

# DS 745: Project One

## Kaggle Passenger Screening Algorithm Challenge:

### Improving the Department of Homeland Security's Threat Recognition Algorithms

---

Rick W. Lentz, MS, MS

25 February, 2018

Please view this document here:

<https://docs.google.com/document/d/1LPfyxLrGH9a9bu1fwd4lqt60Q2ED1ydr4PealesUVEQ/edit?usp=sharing>

—



## Definition: Project Overview

The U.S. Department of Homeland Security (DHS) put forth a community challenge on Kaggle to improve the performance of threat recognition algorithms. The origin of the challenge comes from DHS's Apex Screening Program. APEX is managed by the U.S. Transportation Security Administration (TSA), which was created after the Al-Qaeda perpetrated attacks on September 11th, 2001.

Most people who travel associate the TSA with long lines and frantically shuffling luggage into plastic bins. While this isn't a fun experience, airport security is a critical requirement for safe travel. Despite this perception, TSA's important mission includes all U.S. airport security, which means screening more than two million passengers daily.

The Apex Screening at Speed Program has a goal of reducing high false alarm rates since they create significant bottlenecks at the airport checkpoints. Whenever TSA's sensors and algorithms predict a potential threat, TSA staff needs to engage in a secondary, manual screening process that slows everything down. The program has observed that every year new threats develop. This continual change requires that prediction algorithms be refreshed and tested to meet the evolving threat and quickly process an increasing number of travelers.

## Definition: Problem Statement

In 2017, TSA screened 771,556,886 passengers through 440 federalized airports.

TSA purchases updated algorithms exclusively from the manufacturers of the scanning equipment. These algorithms are proprietary, expensive, and often released in long cycles. Even a modest decrease in false alarms will help TSA significantly improve the passenger experience while maintaining high levels of security.

Thus the problem challenge is posed to the broader data science community: Help the DHS's TSA improve the accuracy of their threat prediction algorithms.

## Definition: Datasets and Inputs

The TSA Apex Screening project team generated training and testing data designed to capture real scanning conditions. The data are captured from volunteers wearing different clothing types (from light summer clothes to heavy winter clothes), different body mass indices, different genders, different numbers of threats, and different types of threats. Due to restrictions on revealing the types of threats for which the TSA screens, the threats in the competition images

are "inert" objects with varying material properties. These materials were carefully chosen to simulate real threats.

The sensors that capture each subject and scenario combination are the latest generation of scanners. Thus the data available present a true challenge in identifying the presence of simulated threats under a variety of object types, clothing types, and body types. The training data divides the human body into 17 total body zones and indicates the probability that a threat is present or not.

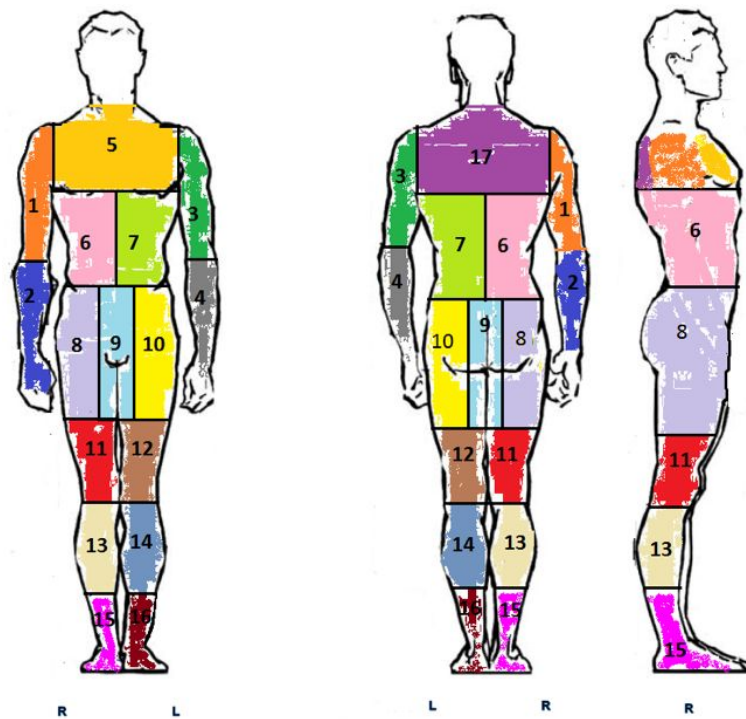
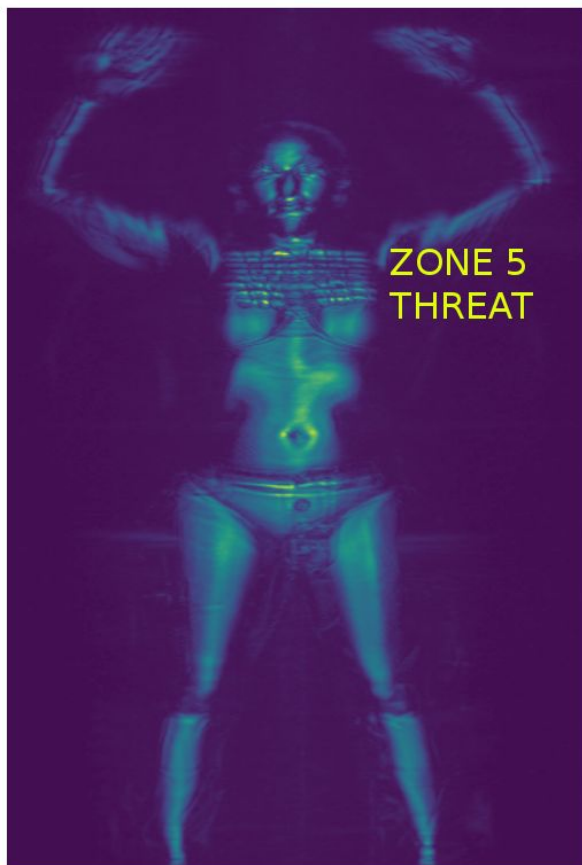


Table 1: Description of each zone is as follows:

Region	Region Name	Region Location Description
1	Right Bicep	Top of the right shoulder to just above the elbow
2	Right Forearm	Elbow to the finger tips
3	Left Bicep	Top of the left shoulder to just above the elbow
4	Left Forearm	Elbow to the finger tips
5	Upper Chest	Front of the body - stops after pectoral area on a person
6	Right Rib Cage and Abs	From the bottom of the pectoral muscle to the top of the waist line, center of the back, rib cage on the right side
7	Left Side Rib Cage and Abs	From the bottom of the pectoral muscle to the top of the waist line, center of the back, rib cage on the left side
8	Upper Right Hip / Thigh	Waist line to the half of persons thigh and extends to back include half of the right buttocks (pocket region)
9	Groin (Sensitive Area)	Below the central part of the waist, includes part of the both right and left inner and Upper Thigh
10	Upper Left Hip / Thigh	Waist line to the half of persons thigh and extends to back include half of the left buttocks (pocket region)
11	Lower Right Thigh	Half of the lower right thigh - include the knee includes
12	Lower Left Thigh	Half of the lower left thigh - including the knee includes
13	Right Calf	Below the knee to the lower leg
14	Left Calf	Below the knee to the lower leg
15	Right Ankle Bone	Starts at the end of the calf muscle and includes the foot
16	Left Ankle Bone	Starts at the end of the calf muscle and includes the foot

Specifically, the dataset contains 1247 body scans acquired by a new generation of millimeter wave scanner called the High Definition-Advanced Imaging Technology (HD-AIT) system. The competition task is to predict the probability that a given body zone (out of 17 total body zones) has a threat present. Thus, there are 19,500 total body scan-zone attributes, 1871 contain threats and 17,629 do not contain threats.



Note: All persons contained in the dataset are volunteers who have agreed to have their data used for this effort. Please do not distribute this report or reference data other than for official use.

## Solution Benchmark and Evaluation Metrics

The solution is to build and improve a model that provides higher accuracy to improve DHS's APEX screening program. The solution file will be submitted to Kaggle following the program's published technical guidance. Kaggle will evaluate the model's performance and provide feedback that will be integrated into the project's report.

The benchmark for this solution are other Kaggle submissions. K-fold cross validation will be used to leverage the training set and adjust the model parameters before submitting on the hold-out set.

The evaluation metric is based on provided metadata on the validation dataset. Performance is calculated over the accuracy of the presence of a threat in the proper body zone. The Kaggle submission will determine the overall performance score of the algorithm on holdout data.

For every scan in the dataset, the solution must submit the probability that a threat is present in each of 17 body zones

$$-\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] ,$$

where:

- $N$  is the 17 \* the number of scans in the test set
- $\hat{y}_i$  is the predicted probability of the scan having a threat in the given body zone
- $y_i$  is 1 if a threat is present, 0 otherwise
- $\log()$  is the natural (base e) logarithm

Note: the actual submitted predicted probabilities are replaced with  $\max(\min(p, 1 - 10^{-15}), 10^{-15})$ . A smaller log loss is better.

## Analysis: Data Exploration

The source data and first derivation data products consist of 106,634 files that consume 648.1 GB. The source data for each scan consists of a single 2.4 GB binary file produced by the HD-AIT system. From these files, various angular, false color images can be generated. The first image planes I generated were rotated the vertical axis and also the mean of all images, by perspective.

## Analysis: Exploratory Visualization

The following visualizations are intended to provide an overview of the options available to provide the algorithm. Again, please do not distribute this document or release any linked content (dark linked, non-indexable videos or animations).

Fig 5 and 6: Sixteen frame, vertical axis rotation (individual subject and subject mean)

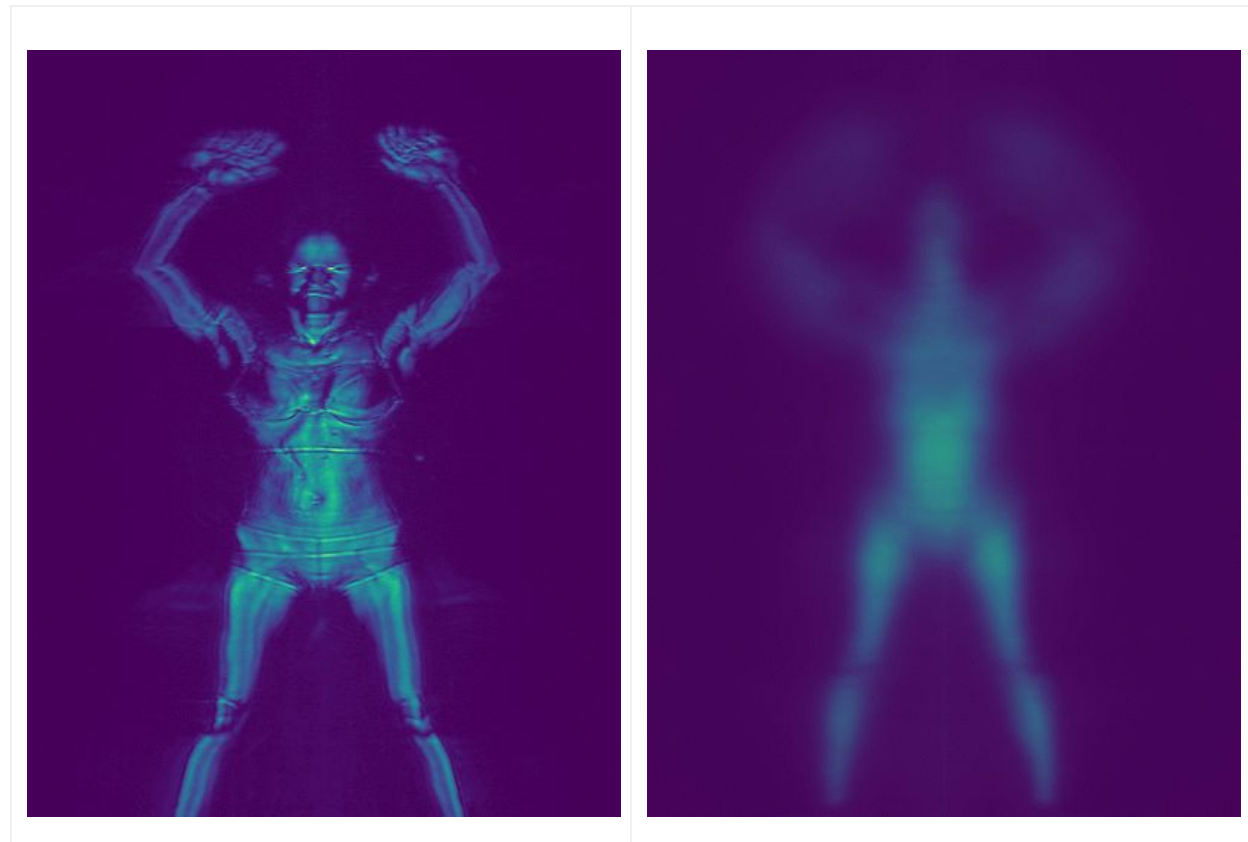
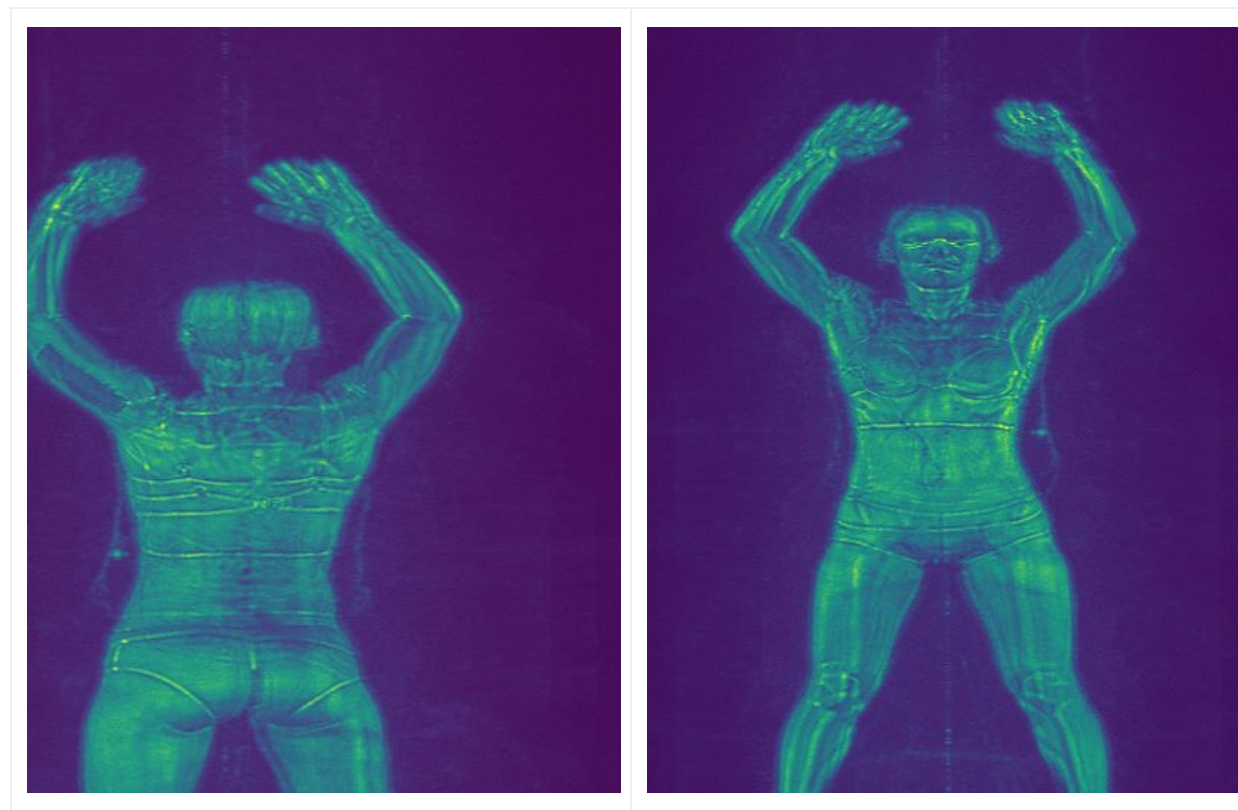




Fig 7 and 8: Single subject, 64 frame split, using Google Photo Animation Assistant. Note the vertical and horizontal drift induced by this technology was similar to that observed by auto alignment algorithms designed to solve for camera origin from multiple 2D camera frames.



Given off the shelf alignment algorithms failed or produced varying results, even with detailed images, another technique was needed. I wasn't sure if the vertical plane, horizontal plane, or combination would perform better, so I produced both. I knew I third party alignments were out, so I generated my own visualizations.



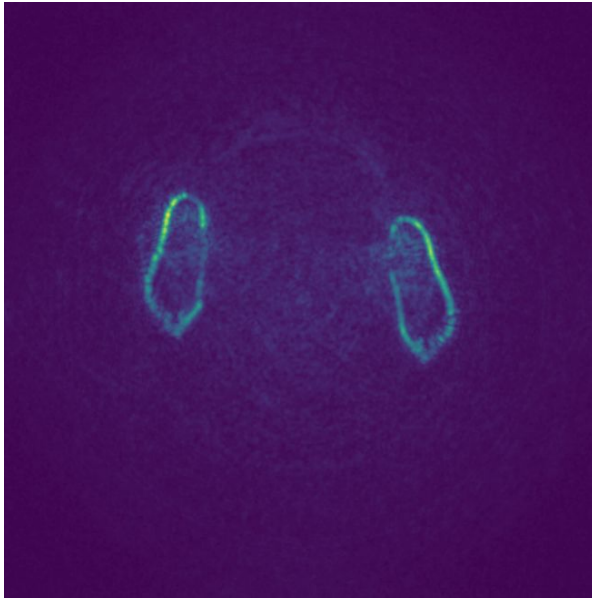


Fig 9: Horizontal video of subject, created from 660 frames (512x512 pixel images)

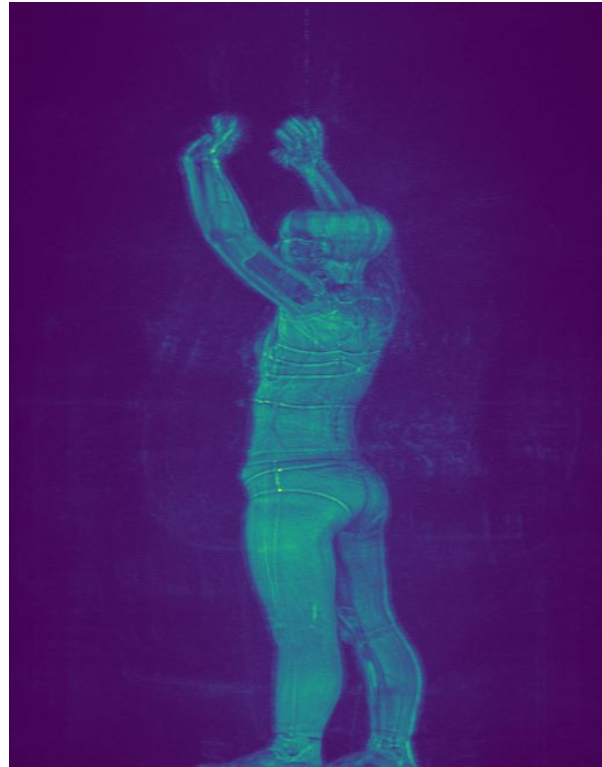
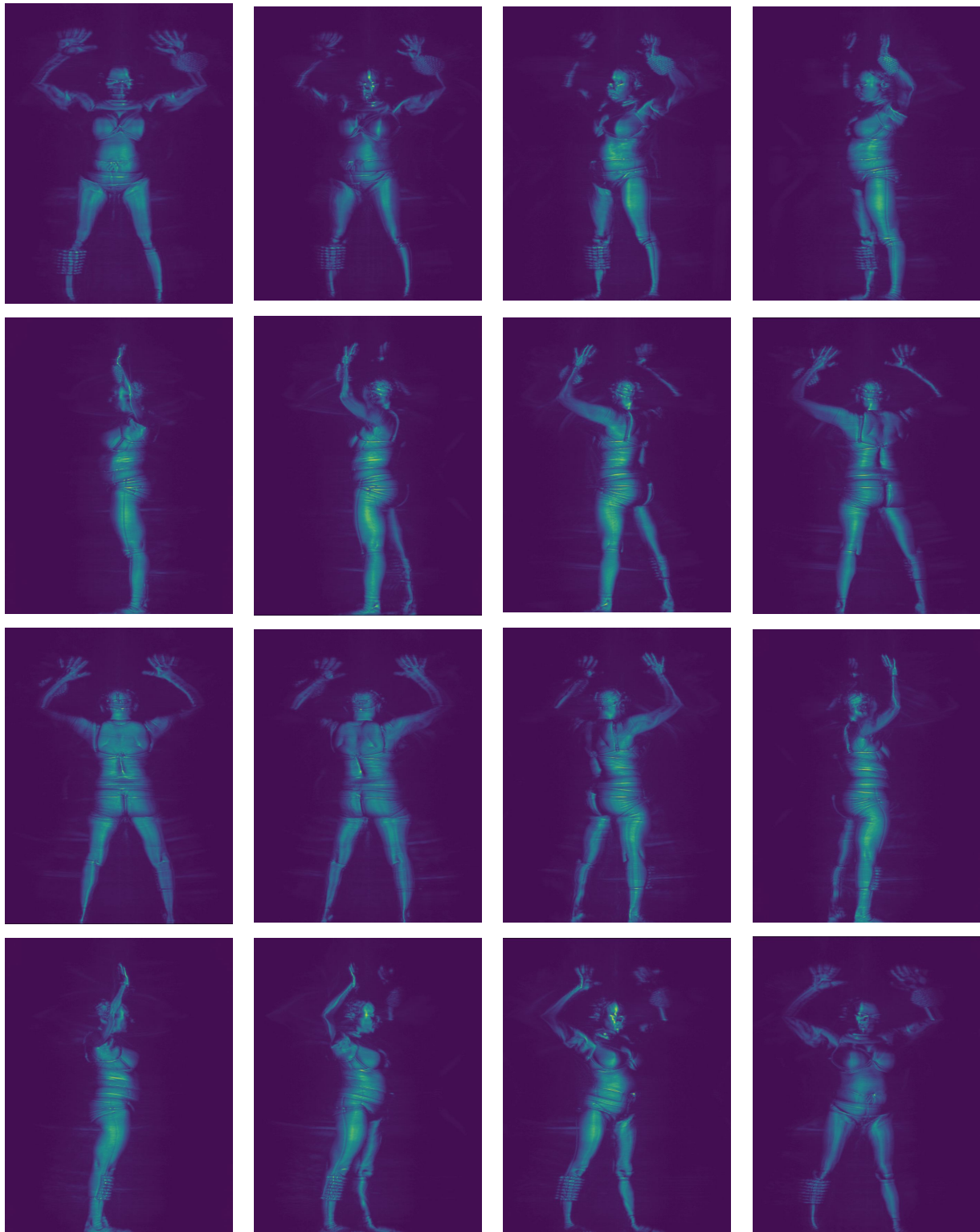


Fig 10: Vertical perspective video of the subject, created from 64 frames (512x660 pixel images)

I didn't have any experience processing the above types of data, nor did I have any experience localizing region based on the threats. Further, I felt that perhaps isolating the view, so each of the sixteen perspectives contains all of the 1247 training scans from that perspective. The idea here was complexity reduction strategy could be exploited and individually trained perspectives could be weighed based on their accuracy (and ability to observe) one or more threat zones.

Figs 11-26: Sample level, fixed perspective videos. 16 perspectives, 1247 frames (each 512x660 pixel images)



## Analysis: Algorithms and Techniques

I changed focus for four months to gain additional experience in convolution neural networks (CNNs). Even given the tight scope of this paper, it is critical to review recent progress from a macro perspective, before detailing how transfer learning and CNNs are applied in this solution.

In review, CNNs are useful for both classifying objects (what the object is) and localization (where the object is). We know that building complicated CNNs architectures and training 50 layer networks is both complex and can be resource intensive, even with GPUs. Transfer learning allows the fine-tuning of a high performing pretrained networks to mitigate these issues.

VGG 19 is composed of 19 components, each is either a convolution or fully connected layer since pooling 'layers' don't count. Using the model is always the final and most trivial part.

This problem also requires threat localization. Threat localization is performed using Single-Shot MultiBox Detector (SSD) (Wei Liu, 2015).

My prior meta model parameters showed that Xception with adadelta seemed to offer the best generalizable performance. In addition to Xception, the following three models were evaluated Inception\_v3, ResNet-50 and VGG-19...

[Removed]

## Methodology: Data Preprocessing

The workflow used in this project is straightforward. To leverage pretrained deep convolutional neural networks, one must first generate bottleneck features. That this means is we take the input images and propagate them through the pretrained network. When we use transfer learning, we don't want the pretrained layers to change, in essence we freeze them. This allows us to avoid significant computational overhead associated with adjusting the many layers of the existing network.

Each directional frame will be rendered using false color. Transfer learning of publically available models will be used to attempt to shortcut training of the deep CNN model. Several prior projects were reviewed [Nei, Somogyi, Bu, Armin] to improve CNN model performance. I selected transfer learning and realized during my earlier projects.



## Methodology: Implementation

[Removed]

## Methodology: Refinement

[Removed]

## Results: Model Evaluation and Validation

[Removed]

## Results: Justification

[Removed]

## Conclusion: Free-Form Visualization

The main dashboard for exploring near three terabytes of data is the custom Plotly Dash application 'APEX Threat Simulation Laboratory'. This application is difficult to share due to the deep access to the underlying information, but the full source is provided in this document's Appendix and on my GitHub referenced below.

The main premise of the free-form visualization is to be able to rapidly access the overall prediction performance. The visualization is composed of four sections. The upper section contains the heading and scan slider. When you move the scan slider, the subject referenced changes in all three data views below.

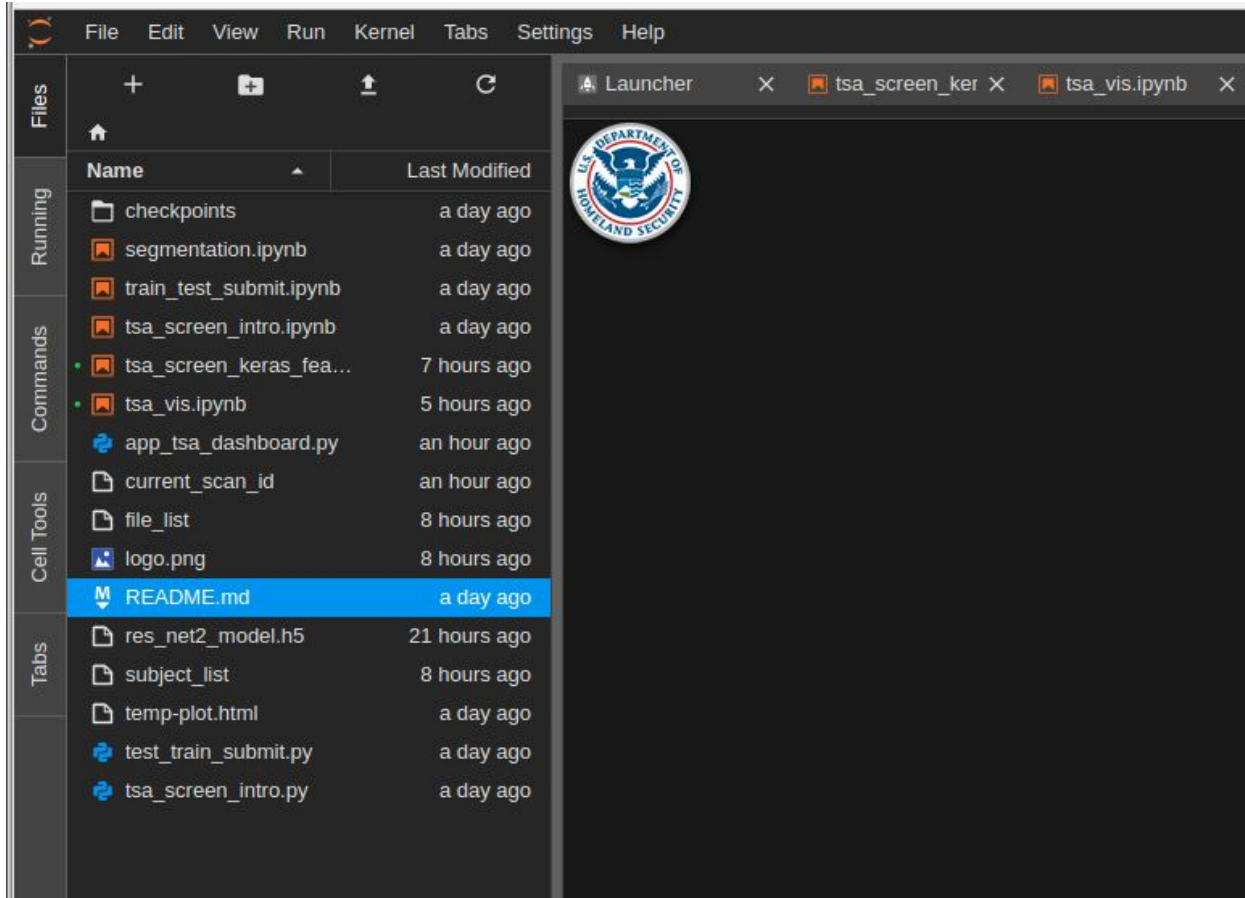
The top perspective is the small data view that consists of only 16 frames. The middle view contains 64 detailed frames using a compute intensive rendering. The bottom view contains 660 frames rendered as z-frames. All three views are updated by simply panning through the frames using the respective sliders. This is much easier as any subject can be referenced and completely explored with minimal effort.

Figs 27: Stage III - Custom Single Page Visualization App (



In reflecting over three iterations, the first static matplotlib renderings didn't offer the data scientist the ability to rapidly gain an understanding of the subjects and the data.

Figs 28: Reflection, Stage One in Jupyter-Lab building Single Frame Renderings



The scientist typed or updated code, etc.

The second iteration produced animated gifs and youtube videos using the static frames. These products ([e.g. single perspective video](#)) displayed helpful information, but the rate of consumption was not controlled by the information consumer.

Figs 29: Stage II Examples, Perspective Renderings - YouTube Unlisted Videos



DASHBOARD

VIDEO MANAGER

Videos

Playlists

LIVE STREAMING

COMMUNITY

CHANNEL

ANALYTICS

TRANSLATIONS & TRANSCRIPTIONS

CREATE

YOUR CONTRIBUTIONS

Help and feedback

YOUTUBE STUDIO BETA

Actions

Add to

0

Feb 18, 2018 9:28 PM

Edit

1

Feb 18, 2018 9:27 PM

Edit

2

Feb 18, 2018 9:27 PM

Edit

3

Feb 18, 2018 9:27 PM

Edit

4

Feb 18, 2018 9:27 PM

Edit

5

Feb 18, 2018 9:27 PM

Edit

Add music

6

Feb 18, 2018 9:27 PM

Edit

8

Feb 18, 2018 8:38 PM

Edit





## Conclusion: Reflection

[Removed]

## Conclusion: Improvement

[Removed]

## References

Liu, W. et al., 2015, SSD: Single Shot MultiBox Detector retrieved from

<https://arxiv.org/abs/1512.02325>

DHS Passenger Screening Kaggle Competition,

<https://www.kaggle.com/c/passenger-screening-algorithm-challenge>

Nei, W., Li X., and Su, Y. (2017). 3D object retrieval based on Spatial+LDA model. *Multimed Tools*

*Appl*, 76, 4091-4104, doi 10.1007/s11042-015-2840-x

Somogyi, A., Lovas, T., and Barsi, A. (2017). Comparison of Spatial Reconstruction Software

Package Packages Using DSLR Images. *Pollack Periodica: An International Journal for*

*Engineering and Information Sciences*, Vol 12 No 2, 17-27, doi 10.1556/606.2017.12.2.2


Bu, S. et. al. (2017). 3D shape recognition and retrieval based on multi-modality deep learning.

*Neurocomputing*, 259, 183-193, doi 10.1016/2016.06.088

Armin, Pohl. (2017). Computer generated images are the next marketing platforms for motor

vehicles. *Automotive Industry Online*, 259, 48-49, accessible via

[http://www.ai-online.com/Adv/Previous/show\\_issue.php?id=7018](http://www.ai-online.com/Adv/Previous/show_issue.php?id=7018)



Lentz, R. 2017. Classification of 132 Dog Breeds using Convolution Neural Networks + Transfer Learning, accessible via

[https://github.com/ricklentz/dog-project/blob/master/dog\\_app.ipynb](https://github.com/ricklentz/dog-project/blob/master/dog_app.ipynb)

Simonyan, K. and Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition (VGG), <https://arxiv.org/abs/1409.1556>

He, K. et. al (2015). Deep Residual Learning for Image Recognition,  
<https://arxiv.org/abs/1512.03385>

Szegedy, C. et al. (2014). Going Deeper with Convolutions (Inception),  
<https://arxiv.org/abs/1409.4842>

David Silver, et. al. 2016.  
<https://storage.googleapis.com/deepmind-media/alphago/AlphaGoNaturePaper.pdf>

Chis Burger 2017. <https://www.tastehit.com/blog/google-deepmind-alphago-how-it-works>

Seth Weidman 2018.  
<https://hackernoon.com/the-3-tricks-that-made-alphago-zero-work-f3d47b6686ef>

Sebastian Ruder 2017. <http://ruder.io/multi-task/index.html#introduction>



John Hessler 2016. <https://blogs.loc.gov/maps/category/game-theory/>

Ryszard Michalski 2001, Machine Learning and Inference Laboratory, George Mason University,  
Fairfax, VA LEARNABLE EVOLUTION MODEL: Evolutionary Processes Guided by Machine  
Learning

—

## Custom Exploratory Visualization Dashboard Code

(Code is colorized for ease of interpreting)

```
import plotly.plotly as py
from plotly.grid_objs import Grid, Column

import time
import numpy as np

from skimage import io

import os
from os import listdir
from os.path import isfile, join

import dash
import dash_core_components as dcc
import dash_html_components as html
import plotly.graph_objs as go
import pandas as pd

import pickle

import dash_html_components as html
import base64

vertical_slice_frames_path = r'/media/cbios/capstone_files/Kaggle/vertical_slice_frames'
horizontal_slice_frames_path = r'/media/cbios/capstone_files/Kaggle/horizontal_slice_frames/'

# dynamically generate slice list
if os.path.exists('file_list'):
    with open('file_list', 'rb') as fp:
        all_slices = pickle.load(fp)
else:
    all_slices = [f for f in listdir(vertical_slice_frames_path) if isfile(join(vertical_slice_frames_path, f))]
    with open('file_list', 'wb') as fp:
        pickle.dump(all_slices, fp)

# dynamically generate unique subject scan id list
if os.path.exists('subject_list'):
    with open('subject_list', 'rb') as fp:
        subject_scan_set = pickle.load(fp)
        subject_scan_set = sorted(subject_scan_set)
else:
    subject_scan_set = set()
    for f in all_slices:
        if len(f) > 10:
            subj_scan_id = f.split(' ')[0].replace('a3d', '')
            subject_scan_set.add(subj_scan_id)
    print(len(subject_scan_set))

    with open('subject_list', 'wb') as fp:
        pickle.dump(subject_scan_set, fp)

# dynamically update s

app = dash.Dash()

colors = {
    'background': '#111111',
    'text': '#EEEEEE',
    'font': '30px',
}
```

\_\_\_\_\_

```
html.Img(id='small_data',src='data:image/png;base64,{}'.format(small_encoded_image.decode())),
dcc.Slider(
    id='small-slider',
    min=0,
    max=15,
    marks={i: 'Label {}'.format(i) if i == 1 else str(i) for i in range(0, 15)},
    value=small_slider_value,
    className='overview',
)
)),

html.Div([
    html.Img(id='medium_data',src='data:image/png;base64,{}'.format(medium_encoded_image.decode())),
    dcc.Slider(
        id='medium-slider',
        min=0,
        max=63,
        marks={i: 'Label {}'.format(i) if i == 1 else str(i) for i in range(0, 63)},
        value=medium_slider_value,
    )
],className='medium-view'),
html.Div([
    html.Img(id='large_data',src='data:image/png;base64,{}'.format(large_encoded_image.decode())),
    dcc.Slider(
        id='large-slider',
        min=0,
        max=659,
        marks={i: 'Label {}'.format(i) if i == 1 else str(i) for i in range(0, 659)},
        value=large_slider_value,
    )
],className='large-view'),

pp.callback(
    dash.dependencies.Output('small-slider', 'disabled'),
    [dash.dependencies.Input('scan-slider', 'value')])
f update_selected_subject(selected_uuid):
    with open('current_scan_id', 'wb') as fp:
        pickle.dump(selected_uuid, fp)
    update_small_source(small_slider_value)
    update_medium_source(medium_slider_value)
    update_large_source(large_slider_value)
    return False

small-slider & small_data
pp.callback(
    dash.dependencies.Output('small_data', 'src'),
    [dash.dependencies.Input('small-slider', 'value')])
f update_small_source(s_slider_value):
    with open ('current_scan_id', 'rb') as fp:
        current_scan_id = pickle.load(fp)
    small_slider_value = s_slider_value
    file_name = horizontal_slice_frames_path+subject_scan_set[current_scan_id]+r'/' +str(small_slider_value)+'.png'
    print("shit")
    small_encoded_image = base64.b64encode(open(file_name, 'rb').read())
    return 'data:image/png;base64,{}'.format(small_encoded_image.decode())

medium-slider & medium_data
pp.callback(
    dash.dependencies.Output('medium_data', 'src'),
    [dash.dependencies.Input('medium-slider', 'value')])
f update_medium_source(m_slider_value):
    with open ('current_scan_id', 'rb') as fp:
        current_scan_id = pickle.load(fp)
```



—



