Artificial Intelligence for Real Time Threat Detection and Monitoring

Steven W. Tolbert
Department of Computer and Electrical Engineering and Computer Science
Florida Atlantic University
Boca Raton, United States
stevenwtolbert@gmail.com

***Abstract*—This project attempts to design a novel method of determining the threat level of a given operator in a conflict based on audio and visual data. Work in the field of threat monitoring and detection has shown the ability to detect weaponry within an image; however, not much has been done in discerning the intent of the wielder and their relative threat to an observer. This project aims to solve this problem by developing a theoretical framework for dynamically calculating a threat vector and observing it's trajectory as it moves through n-dimensional space.**

I. Introduction

The current state of machine learning for threat detection and monitoring is highly commercialized with the majority of research being done behind closed doors and focused on the objective of raw object detection. The question of if we can detect a weapon in a given frame is not the objective of this analysis, for this task one can look towards [1],[2],[3],[4]. It will be taken as a given that weapon detection is a solved procedure limited only by the scale of data; although, advancements can always be made we currently live in a time where this task has become fairly trivial for well behaved use-cases. The general question of understanding intent is a much harder one. Current research, such as the work done by private consulting firms [5] work towards this general goal with respect to knives by not only detecting the presence of a knife but how it is wielded by an operator. Beyond looking at the problem from physically apparent features, threat detection and intent has deep roots in neurocognitive research that can be leveraged to create applicable intelligent systems for real-time threat detection. Work done by [6] for example details how important eyes are in the role of recognizing fear in an individual. It is clear that threat detection and intent is a rich problem with several angles to be explored bridging the state of the art in artificial intelligence with the cutting edge research being done in neurocognition.

A. A General Approach to Threat Monitoring and Detection

With that in mind, the practical building of this system will now be considered. The threat detection and monitoring system described in the next sections rely on an ensemble of models being created. As stated previously, this project is not designed to create the best weapon detector and as such will simply leverage the state of the art in real-time object detection as of December 2020 which will be YOLOv5 fine tuned for our use cases. In the interest of conserving scope we are also limiting our threat detection to threats where a handgun can be seen, of course threats can come in wildly different forms, and expanding this model for other threats will be of interest for future research. After the model detects a handgun in a frame, the model will create a new feature vector based on the features seen in the frame. Questions such as if a weapon is commonly found in this geography, what is the angle of the weapon towards the observer, and are there visible signs of distress in the wielder will become critical in determining the intent of the wielder. The system designed will also have access to the audio recorded during the video which allows for the analysis and detection of hostile speech in a given confrontation. The inclusion of audio presents interesting challenges with tying a dynamic time dependent feature with with a static frame. This can be done in theory by actively transcribing audio into text and then performing sentiment analysis on complete phrases with transformers such as BERT [7] or the optimized version RoBERTa [8] which drops next sentence prediction for better performance in other tasks such as classification. One can then map the sentiment result back to the current frame in time and create a more robust and dynamic feature vector describing the frame. This will be a goal of future iterations of this project; however, in the interest of conserving scope this analysis will focus only on visual data provided from the observer. The new feature vector produced from the live frame is then plotted in N-dimensional space where N depends on the number of models running. Where a scenario exists in this space provides context into the threat. The idea in general then is that threats to an observer will exist in a different feature space than non-threats. For this model to work a "threat-space" must be created from labeled data and the difficultly lies in curating a set of models that create a well segregated threat-space.

B. System Summery

The objective of threat detection and monitoring is to understand when an observer should consider a given operator as a threat. A threat will be defined as a person or thing that intends to cause harm to an observer. To limit the scope of this project we define a singular observer from the perspective of the camera recording the data. For every confrontation there exists several data points that help determine the credibility of a perceived threat. These data points will be limited to visual data continuously recorded during the confrontation and it is the goal of this project to actively use these data points to assess threat in real-time.

II. Related Works

It is the purpose of this work to develop a system for threat detection and monitoring; however this purpose is rather niche and the literature surrounding the topic is limited. To zoom out on this goal and look at it from the view of continuous classification through videos leads us to looking at advancements that have been achieved through CNNs, RNNs, and LSTMs; however, these models focus on the end-state of the phenomena attempting to understand what label should be assigned to a given frame based on the data present. Threat modeling however requires an understanding of how scenarios evolve as well as the current state. The vast majority of work done in the field of modeling threats through AI concerns itself with the preliminary goal of detecting a weapon. Such research has yielded great results and advancements allowing for detection in non-well behaved use cases such as detecting a weapon under concealment [9] or under low-resolution conditions [10]. However little has been done in understanding intent and quantification of threat given a conflict scenario. The work referenced in [5] comes closest to this goal by having the models learn the difference between a wielded knife and one sitting on a table for example. While works concerning itself with threat detection is sparse, there is no shortage of literature regarding the general goal of object detection which is used as the primary step on our threat detection pipeline. For this task we employ the use of the "You Only Look Once" (YOLO) framework detailed in [1], specifically working with the latest iteration YOLOv5. The YOLO architecture is an extremely clever way of doing multiple object detection in a frame by predicting the bounding boxes of objects belonging to defined classes. This is done by first dividing the image into a NxN grid, the model then learns if a grid cell contains an objects and attempts to fit the anchor boxes to the detected object, then non-max suppression is applied to determine the likely object that should be classified within the bounding box in the case when an object is detected multiple times. As an aside, the version of YOLO we will be using, YOLOv5, has some controversy surrounding its development as it's technically not a fork of the original YOLO iterations; however, performance testing has shown that YOLOv5 performs at the fastest speeds for real-time object detection and as such will be used for this project.

III. Main Body

As stated, the goal of this project is to understand threat detection and monitoring. The ability to determine if an operative should be considered a threat in real-time would provided limitless opportunities for building and designing safer systems in practical applications. For this goal a new space must be created in which features of non-threatening spaces are clustered together while features of threatening spaces are clustered together in a different area of N-dimensional space.

A. Threat Spaces

This task becomes challenging as your space is populated from features derived from the models, therefore it becomes a large and iterative process of trying different models and model combinations to populate the space in a well separable way that creates well defined clusters of threatening scenarios and non-threatening scenarios.

B. Vector Mapping

For this task we will consider an ensemble of models working in parallel, this allows for the work of large neural networks doing separate tasks to be spread across multiple devices. For every frame in the video we can create a feature vector $\vec{x}_{i,t}$ where i represents the video being mapped and t represents the frame of that video.

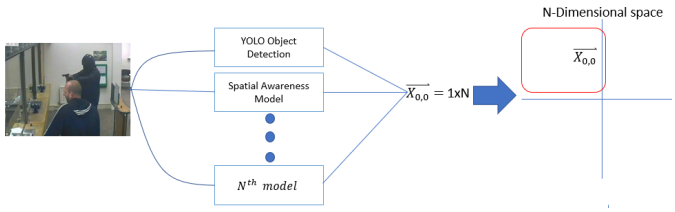

Fig. 1. How a Single Frame is Mapped to high Threat-Space

The goal then is to design a set of models that create a well divisible feature space in which high threat scenarios exist in one region and low-threat scenarios exist in another region.

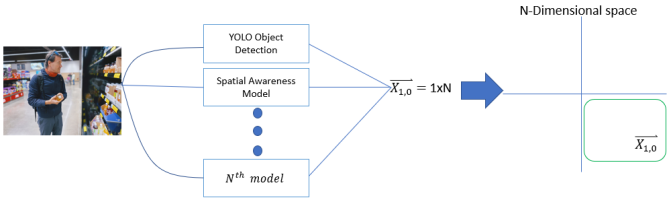

Fig. 2. How a Single Frame is Mapped to low Threat-Space

Possible models for creating this new threat-space include models such as a YOLO object detector which can detect the presence of weaponry in an image, or a model that is able to understand the context of where the conflict is taking place, a gun in the hands of a hunter for example shouldn't create as much threat as the gun in the hands of a criminal. By having N different models on each frame we can then create a general idea of context around images rather than making claims about generalities.

C. Frames as Kinematic Objects

As this analysis using videos and not singular frames, we can go beyond the base classification task and describe how threat evolves in time. As every frame creates a time-dependent dynamic feature vector this analysis will represent this in general as:

$$\vec{x}_{i,t} = a(i,t)\hat{x^0} + b(i,t)\hat{x^1} + \cdots + c(i,t)\hat{x^N} \tag{1}$$

Each directional unit vector corresponds to a different model in the ensemble of models listed from model 0 to model N. The coefficients of each directional unit vector represent the output of that model for video i, at time t. It is possible to create a dynamic representation of the video to not only can we map the frame to feature space, but also see how it evolves in time. It would make sense dynamic scenarios to follow dynamic classifications, as threat is not a immutable variable we should expect threat to rise and fall with respect to the physical situation occurring in the conflict. As we essentially have n-dimensional positions now, we could also consider n-dimensional velocities and accelerations which poses an interesting way of describing escalations and deescalation in conflict. Consider the classic example of a bank robbery where the bank robber will play the role of the operative. For this scenario we would like to determine their threat. When initially calculating threat of the operative, he/she would place the operative into a feature space surrounded by low-threat scenarios as they are likely to at first be concealing their intent. As soon as the operative is revealed to be a robber and begins audio and visual traits of high-threat scenarios their feature vector will move in n-dimensional space. For every frame calculated we will have a new n-dimensional position vector. These position vectors can then be interpolated to a well behaved function $f(\vec{x}_{i,t})$ which provides a general description of how threat evolves.

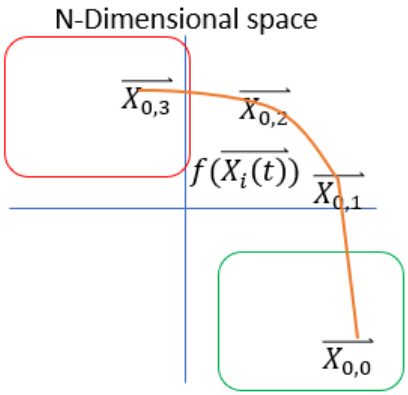

Fig. 3. How multiple frames can traverse threat-space

If the polynomial interpolation is well-behaved we can then take first and second derivatives of our interpolated function $f(\vec{x}_{i,t})$ to produce a quantity of "threat-velocity" and "threat-acceleration".

$$Threat_{Position} = f(\vec{x}_{i,t}) \tag{2}$$

$$Threat_{Velocity} = f'(\vec{x}_{i,t}) \tag{3}$$

$$Threat_{Acceleration} = f''(\vec{x}_{i,t}) \tag{4}$$

The units for these values will be threat, threat per unit time and threat per unit time squared respectively. For the case of the operative robbing the bank while initially concealed, the change in feature space would be large leading to large threat-acceleration values. Considering threat in this manner allows for interesting calculations to be yielded. Threat accelerations and velocities for example can they be used to dynamically calculate threat for multi-operative scenarios to determine priority of target.

D. Parallel Deployment

As the models are computed in parallel, the feature vector is able to be computed at each time step at the speed of the slowest model. For real time prediction, models should be tuned to produce results at the frame rate the camera can provide, although such speeds are likely unobtainable therefore what this analysis considers "real-time" is real-time in practicality with only slight latency between frame and prediction. Deployment than can be done through many online endpoints which are passed the frame of the video and returns the result from the model the endpoint was designed for. The main program will exist locally waiting for a response from all endpoints and accumulate the results into the feature vector representing that frame and stored into an array. The next frame repeats the same process producing a new feature vector representing the second frame. Once you have at least two frames you can begin interpolating your points in N-Dimensional space to create your trajectories.

E. Threat Trajectories

To help illustrate this idea we will construct the 1-D case by first consider a three frame scenario where each frame corresponds to t_0, t_1, and t_2. For each frame we will calculate a threat based on the 1 model we are running (as this is the 1-D case) For further simplification we will restrict the output of this model to a binary class either 0 or 1. Passing each frame into the model would then produce a set values (1 value per frame). Let us define these as follows: $x(t_0) = 0, x(t_1) = 1, x(t_0) = 0$ This kind of behavior would correspond to an event occurring and then returning back to the previous state within 3 frames. Plotting these points, we can see that we are able to fit a parabola to this set of discrete points $x = 2t - t^2$.

From our interpolated function we can create the threat velocity and acceleration

$$Threat_{Position} = 2t - t^2 \tag{5}$$

$$Threat_{Velocity} = 2 - 2t \tag{6}$$

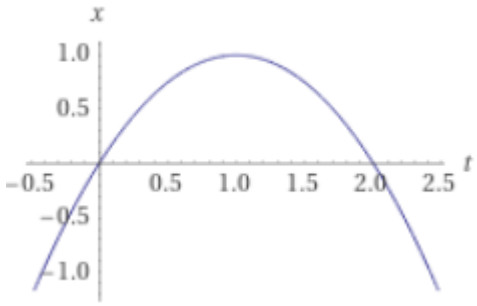

Fig. 4. $x = 2t - t^2$

$$Threat_{Acceleration} = -2 \tag{7}$$

One can also consider the linear behavior between singular frames to see how threat increases in the first region from 0 to 1 and then decreases in the second region from 0 to 2. The corresponding threat-velocity in region 1 is positive while the corresponding threat-velocity in region 2 is negative. As our interest is in real time analytic, we do not have the luxury of fitting the entire curve to an analytic function. Therefore let us consider how we can model the linear behavior between frames to yield approximate results.

There are several methods of interpolating the curve numerically including simple nearest-neighbor approaches or more complex radial bias function interpolations. For this analysis we will use a linear interpolation method to estimate the arc length of the curve between frames. The kinematics are then defined discretely using the estimated distance between frames in feature space.

Recall how we defined a frame mathematically as a vector, it is then possible to calculate an euclidean distance between vectors as:

$$\Delta x = \sqrt{[a_{t+1} - a_t]^2 \hat{x^0} + \cdots + [z_{t+1} - z_t]^2 \hat{x^N}} \tag{8}$$

$$\vec{v} = \frac{\Delta x}{\Delta t} \tag{9}$$

$$\vec{a} = \frac{\Delta v}{\Delta t} \tag{10}$$

Therefore for every frame we will be keeping track of its position, its velocity, and its acceleration relative to the previous frame. As we are doing frame-by-frame calculations for the threat kinematics, the above dependence on Δt will always be 1 as the time between frames is uniform under ideal conditions. Practically if there are missing frames then a more robust method of calculating Δt is required.

F. Position Relative to Threat Cluster

The final requirement for this model to function is to understand position relative to the clusters. The vectors produced in the previous step are almost meaningless if there is no reference of what those objects are moving towards in N-Dimensional space. Therefore we must make the assumption that we have already created the clusters which have well defined centers in N-Dimensional space. The angle our kinematic vectors make with the vector pointing to the centers of each threat-region tell us in what direction threat is evolving.

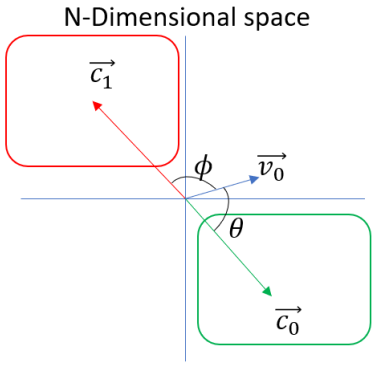

Fig. 5. Angle between a velocity vector and the two clusters.

It is then the case when the $\cos(\theta)$ or $\cos(\phi)$ is equal to zero the kinematic vector is pointing in the direction of a cluster.

$$\cos(\theta) = \frac{\vec{v_0} \cdot \vec{c_0}}{|\vec{v_0}||\vec{c_0}|} \tag{11}$$

$$\cos(\phi) = \frac{\vec{v_0} \cdot \vec{c_1}}{|\vec{v_0}||\vec{c_1}|} \tag{12}$$

G. Manifolds

The work described provides a general framework for describing threat evolution in time; however, there is no memory in the system. An operative can move from a low-threat space into a high-threat space and then back into a low-threat space. This circular scenario does properly describe threat evolution, once a target becomes a threat in a conflict it shouldn't be able to easily loop back to a low-threat space.

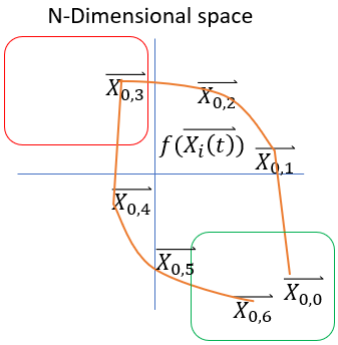

Fig. 6. Threat loops.

A potential solution to this problem is the introductions of dynamic manifolds in N-Dimensional spaces that effectively change the distance between frames changing the kinematics affecting threat velocity and threat acceleration. With the introduction of manifolds, further work using Riemannian

geometry would need to be done in order to properly calculate distances between frames in this new N-Dimensional space. Once the system becomes critical by entering the high-threat space it should be harder to escape the high-threat space and return to a low-threat space. The inverse should also hold true, if you exist in a low-threat space, slight grievances shouldn't rapidly accelerate your threat.

IV. Experiments

In order to build the system proposed in the previous sections we must first construct a set of models and then create a local system to orchestrate the creation of the feature vectors as well as the tracking of those vectors trajectories. The full design involving many models of both audio and video type data falls outside of the scope of this project due to the amount of resources and time needed to create the full pipeline. It is then the case we will concern ourselves with the 1-D version of this problem which reduces the complexities of trajectories in N-Dimensional to a simple line with a single variable driving the trajectory. The most obvious choice in threat detection and monitoring is then the presence of a weapon in the frame of which we will restrict ourselves to the detection of handguns in the frame. After we have built the handgun detector we can then consider how threat will change in this 1-D space, how we can interpolate the data in order to produce threat velocities and accelerations as well as generalizing this model for N-Dimnesional space.

A. Benchmarking Methodology

The obvious question next becomes how can we benchmark this analysis. Considering that this project has only laid down the theoretical foundations at this stage, most benchmarking methods will not properly capture the full ability of the models framework. Video classification essentially is image classification over a set of images, therefore we can consider the frameworks performance in terms of it's ability to classify threats in comparison to a simple binary image classifier. However, a simple binary image classifier fails to capture the time evolution component of videos. The introduction of more complex networks involving CNNs, RNNs, and LSTMs attempts to solve this problem allowing for continuous classification keeping track of what frames it's already processed, but it doesn't quite describe the evolution of those frames, only the end-state. Therefore, our ability to benchmark is somewhat limited not only to the current models available but also due to the fact we have restricted ourselves to the 1-D case for this analysis in the interest of time and resources. As we are restricted to the 1-D case the threat-space is simplified to a binary space where the presence of a weapon would send us to a high-threat region and therefore the operative is a threat and an operative with no weapon would be sent to a low-threat region where they would not be considered a threat. Under such restrictions comparing the results of the 1-D framework to a binary classifier would be comparable.

B. The Binary Threat Detector for Benchmarking

The most basic binary threat classifier would be a multi-layer perceptron neural network trained on a set of images divided into threats and non-threats. For this task the analysis pulled 1000 images from the Open Images Dataset [12] resized to 256x256 and converted to grayscale to be used as the non-threat images. The model then used 1000 images from the data set used to train the YOLO model which contain guns being used in a threatening manner resized to 256x256 and converted to grayscale to form the threat data. The binary classifier threat model was trained and tested on the data described with a 60/40 split. The architecture of the multi-layer perceptron follows: 2 hidden layers with 50 hidden units in the first layer and 10 hidden units in the second layer with a logistic activation function. The performance of the model was then tested through 5 fold cross validation and an aggregated accuracy score is produced.

C. Benchmarking Results

The resulting cross validated accuracy score is $63\% \pm .13$ what this tells us is that there are no well defined global features that accurately represent threat in a given image and the MLP is only able to perform slightly better than random chance. Therefore our approach of using an ensemble of models directly searching for various features that attribute to threat should be able to provide much better results.

D. The Handgun Detector

The first task of this project was to train the YOLOv5 architecture to be able to detect handguns in images. This project uses the largest pretrained model YOLOv5x to determine the initial weights in which to start training from. The rest of the configuration settings are mostly default: 50 epochs, 640 image size for both train and test sets, batch size of 64. Larger batch sizes help improve training time and can optimize convergence, but requires large amounts of VRAM in order to fit on the GPU for a given epoch. It is recommended to increase the number of epochs to improve model performance; however, due to the scale of data being used the number of epochs was limited due to budget and time. The model was trained on an AWS Sagemaker ml.p2.8xlarge server for several hours on a data set of 2986 images. The data was split into training, validation, and testing sets with 70/20/10 split respectively. The data was sourced from a public data set provided by the University of Grenada department of Soft Computing and Intelligent Information Systems [11] which contains the labeled 2986 handgun images including bounding box annotations. These images were resized to 256x256 for uniformity while training and testing the model. Further considerations were made in regard to this data set as the YOLOv5 architecture the data must be formatted in the YOLOv5 PyTorch format including a data.yaml file which describes the classes you will be training for prediction.

E. Handgun Detector Training and Cross-Validation

The following figures show the loss related to both the predicted bounding box and the loss related to the given cell containing an object as well as their validation scores displayed as Box and Objectiveness respectively. We can see from these plots that the model is progressively learning through every epoch.

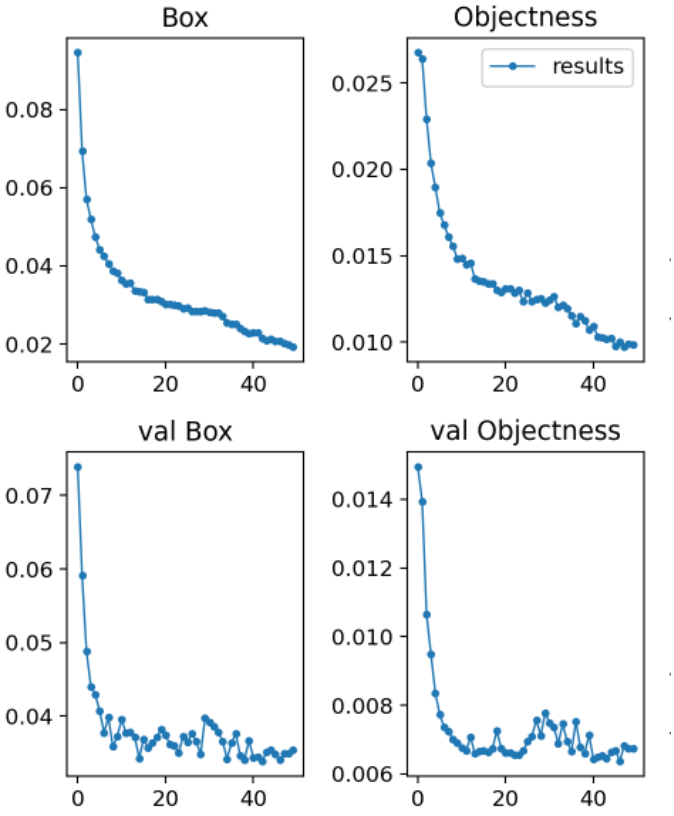

Fig. 7. Loss Function Graphs for Bounding Boxes and Objectness respectively.

Further metrics are computed in the form of precision, recall, mAP@.5 and mAP@.5:.95. These metrics are defined as follows:

$$precision = \frac{TP}{TP + FP} \tag{13}$$

$$recall = \frac{TP}{TP + FN} \tag{14}$$

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{15}$$

The resulting curves for these metrics are shown in figure 8 showing fairly high precision, recall, and mAP metrics. The models architecture is summarized as having 484 layers and 88390614 trainable parameters.

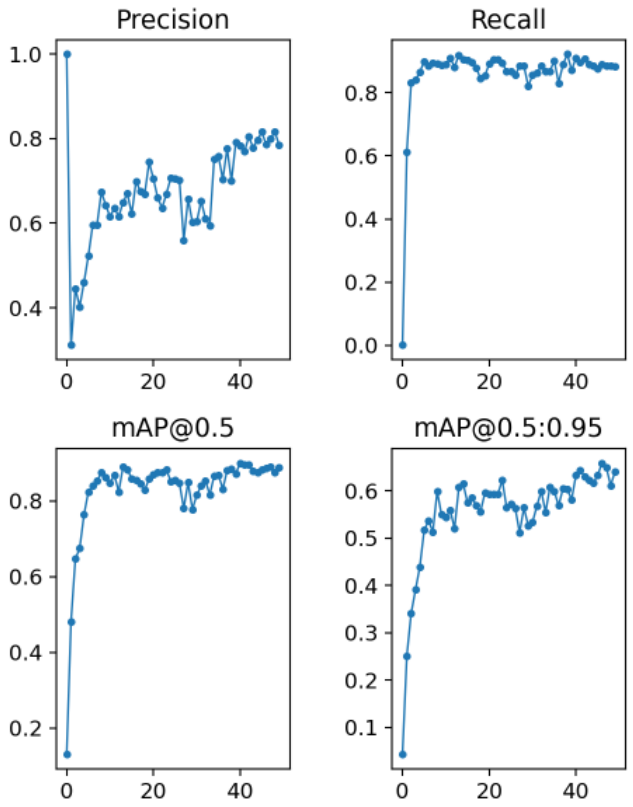

Fig. 8. Metrics for Precsion, Recall, and mAP through training.

F. Handgun Detector Test Results

The model has shown promising results in training and cross-validation, therefore we can now consider the test results via the precision recall curve on the unseen test data using the best weights trained from the model. Using the 587 images in the test set which contains 687 targets (as a single image could contain multiple handguns) the model was tested for precision, recall, and mAP metrics.

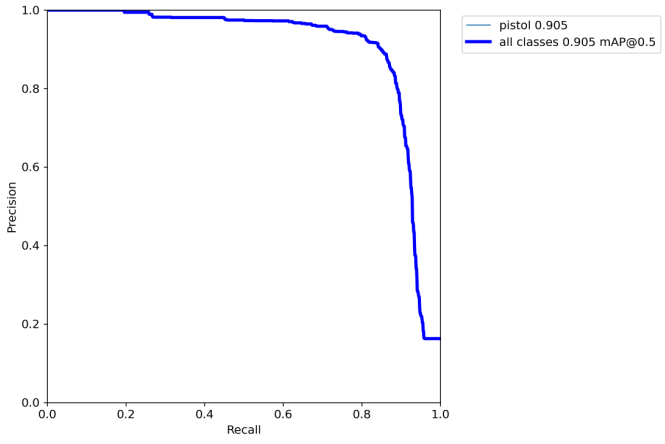

Fig. 9. Precision Recall Curve of the YOLOv5x model tuned for Handgun detection

Fig. 10. Examples of Handguns predicted in the Test Set

As we can see from the precision recall curve in figure 9 and the test images in figure 10, the model is able to successfully detect the presence of a handgun in a image even if the handgun is oriented in non-ideal ways, non traditionally shaped, or including multiple handguns in the image. These results can be summarized in the following table using the results from the test set:

TABLE I
METRICS AS PERFORMED ON TEST DATA

	Images	*Targets*	*Precision*	*Recall*	*mAP@.5*
Metrics	587	687	.803	.89	.905

G. 1-D Threat Measurement

The most trivial example of threat monitoring is in the case of 1-dimension with no manifolds where each frame only passes through one model producing a feature vector that contains only a single value. The threat-space produced is binary i.e. the presence of a weapon immediately would send the operative into a high-threat region while no weapon would send it to a low-threat region. Therefore threat measurement in 1-D would follow the same precision and recall metrics as the YOLO model metrics described in the previous section. Threat velocity and threat acceleration are also trivially obtained in the 1-D case as going from 0 to 1 would yield a threat velocity of $1\frac{t}{s}$ and a threat acceleration of $1\frac{t}{s^2}$

H. 1-D Threat Compared to General to MLP Results

The results provided show we are able produce classifications better than the slightly above random chance predictions produced from the multi-layer perceptron. These results are limited to the 1-D case where the threat-space is restricted to to a binary space, next steps would involve expanding the model to N-dimensions.

V. CONCLUSIONS

The research conducted shows how threat detection and monitoring can be achieved by tracking trajectories of feature vectors in N-dimensional space as well as provides a general framework for completing this task in real-time by having all models run in parallel and then concatenating the results into one feature vector for each frame. The full study of this methodology exceeds the scope of this project and can not fully be explored under the current constraints of this project. The research conducted can easily be expanded upon by considering how multiple targets affect threat trajectories as well expanding the model beyond the 1-D case into N-Dimensional threat-space. While the primary goal of this research was to model threat monitoring and detection, the same methodology could be applied to other time-dependent dynamic systems.

REFERENCES

[1] Joseph Redmon and Santosh Divvala and Ross Girshick and Ali Farhadi "You Only Look Once: Unified, Real-Time Object Detection" 2016, 1506.02640, arXiv

[2] J.Santaquiteria, A.Velasco-Mata, N.Vallez, G.Bueno, J.Alvarez, O.Deniz, "Handgun detection using combined human pose and weapon appearance"2020, 210.13753, arXiv

[3] Alexander Egiazarov and Vasileios Mavroeidis and Fabio Massimo Zennaro and Kamer Vishi "Firearm Detection and Segmentation Using an Ensemble of Semantic Neural Networks" 2020, 2003.00805, arXiv

[4] Zhong Zhou and Isak Czeresnia Etinger and Florian Metze and Alexander Hauptmann and Alexander Waibel "Gun Source and Muzzle Head Detection" 2020, 20001.11120, arXiv

[5] David A. Noever and Sam E. Miller Noever "Knife and Threat Detectors" 2020, 2004.03366, arXiv

[6] Elsherif, Mahmoud Medhat et al. "The perceptual saliency of fearful eyes and smiles: A signal detection study." 2017, PloS one vol. 12,3 e0173199, PloS

[7] Jacob Devlin and Ming-Wei Chang and Kenton Lee and Kristina Toutanova "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" 2019, 1810.04805, arXiv

[8] Yinhan Liu and Myle Ott and Naman Goyal and Jingfei Du and Mandar Joshi and Danqi Chen and Omer Levy and Mike Lewis and Luke Zettlemoyer and Veselin Stoyanov "RoBERTa: A Robustly Optimized BERT Pretraining Approach" 2019, 1907.11692, arXiv

[9] Blum, R., Xue, Z., Liu, Z., Forsyth, D. S. "Multisensor concealed weapon detection by using a multiresolution mosaic approach" 2004, VTC2004-Fall. 2004 (Vol. 7, pp. 4597-4601). IEEE, IEEE 60th Vehicular Technology Conference

[10] Grega, M., Matiolański, A., Guzik P. ,Leszczuk, M. "Automated detection of firearms and knives in a CCTVimage.", 2016, 16(1), 47., Sensors

[11] Olmos, R., Tabik, S., Herrera, F. "Automatic handgun detection alarm in videos using deep learning.", 2018, 275, 66-72. doi.org/10.1016/j.neucom.2017.05.012, Neurocomputing

[12] Krasin I., Duerig T., Alldrin N., Ferrari V., Abu-El-Haija S., Kuznetsova A., Rom H., Uijlings J., Popov S., Kamali S., Malloci M., Pont-Tuset J., Veit A., Belongie S., Gomes V., Gupta A., Sun C., Chechik G., Cai D., Feng Z., Narayanan D., Murphy K. OpenImages: A public dataset for large-scale multi-label and multi-class image classification, 2017. Available from https://storage.googleapis.com/openimages/web/index.html.