EMADE for Intelligence Surveillance and Reconnaissance Applications

Jason Zutty, Rodd Talebi, Austin Dunn, Roland Samuelson, Gregory Rohling
USA

jason.zutty@gtri.gatech.edu, rodd.talebi@gtri.gatech.edu, austin.dunn@gtri.gatech.edu,
roland.samuelson@gtri.gatech.edu, greg.rohling@gtri.gatech.edu

ABSTRACT

Researchers from the fields of computer vision and deep learning, have been using image processing techniques to detect, recognize, and identify objects of interest within overhead imagery. However, as with all machine learning problems, researchers tend to have a limited scope of methods to apply (defined by their background) and a limited amount of time to tune their models.

In recent years, the field of automated machine learning (autoML) has quickly attracted a significant amount of attention both in academia and industry. The driving force is to reduce the amount of human intervention required to process data and create models for classification and prediction, a tedious and arbitrary process for a data scientist that often does not result in a global optimum with respect to accuracy or other metrics.

Our entry into the field is EMADE, the Evolutionary Multi-objective Algorithm Design Engine, which affords several benefits not found in other autoML solutions including the ability to stack machine learning models, process time-series data using dozens of signal-processing techniques, and efficiently evaluate algorithms on multiple objectives.

We show several modifications to EMADE to take it from only supporting processing feature and time-series data to processing more computationally-intense high-resolution imagery. To demonstrate EMADE's capabilities, we leverage the xView data challenge (http://www.xviewdataset.org), a dataset of overhead imagery comprised of 1 million object instances of 60 different classes. We show results on using EMADE to produce improved algorithms for the xView data challenge over algorithms created by researchers for ISR applications. One such solution produced by EMADE is a histogram of oriented gradients on a canny edged detected wavelet transform of a 224x224 image chip. When scored as a binary classifier for buildings, this evolved algorithm achieved a specificity of 97.2% and a sensitivity of 94.3%.

INTRODUCTION

The field of intelligence, surveillance, and reconnaissance (ISR) is currently ripe for automation with machine learning. A large portion of ISR is the identification of potential points, persons, or objects of interest in feeds of imagery. Screening this imagery is a mundane and repetitive task currently handled by scores of analysts who are hand-annotating image data. Machine learning can serve this industry to supplement the analysts by serving as a prescreening tool, drawing attention to objects of interest. Because much of their burden is relieved, an analyst is able to be more productive and less prone to errors.

Researchers from the fields of computer vision and deep learning, have been using image processing techniques to detect, recognize, and identify objects of interest within overhead imagery. However, as with all machine learning problems, researchers tend to have a limited scope of methods to apply (defined by their background) and a limited amount of time to tune their models.

In recent years, the field of automated machine learning (autoML) has quickly attracted a significant amount of attention both in academia and industry. The driving force is to reduce the amount of human intervention required to process data and create models for classification and prediction, a tedious and arbitrary process for a data scientist that often does not result in a global optimum with respect to accuracy or other metrics.

AutoML frameworks such as TPOT [1], s-mGP-ML and t-mGP-ML [2], and AutoStacker [3] all feature unique ways of applying genetic programming (GP) in order to evolve machine learning pipelines. None of these, however, support the ingestion of imagery that would be needed in order to enter the field of ISR. The only autoML framework to presently support working with imagery is Google Cloud AutoML (https://cloud.google.com/automl/), which is currently in its alpha stage.

Our entry to the field is EMADE, the Evolutionary Multi-objective Algorithm Design Engine, which affords several benefits not found in other autoML solutions including the ability to stack machine learning models, process time-series data using dozens of signal-processing techniques, and efficiently evaluate algorithms on multiple objectives.

In this paper we show several modifications to EMADE to take it from only supporting processing feature and time-series data to processing more computationally-intense high-resolution imagery. Our modifications include support for batch learning, integration of deep neural network models, and improved caching of intermediary results. The methods shown here can be adapted to other autoML frameworks in order to more efficiently evaluate algorithms on imagery applications. We also show EMADE's greatest strength is its ability to leverage well-tested methods in newly evolved solutions, and the ability to build off these existing algorithms guarantee better results.

ANALYSIS OF EXISTING METHODS

The techniques (computer vision) used to perform such tasks fall into two major schools of thought: the use of standard machine learning classifiers (bag of visual words, support vector machines, etc) with emphasis in preprocessing the images or video frames; and the use of convolutional neural networks with the emphasis rather on the architecture of the network itself.

The motivation behind the more traditional approach is that statistical methods will be used to first isolate and detect objects (preprocessing) and then pass the object through some machine learner (classifier). Sample methods include simple thresholdings [4], histogram of oriented gradients [5], Gaussian mixture models [6], kernel density estimations [7], and Bayesian classifiers [8]. Despite being statistically and mathematically validated, these methods lack robust performance in accuracy and speed.

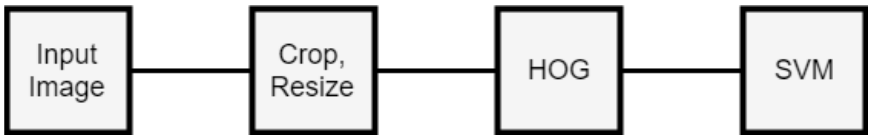

Figure 1. Block diagram for processing imagery using the Histogram of Oriented Gradients (HOG) method.

A popular method, Figure 1, is with histogram of oriented gradients (HOG) and support vector machines (SVM). First step is to crop and resize the training and testing data to the same image ratio and dimensions. Then apply HOG to the frame to capture how and where the pixel values change throughout; this will serve as the frame features that we will then pass through the SVM to build a classification model.

In 2012, with the introduction and success of AlexNet [9], which used a large and deep convolutional neural network (CNN) architecture with nonlinear activation function ReLU (rectified linear unit), the field of computer vision shifted to fully adopt CNNs. Neural network frameworks were developed, massive image and video datasets were made public for training, and benchmarks were established to push the state of the art in multi-object localization, segmentation, classification, and tracking while also minimizing computation time. Despite the incredible success and attention that CNNs have brought to the field, the design of the network architectures still remains relatively subjective and experimental.

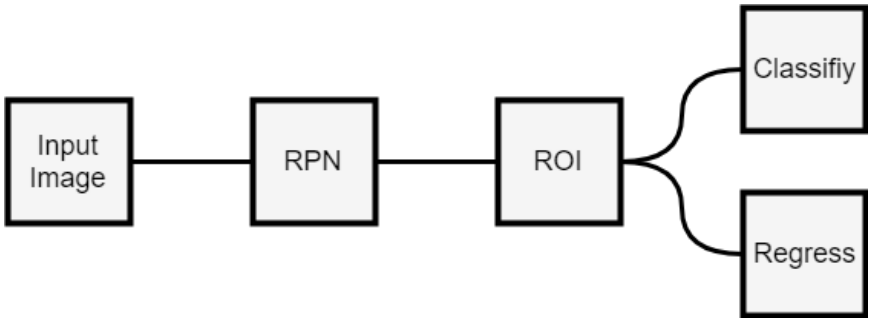

Figure 2. Example of process flow for Faster R-CNN algorithm.

A successful object detector and classifier for video data is Faster R-CNN [10]. This network, Figure 2, can be broken down into an ensemble of other networks: a Region Proposal Network (RPN) to gather cropped frames with various ratios and dimensions that have high likelihood of containing a foreground object, a Region of Interest (ROI) Pooling to reduce overlapping regions into a fixed feature map, a classifier to determine the label of the proposed region, and a regressor to improve how well the object was bounded in the image.

The recent paradigm shift of computer vision accelerated growth in many relevant applications like that of autonomous vehicles, yet the success in aerial imagery and video remains lacking. One of the unique challenges is the large and small scales of the images and of the objects, respectively. A typical image covers several square kilometers with tens of millions of pixels (>10MB) while any single object can fill up to 50 pixels total. Many papers have attempted to adapt various computer vision techniques to the problem of aerial imagery [11] [12] [13] [14], but they are limited in scope to higher-altitude satellite imagery, as opposed to lower-altitude surveillance video.

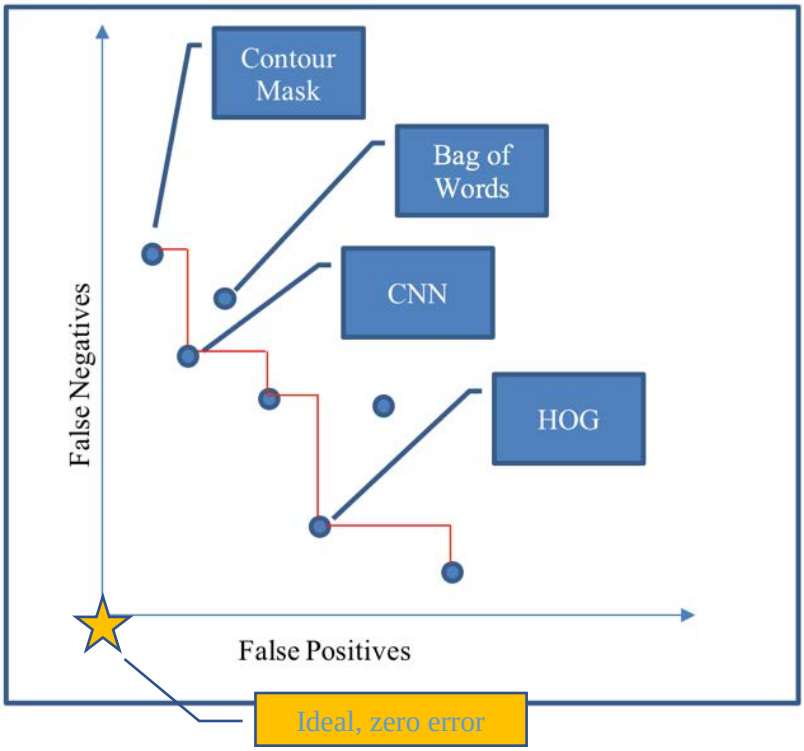

Figure 3. Example (non-real) illustration of multi-objective error plot.

The proper strengths and weaknesses for any model are primarily scoped by the error rates, and by the type and quality of training data. A model trained on satellite data will perform poorly on low-altitude unmanned-aerial-vehicle data. Also, two models with similar cumulative error rates may indeed vary inversely in false positive and false negative rates. In the ISR domain, false positives may not be a large issue with a human analyst still supervising the model, but a false negative will let a target go unnoticed which can be costly. If the models share a common validation data set, we can plot the algorithms in this 'error-space' as shown in Figure 3. (This plot is not actual data, but merely an illustration to analyze algorithms.) Now we can more objectively and characteristically compare models.

In a minimization problem like this, our goal is to push the solutions to zero error (0,0); with multiple objectives, we expect to develop a 'front' of solutions that do not 'dominate' each other in all objectives; our front of best solutions is drawn in red. While, from the plot (of fake data), we can't say that a CNN model performs strictly better than a HOG, we can say that both models perform better than the Bag of Words in all objectives. This is an important characteristic of our multi-objective framework EMADE, and a strategy to encourage a diverse pool of solutions.

THE EMADE FRAMEWORK

We developed a framework called EMADE (Evolutionary Multi-objective Algorithm Design Engine) that serves to automatically explore and push the trade-off space shown in Figure 3 towards the theoretical ideal solution that exists in the bottom left-hand corner of this graph, i.e., a solution with no errors.

To accomplish this exploration, development of algorithms with EMADE leverages the best human-derived machine learning, image processing, signal processing, and track processing building blocks (also known as primitives) from the targeted as well as other related application domains. Given these building blocks, tree structures of these building blocks can be used to describe an algorithm. We then seed the evolutionary process with the best human-derived algorithms from the targeted domain. Seeding is accomplished by building and testing the tree structures for human-derived algorithms from that domain. EMADE then evolves new algorithms through bio-inspired approaches using concepts such as mating, mutation, and natural selection to maintain a population of algorithms. In this metaphor, each algorithm is a genome, and the genes are the fundamental building blocks that constitute them. The results of this evolutionary process are not a single algorithm, but a set of algorithms that provide a trade-off in the objectives measured, e.g., false positive and false negatives. Each point in this trade-off space represents the performance of one such algorithm.

EMADE successfully produced classification and regression algorithms for feature-data [15] [16] as well as classification algorithms for time-series data [17]. The remainder of this paper shows unique features of EMADE for processing imagery and discusses its potential for data fusion.

IMAGE PROCESSING IN EMADE

Imagery algorithms are computationally expensive, requiring both significant time and memory to evaluate. Because EMADE requires the evaluating thousands of potential algorithms to evolve algorithms, we implemented a caching mechanism to save processing time, and a batched processing methodology for machine learners that support it to reduce memory load.

To cache results, EMADE uses a hash string representation (parse tree) of each created algorithm with an SHA-256 hash. EMADE stores the objective score vector of the individual with its associated hash. When a new individual is created through mating, mutation, or generation, its string representation is hashed to determine if it already exists in the hash table. If it does, the values are pulled from the table, and it is not sent for evaluation. This trades off the relatively inexpensive cost of hashing each new genome for the time intensive cost of reevaluating an already computed individual.

EMADE also hashes results of machine learning training (in effect, subtrees). Before training a machine learning algorithm, EMADE hashes the input feature data along with the learner type and parameters. If this hash exists then EMADE reads the trained machine learner from disk, else, it writes the trained learner to disk when the training completes. This hashing method is significantly cheaper than retraining a learner on the same set of features. It also helps as subtrees are exchanged in mating, meaning multiple individuals reuse the same trained and evaluated branches. At the end of each generation, the cache is pruned to maximize an expected benefit:

$$B = \sum_i b_i = \sum_i r_i \, t_i,$$

where b_i is the benefit that hashing learner i provides, r_i represents the probability that the trained model will be used again on the same data, with the same model parameters, and t_i is the time it took to train model i. The cache uses a greedy method to select trained models with the largest benefit until it reaches a cache size limit.

Another speedup leveraged in EMADE is a tiered dataset capability [15]. Tiered datasets allow EMADE to quickly test new algorithms on small datasets before spending further computational resources on an evaluation that may prove fatal. This capability is essential for processing large numbers of image processing algorithms. Testing an algorithm on a small number of images can quickly reveal errors that will emerge on a large dataset, such as dimensionality agreement or failure to construct useful features.

ADDING COMPUTER VISION PRIMITIVES TO EMADE

Table 1. Wrapped OpenCV Functions as Primitives in EMADE.

Mininmum to zero	To unit8	To uint8 scaled	To float
To float normalize	Edge detection Canny	Corner detection Harris	Corner detection min eigenval
Highpass Fourier ellipsoid	Lowpass Fourier shift	Highpass Fourier shift	Highpass Fourier Gaussian
Highpass Fourier uniform	Highpass unsharp mask	Highpass Laplacian	Highpass Sobel derivative
Lowpass filter median	Median blur	Lowpass filter average	Blur
Lowpass filter Gaussian	Lowpass filter bilateral	Lowpass Fourier ellipsoid	Lowpass Fourier Gaussian
Lowpass Fourier uniform	Threshold binary	Threshold to zero	Morph erosion rect
Morph erosion ellipse	Morph erosion cross	Morph dilate rect	Morph dilate ellipse
Morph dilate cross	Morph open rect	Morph open ellipse	Morph open cross
Morph close rect	Morph close ellipse	Morph close cross	Morph gradient rect
Morph gradient ellipse	Morph gradient cross	Morph tophat rect	Morph tophat ellipse
Morph tophat cross	Morph blackhat rect	Morph blackhat ellipse	Morph blackhat cross
Contours all	Contours min area	Contours max area	Contours convex concave
Contours min length	Contours max length	Contour mask	Contour mask min area
Contour mask max area	Contour mask convex	Contour mask min length	Contour mask max length
Contour mask range length	Contour mask min enclosing circle	Contour mask max enclosing circle	Contour mask min extent enclosing circle
Contour mask max extent enclosing circle	Contour mask range extent enclosing circle	Contour mask min aspect ratio	Contour mask max aspect ratio
Contour mask range aspect ratio	Contour mask min extent	Contour mask max extent	Contour mask range extent
Contour mask min solidity	Contour mask max solidity	Contour mask range solidity	Contour mask min equ diameter
Contour mask max equ diameter	Contour mask range equ diameter	Threshold n largest	Threshold n largest binary

Table 1 shows image processing techniques implemented from OpenCV that are wrapped in EMADE; however, most of these are not state-of-the-art features for object detection or recognition within imagery. After a brief literature review, we implemented the following techniques as primitives in EMADE:

- Histogram of oriented gradients (HOG)
- DAISY features [18]
- Grey level co-occurrence matrices (GLCM) [19]
- Batched stochastic gradient descent (SGD)
- Batched passive aggressive classifier
- Deep neural network classifier (DNN)

Capabilities like HOG, DAISY, and GLCM, are important for EMADE because they allow traditional machine learning methods (i.e. non-DNN) to operate effectively on image data. They can operate effectively because these techniques are able to find useful features within images that are robust to scaling, translations, and rotations.

PROCESSING THE XVIEW DATASET

This paper represents the first foray into creating machine learning algorithms for imagery with EMADE. As such, we bounded the task to the creation of a binary classifier of image chips. From the xView dataset, we created 224x224 pixel image chips. We chose this size to support the implementation of pre-trained deep neural networks (that are configured to these dimensions) as primitives. For each chip, we labeled the chips that contained buildings as positive test cases and those that did not contain buildings as negative test cases.

DEEP NEURAL NETWORK RESULTS

The current leading algorithms for image classification from large datasets are deep neural networks. To compare performance with DNNs, we trained a Resnet v2 101 model [20] on an equal amount of building and background images. Since this a traditional classification task, we did not have to make major modifications except for balancing the data, which is important for effective learning with a DNN. This Resnet architecture achieved an accuracy of 91.6%, a true positive rate of 90.5%, and a precision of 92.6%. Table 2 shows the confusion matrix for the trained Resnet architecture.

Table 2. Performance of Resnet v2 101 Model.

	Not Building	**Building**
Not Building	8455	648
Building	869	8228

EMADE RESULTS

We ran EMADE on a subset of the xView dataset to produce a binary classifier. We constructed a two-tiered dataset structure to allow the individuals to fail fast. The first tier contained 300 training images and 100 testing images. The second tier contained 3,200 training images and 800 testing images.

An evaluation on the first-tier dataset took EMADE on average 72.53 seconds, while the second-tier averaged 522.31 seconds. Out of 5,270 candidate algorithms, EMADE found 3053 that did not perform well enough to promote to the next tier. This means the tiered dataset structure saved 381 CPU-hours of processing time at the expense of 44.66 CPU-hours of redundant computation on successful individuals. Over the course of the optimization, EMADE ran for a total of approximately 430 CPU-hours. Without a tiered structure, EMADE would have taken 765 CPU-hours. Therefore, the tiered dataset structure offered a savings of about 44 percent.

For the xView problem, we also compiled statistics on the success of the caching implementation. Reuse of existing cached results from primitives saved 68 CPU-hours of processing time. This savings came at a cost of initially storing the cached results, which took 10 CPU-hours to process the 4,000 images. Our caching implementation netted 58 hours, which represents about a 12% improvement overall in EMADE throughput.

Figure 4 shows the non-dominated frontier produced by EMADE after evaluating over five thousand candidate algorithms. Note the Resnet DNN architecture described in the previous section received a score of 0.0375 false positive rate and 0.0025 false negative rate. This DNN outperforms a great deal of the more traditional computer vision techniques in EMADE, and was co-dominant with five EMADE non-dominated solutions. At the time we ran this experiment, we were unable to get the DNN architecture running on the cluster EMADE used, and so we could not implement it as a primitive in EMADE. In Figure 4, the green points were hand-crafted algorithms based on computer vision techniques used to seed the optimization. Note that almost all of these techniques are dominated (outperformed in both dimensions) by EMADE solutions.

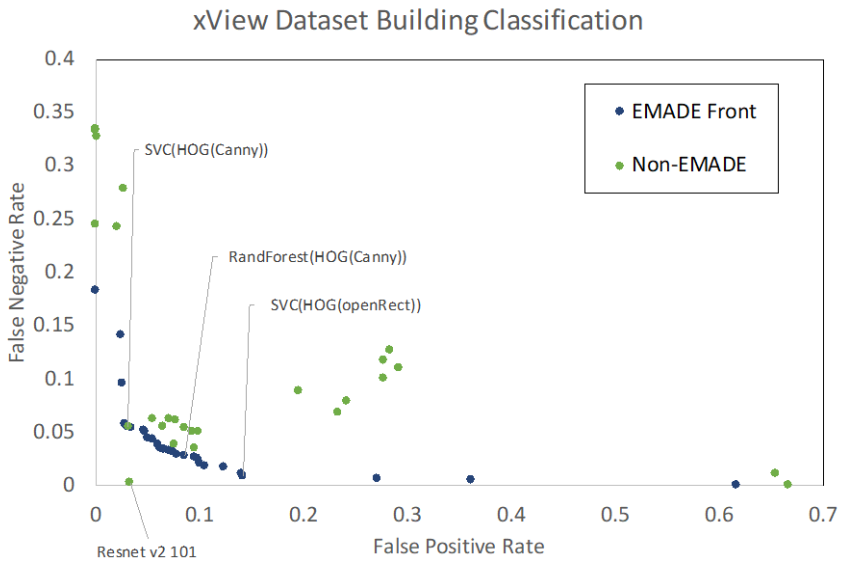

Figure 4. Non-dominated frontier of xView binary classification for buildings in EMADE.

Figure 5 shows one of the EMADE evolved solutions that was co-dominant with the DNN. The evolved solution outperformed the DNN on false positive rate, while underperforming on true positive rate.

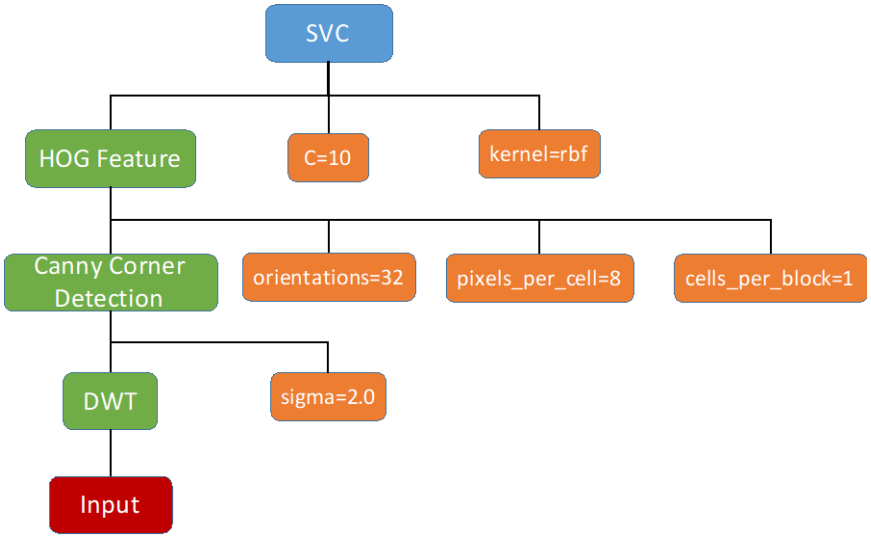

Figure 5. An evolved solution produced by EMADE with false positive rate of 0.0285 and false negative rate of 0.0575.

EMADE AS A TOOL FOR FUSION

Although the results in this paper focus on a single imagery dataset, EMADE can serve as a fusion engine in the ISR domain through its data-agnostic processing. EMADE maintains an instance-oriented representation of data, but each instance can be a list of data sources for the observation. Because EMADE dispassionately constructs algorithms from fundamental building blocks from the literature, we can also implement vital steps of the fusion process into the search space for possible algorithm pipelines. Opening the fusion process to EMADE means it can optimize the interpolation and registration steps alongside steps like feature engineering and classification.

CONCLUSIONS

EMADE is a powerful autoML tool, agnostic to both domain and data, capable of producing a non-dominated frontier of solutions across competing performance measures. This work demonstrates EMADE's capabilities on the important domain of image processing for ISR applications. We showed the ability to combine and optimize high-level, state-of-the-art techniques into new algorithms. We improved EMADE by creating new primitives, developing seed algorithms, and making changes to the infrastructure to support image data and increase throughput by caching data on disk.

Future work could look at how to better implement deep neural networks into EMADE as primitives that can be trained during the evolutionary process. One could also identify how to optimize not only the hyperparameters but the architectures (i.e., layers and configurations) of the deep neural networks as well.

REFERENCES

[1] R. S. Olson, N. Bartley, J. H. Moore and R. J. Urbanowicz, "Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, New York, NY, USA, 2016.

[2] T. Křen, M. Pilát and R. Neruda, "Multi-objective evolution of machine learning workflows," in *IEEE Symposium Series on Computational Intelligence (SSCI)*, Honolulu, HI, USA, 2017.

[3] B. Chen, H. Wu, W. Mo, I. Chattopadhyay and H. Lipson, "Autostacker: A Compositional Evolutionary Learning System," *arXiv preprint arXiv:1803.00684*, 2018.

[4] O. Javed, K. Shafique and M. Shah, "A Heierarchical Approach to Robust Background Subtraction Using Color and Gradient Information," in *Motion and Video Computing*, 2002.

[5] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *Computer Vision and Pattern Recognition*, 2005.

[6] Z. Zivkovic and F. Van Der Heijden, "Efficient Adaptive Density Estimation per Image Pixel for the Task of Background Subtraction," *Pattern Recognition Letters*, vol. 27, no. 7, pp. 773-780, 2006.

[7] A. Elgammal, D. Harwood and L. Davis, "Non-parametric Model for Background Subtraction," in *European Conference on Computer Vision*, 2000.

[8] A. B. Godbehere, A. Matsukawa and K. Goldberg, "Visual Tracking of Human Visitors Under Variable-Lighting Conditions for a Responsive Audio Art Installation," in *American Control Conference*, 2012.

[9] A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, Lake Tahoe, Nevada, 2012.

[10] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-time Object Detection with Region Proposed Networks," in *Advances in Neural Information Processing Systems*, 2015.

[11] S. Ali and M. Shah, "COCOA: Tracking in Aerial Imagery," in *Airborne Intelligence, Surveillance, Reconnaissance (ISR) Systems and Applications*, Orlando, Florida, United States, 2006.

[12] N. Audebert, B. Le Saux and S. Lefèvre , "Segment-before-Detect: Vehicle Detection and Classification through Semantic Segmentation of Aerial Images," *Remote Sensing*, vol. 9, no. 4, p. 368, 2017.

[13] L. W. Sommer, T. Schuchert and J. Beyerer, "Deep Learning Based Multi-category Object Detection in Aerial Images," in *Automatic Target Recognition XXVII*, Anaheim, California, United States, 2017.

[14] T. Tang, S. Zhou, Z. Deng, H. Zou and L. Lei, "Vehicle Detection in Aerial Images Based on Region Convolutional Neural Networks and Hard Negative Example Mining," *Sensors*, vol. 17, no. 2, p. 336, 2017.

[15] J. Zutty, D. Long and G. Rohling, "Increasing the Throughput of Expensive Evaluation Through a Vector Based Genetic Programming Framework," in *Genetic and Evolutionary Computation Conference*, Denver, Colorado, USA, 2016.

[16] J. Zutty, D. Long, H. Adams, G. Bennett and C. Baxter, "Multiple objective vector-based genetic programming using human-derived primitives," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, Madrid, Spain, 2015.

[17] J. D. Griffies, J. Zutty, M. Sarzen and S. Soorholtz, "Wearable sensor shown to specifically quantify pruritic behaviors in dogs," *BMC veterinary research*, vol. 14, no. 1, p. 124, 2018.

[18] E. Tola, V. Lepetit and F. Pascal, "A fast local descriptor for dense matching," in *IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, AK, USA, 2008.

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

[20] K. He, X. Zhang, S. Ren and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*, 2016.