters, the baseline SVM-based feature-fusion approach outperformed the CNN. Lastly, iECO feature learning was demonstrated for FLIR-based EHD. In the future, we aim to further apply our fuzzy integral-based multiple kernels methods for EHD as FIMKL has been shown to be superior to MKLGL for benchmark data sets. We also aim to extend the deep learning approaches for online and active learning for EHD.

Acknowledgments This work is funded in part by a National Institute of Justice grant (2011-DN-BX-K838), U.S. Army (W909MY-13-C0013, W909MY-13-C0029) and Army Research Office (W911NF-14-1-0114 and 57940-EV) in support of the U.S. Army RDECOM CERDEC NVESD. Superior, a high performance computing cluster at Michigan Technological University, was used in obtaining results presented in this work.

References

1. Anderson, D.T., Havens, T.C., Wagner, C., Keller, J., Anderson, M.F., Wescott, D.J.: Extension of the fuzzy integral for general fuzzy set-valued information. *IEEE Trans. Fuzzy Syst.* **22**(6), 1625–1639 (2014)
2. Anderson, D.T., Stone, K., Keller, J.M., Spain, C.: Combination of anomaly algorithms and image features for explosive hazard detection in forward looking infrared imagery. *IEEE J. Sel. Topics Appl. Earth Obs. Remote Sens.* **5**(1), 313–323 (2012)
3. Anderson, D.T., Stone, K., Keller, J.M., Rose, J.: Anomaly detection ensemble fusion for buried explosive material detection in forward looking infrared imaging for addressing diurnal temperature variation. In: *Proceedings of the SPIE*, vol. 8357, p. 83570T (2012)
4. Becker, J., Havens, T.C., Pinar, A., Schulz, T.J.: Deep belief networks for false alarm rejection in forward-looking ground-penetrating radar. In: *Proceedings of the SPIE* (2015)
5. Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: *ACM Workshop on COLT*, pp. 144–152 (1992)
6. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. *ACM Trans. Intell. Sys. Tech.* **2**(3), 1–27 (2011)
7. Collins, L.M., Torriente, P.A., Throckmorton, C.S., Liao, X., Zhu, Q.E., Liu, Q., Carin, L., Clodfelter, F., Frasier, S.: Algorithms for landmine discrimination using the NIITEK ground penetrating radar. *Proc. SPIE.* **4742**, 709–718 (2002)
8. Cortes, C., Vapnik, V.N.: Support-vector networks. *Mach. Learn.* **20**(3), 273–297 (1995)
9. Costley, R.D., Sabatier, J.M., Xiang, N.: Forward-looking acoustic mine detection system. *Proc. SPIE.* **4394**, 617–626 (2001)
10. Cremer, F., Chavemaker, J.G., deJong, W., Schutte, K.: Comparison of vehicle-mounted forward-looking polarimetric infrared and downward-looking infrared sensors for landmine detection. *Proc. SPIE.* **5089**, 517–526 (2003)
11. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005. *CVPR 2005*. vol. 1, pp. 886–893 (2005)
12. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: a large-scale hierarchical image database. In: *IEEE Conference on Computer Vision and Pattern Recognition*, 2009. *CVPR 2009*, pp. 248–255 (2009)

13. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. CoRR abs/1310.1531 (2013), <http://arxiv.org/abs/1310.1531>
14. Gader, P.D., Grandhi, R., Lee, W.H., Wilson, J.N., Ho, K.C.: Feature analysis for the NIITEK ground penetrating radar using order weighted averaging operators for landmine detection. In: Proceedings of the SPIE, vol. 5415, pp. 953–962 (2004)
15. Havens, T.C., Becker, J.T., Pinar, A.J., Schulz, T.J.: Multi-band sensor-fused explosive hazards detection in forward-looking ground-penetrating radar. In: Proceedings SPIE, vol. 9072, p. 90720T (2014)
16. Havens, T.C., Ho, K.C., Farrell, J., Keller, J.M., Popescu, M., Ton, T.T., Wong, D.C., Soumekh, M.: Locally adaptive detection algorithm for forward-looking ground-penetrating radar. In: Proceedings of the SPIE, vol. 7664, p. 76442E (2010)
17. Havens, T.C., Keller, J.M., Stone, K., Ho, K.C., Ton, T.T., Wong, D.C., Soumekh, M.: Multiple kernel learning for explosive hazards detection in FLGPR. In: Proceedings of the SPIE, vol. 8357, p. 83571D (2012)
18. Havens, T.C., Spain, C.J., Ho, K.C., Keller, J.M., Ton, T.T., Wong, D.C., Soumekh, M.: Improved detection and false alarm rejection using ground-penetrating radar and color imagery in a forward-looking system. In: Proceedings of the SPIE, vol. 7664, p. 76441U (2010)
19. Havens, T.C., Stone, K., Keller, J.M., Ho, K.C.: Sensor-fused detection of explosive hazards. In: Proceedings of the SPIE, vol. 7303, p. 73032A (2009)
20. Hinton, G., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. *Neural computation* **18**(7), 1527–1554 (2006)
21. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. *Science* **313**(5786), 504–507 (2006)
22. Hu, L., Anderson, D.T., Havens, T.C.: Multiple kernel aggregation using fuzzy integrals. In: IEEE International Conference Fuzzy Systems, pp. 1–7 (2013)
23. Hu, L., Anderson, D.T., Havens, T.C., Keller, J.M.: Efficient and scalable nonlinear multiple kernel aggregation using the choquet integral. In: Laurent, A., Strauss, O., Bouchon-Meunier, B., Yager, R. (eds.) *Information Processing and Management of Uncertainty in Knowledge-Based Systems. Communications in Computer and Information Science*, vol. 442, pp. 206–215. Springer (2014)
24. Jarrett, K., Kavukcuoglu, K., Ranzato, M., LeCun, Y.: What is the best multi-stage architecture for object recognition? In: 2009 IEEE 12th International Conference on Computer Vision, pp. 2146–2153 (2009)
25. JIEDDO COIC MID: Global IED monthly summary report (2012)
26. Kandel, E.R., Schwartz, J.H., Jessell, T.M.: *Principles of Neural Science*, 4 edn. McGraw-Hill, New York (2000)
27. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. Curran Associates, Inc. (2012)
28. Lanckriet, G.R.G., Cristianini, N., Bartlett, P., Ghaoui, L.E., Jordan, M.I.: Learning the kernel matrix with semidefinite programming. *J. Mach. Learn. Res.* **5**, 27–72 (2004)
29. Lawrence, S., Giles, C.L., Tsoi, A.C., Back, A.D.: Face recognition: a convolutional neural-network approach. *IEEE Trans. Neural Netw.* **8**(1), 98–113 (1997)
30. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. *Proc. IEEE* **86**(11), 2278–2324 (1998)
31. LeCun, Y., Jackel, L.D., Bottou, L., Brunot, A., Cortes, C., Denker, J.S., Drucker, H., Guyon, I., Müller, U., Säckinger, E., Simard, P., Vapnik, V.: Comparison of learning algorithms for handwritten digit recognition. In: International conference on artificial neural networks, vol. 60 (1995)
32. Lowe, D.G.: Object recognition from local scale-invariant features. In: International Conference Computer Vision, pp. 1150–1157 (1999)
33. Matas, J., Chum, O., Urban, M., Pajdla, T.: Robust wide-baseline stereo from maximally stable extremal regions. *Image Vis. Comput.* **22**(10), 761–767 (2004)

34. Mercer, J.: Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. R. Soc. A* **209**, 441–458 (1909)
35. Ojala, T., Pietikainen, M., Harwood, D.: A comparative study of texture measures with classification based on featured distributions. *Patt. Recognit.* **29**(1), 51–59 (1996)
36. Palm, R.B.: Prediction as a candidate for learning deep hierarchical models of data. Ph.D. thesis, Technical University of Denmark (2012)
37. Playle, N., Port, D.M., Rutherford, R., Burch, I.A., Almond, R.: Infrared polarization sensor for forward-looking mine detection. *Proc. SPIE*. **4742**, 11–18 (2002)
38. Price, S.R., Anderson, D.T., Luke, R.H.: An improved evolution-constructed (iECO) features framework. In: *IEEE Symposium Series on Computational Intelligence* (2014)
39. Sarikaya, R., Hinton, G.E., Ramabhadran, B.: Deep belief nets for natural language call-routing. In: *IEEE International Conference Acoustics, Speech and Signal Processing*, pp. 5680–5683 (2011)
40. Scott, G.J., Anderson, D.T.: Importance-weighted multi-scale texture and shape descriptor for object recognition in satellite imagery. In: *IEEE International Geoscience and Remote Sensing Symposium*, pp. 79–82 (2012)
41. Stauffer, C., Grimson, W.E.L.: Adaptive background mixture models for real-time tracking. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1999, vol. 2 (1999)
42. Stone, K., Keller, J.M.: Clutter rejection by cluster analysis in an automatic detection system for buried explosive hazards in forward looking imagery. In: *Proceedings of the SPIE* (2013)
43. Stone, K., Keller, J.M.: Convolutional neural network approach for buried target recognition in FL-LWIR imagery. In: *Proceedings of the SPIE* (2014)
44. Stone, K., Keller, J.M., Ho, K.C., Gader, P.D.: On the registration of FLGPR and IR data for the forward-looking landmine detection system and its use in eliminating FLGPR false alarms. In: *Proceedings of the SPIE*, vol. 6953 (2008)
45. Stone, K., Keller, J.M., Popescu, M., Havens, T.C., Ho, K.C.: Forward-looking anomaly detection via fusion of infrared and color imagery. In: *Proceedings of the SPIE*, vol. 7664, p. 766425 (2010)
46. Stone, K.E., Keller, J.M., Anderson, D.T., Barclay, D.B.: An automatic detection system for buried explosive hazards in fl-lwir and FL-GPR data. In: *Proceedings of the SPIE Conference Detection and Sensing of Mines, Explosive Objects, and Obscured Targets* (2012)
47. Xu, Z., Jin, R., Yang, H., King, I., Lyu, M.R.: Simple and efficient multiple kernel learning by group lasso. In: *Proceedings of the Interence Conference Machine Learning*, pp. 1175–1182 (2010)

Recent Advances in Computational Intelligence in
Defense and Security

Abielmona, R.; Falcon, R.; Zincir-Heywood, N.; Abbass,
H.A. (Eds.)

2016, XI, 752 p. 236 illus., 72 illus. in color., Hardcover

ISBN: 978-3-319-26448-6