Thriving Skies

AI Modeling for X-Ray Baggage Screenings

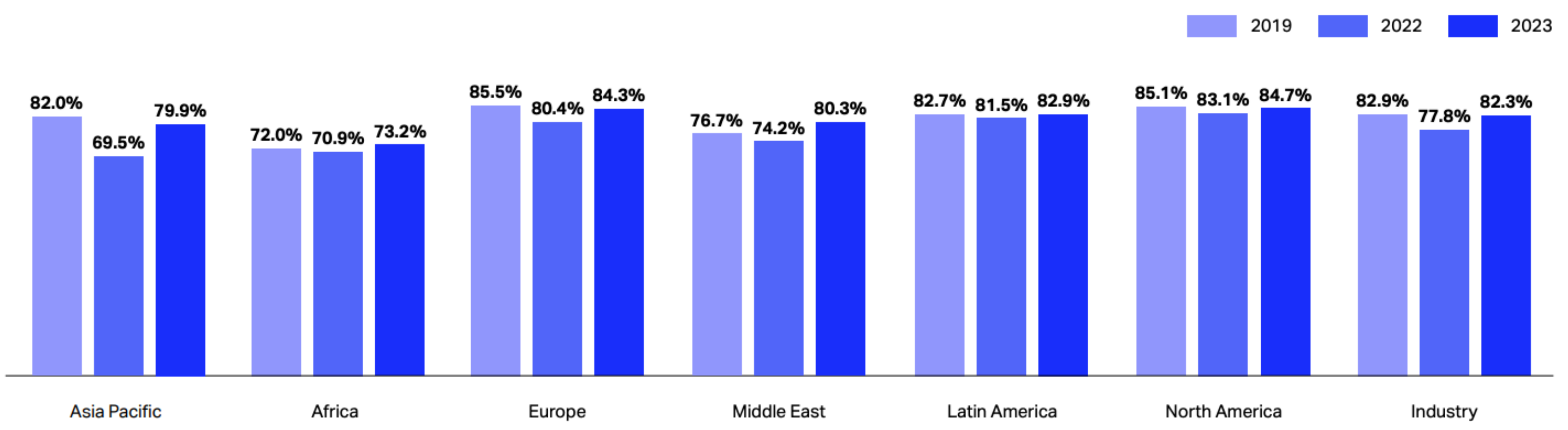
Ryan Busman, Elaine Kwan, Deb Peters, Nick Watkins

April 22, 2024

The Aviation Industry

In 2023, air passenger traffic had reclaimed its pre-pandemic vigor, soaring to 94.1% of 2019 levels.

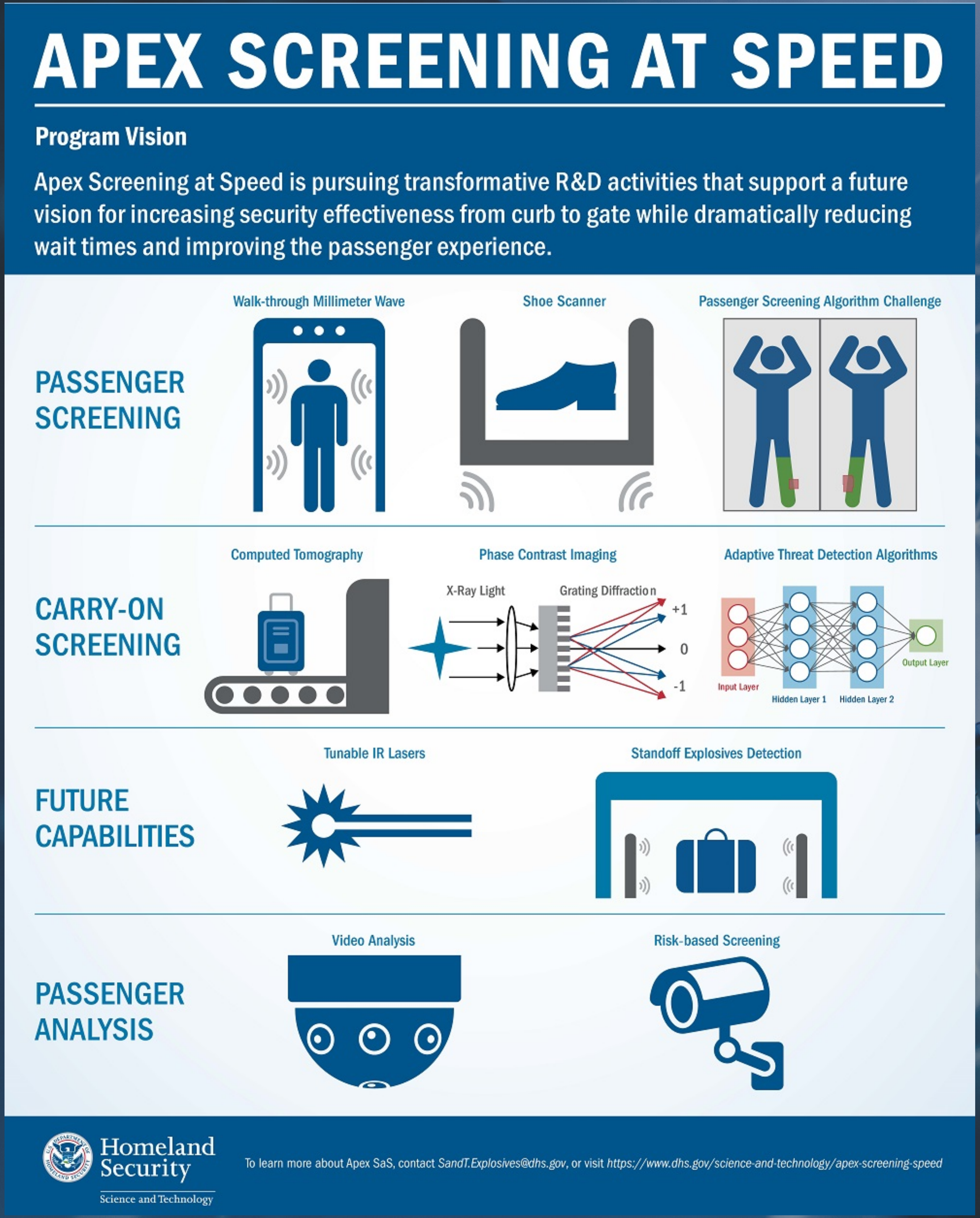
The International Air Transport Association (IATA) anticipates a banner year in 2024, with the global airline passenger industry poised to achieve record growth.



Sources: IATA Sustainability and Economics, IATA Monthly Statistics

In 2024...

- 4.7 billion passengers
- $964 billion in revenue
- $49.3 billion in operating profit



X-Ray Baggage Screening

Transportation security administration (TSA) personnel screen approximately 2 million passengers a day at 440 US airports for a total of approximately 3.3 million carry-on bag screenings.

False positives, bag checks, and manual security measures, not only disrupt the flow of passengers but also pose significant costs and privacy concerns.

GOALS

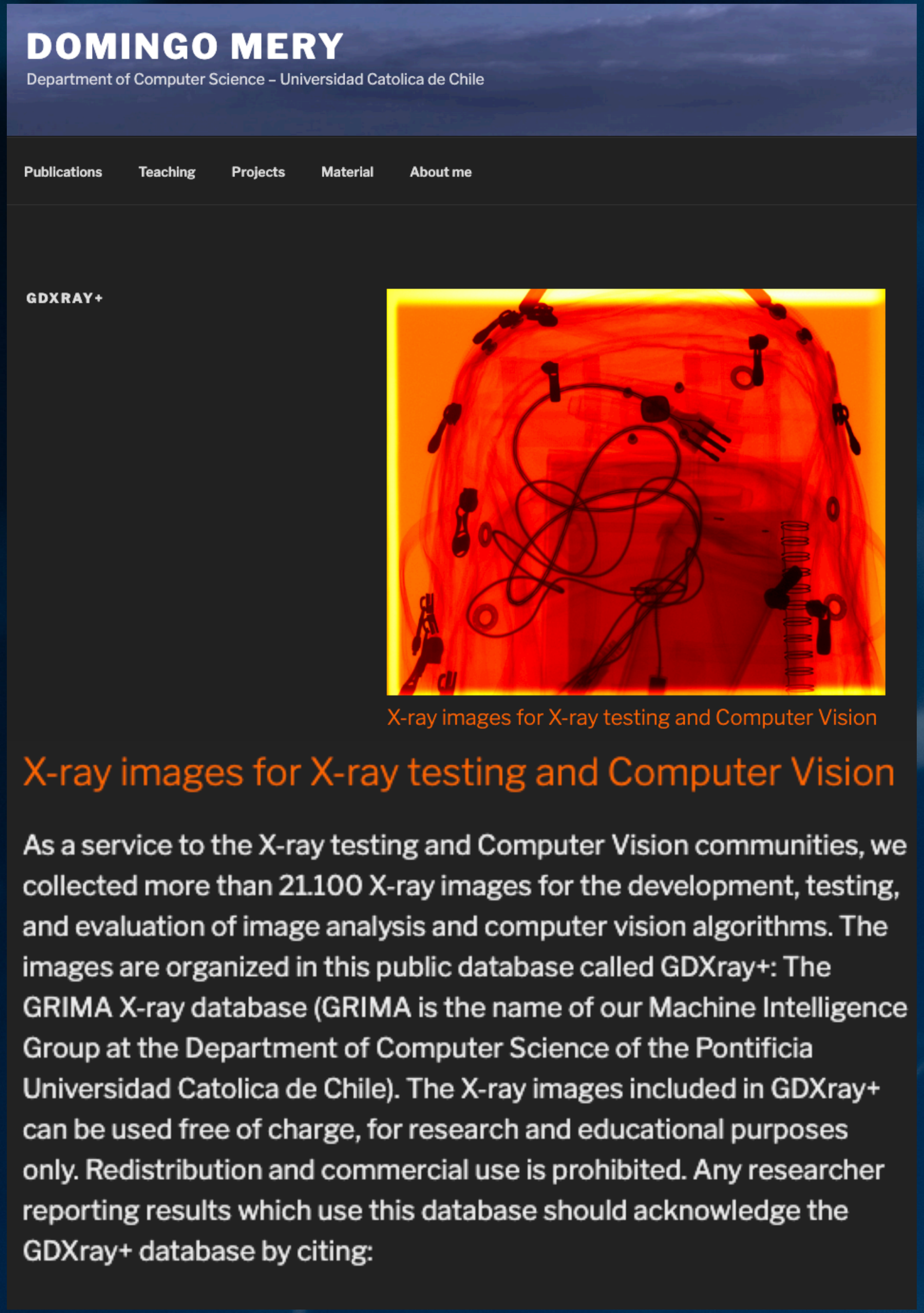
1: Improve Accuracy

Improve accuracy of dangerous object detection in carry-on passenger bags and for personal objects

2: A Screening Interface

Create a webs-based interactive tool that can be used to assess the likelihood of a dangerous object's presence

Data Fetching & Preprocessing



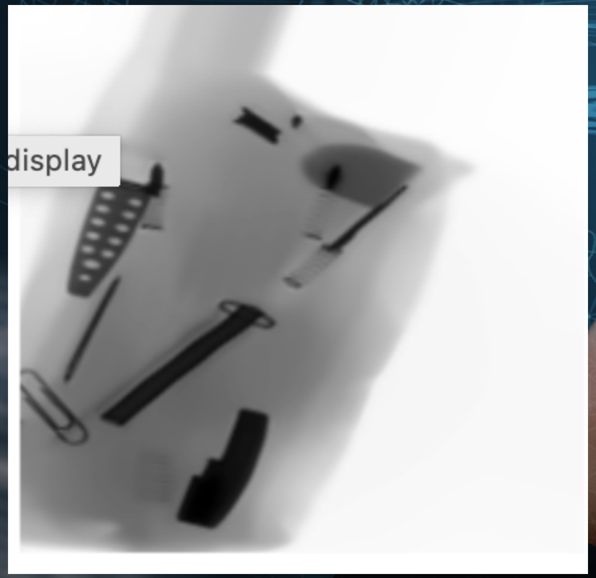
https://domingomery.ing.puc.cl/material/gdxray/

GDXRAY+ Dataset: Baggage Subset of 8,150 X-ray images arranged into 77 series

1. Loaded and inspected the dataset after visually filtering and classifying the images (i.e., several were excluded for being too dark or ambiguous).
2. We used 3,385 original images from 5 categories objects: benign, gun, knife, razor blade or shuriken (throwing star)
3. Checked the file path, image sizes, image modes and image format
4. Images converted to a NumPy Array and stored to an empty list
5. Resized the images to the standard target size of 224 x 224 pixels
6. Normalized the pixel values using the PIL library.
7. **Required additional library or technology NOT covered in class** was OpenCV library to remove unwanted noise from the images
8. Images converted to a consistent greyscale using the PIL Library
9. Augmented the images using the Keras ImageDataGenerator; this generated 10,155 images
10. All total, the X-trained dataset had 13,540 images.

Data Augmentation

Augmentations improve the performance of deep learning models for computer vision tasks such as classification, segmentation, and object detection.



Model Selection and Architecture

A Convolutional Neural Network (CNN)

The layers are arranged in such a way so that they detect simpler patterns first (lines, curves, etc.) and more complex patterns (faces, objects, etc.) further along.

Our model had 9 layers: an input layer, output layer, and 7 hidden layers.

```
# define input shape
input_shape = X_test_np[0].shape
# Define CNN model
model = keras.Sequential([
    layers.Input(shape=input_shape),
    layers.Conv2D(32, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(5, activation='sigmoid')  # 5 classes of images
])

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
batch_size = 32#64
epochs = 6
history = model.fit(
    X_train, y_train,
    validation_data=(X_test, y_test),
    epochs=epochs
)
```

Training and Testing

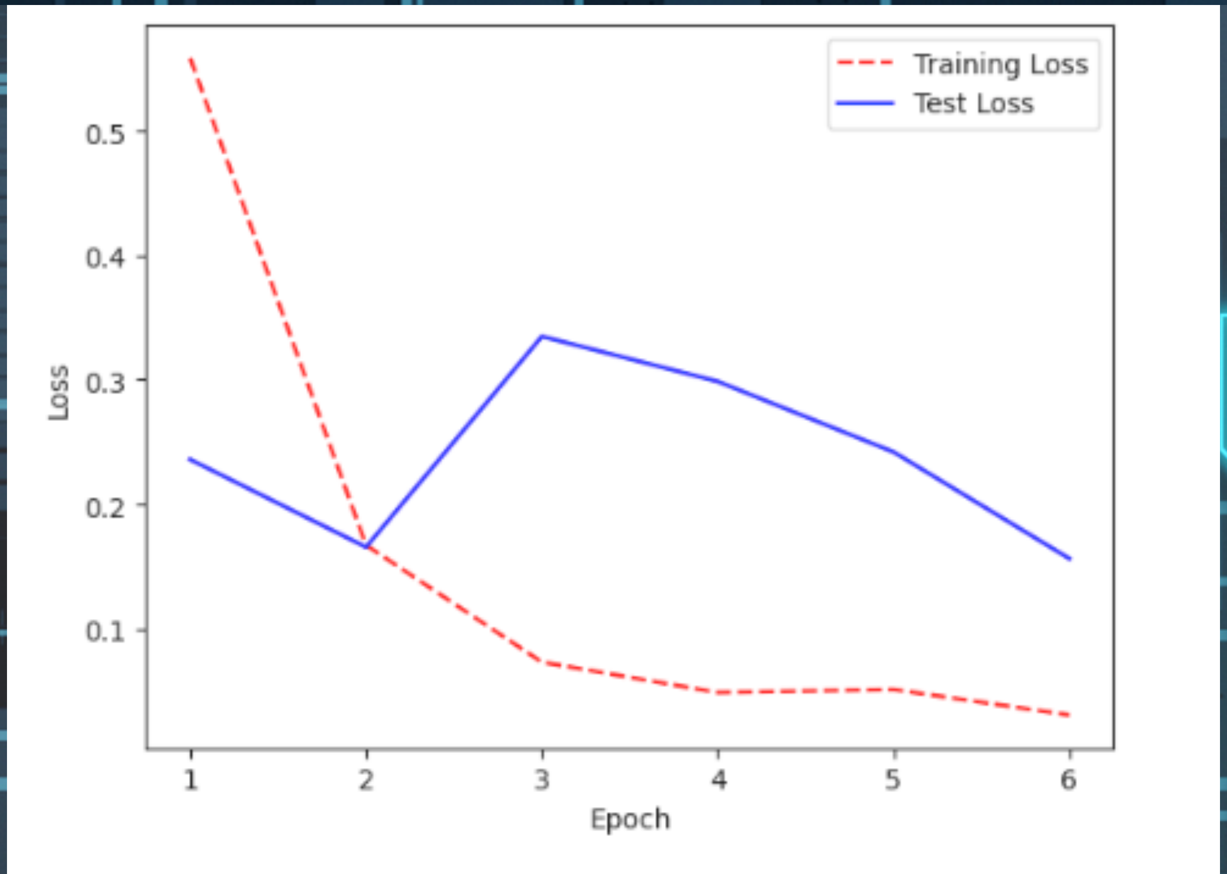
Hyperparameters include the:

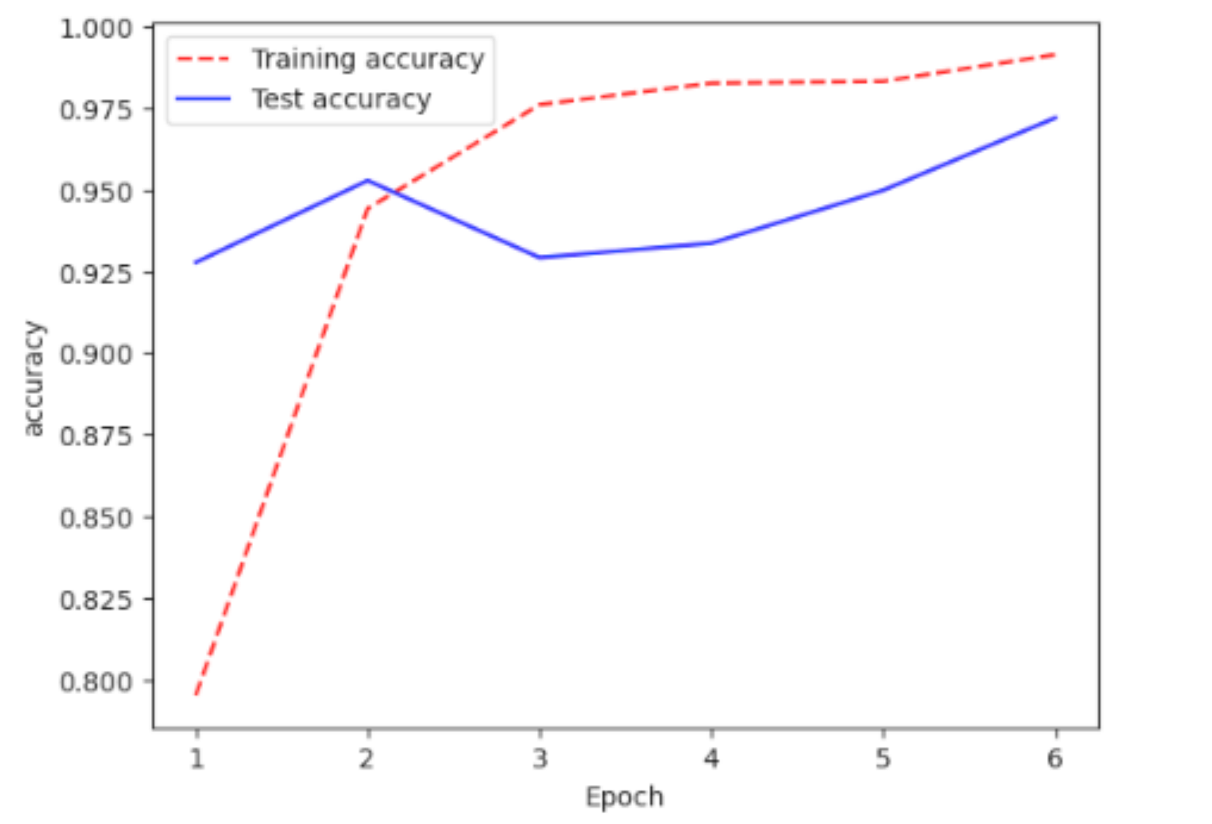
- Number of neurons
- Activation functions
- Optimizer,
- Learning rate
- Batch size
- Epochs

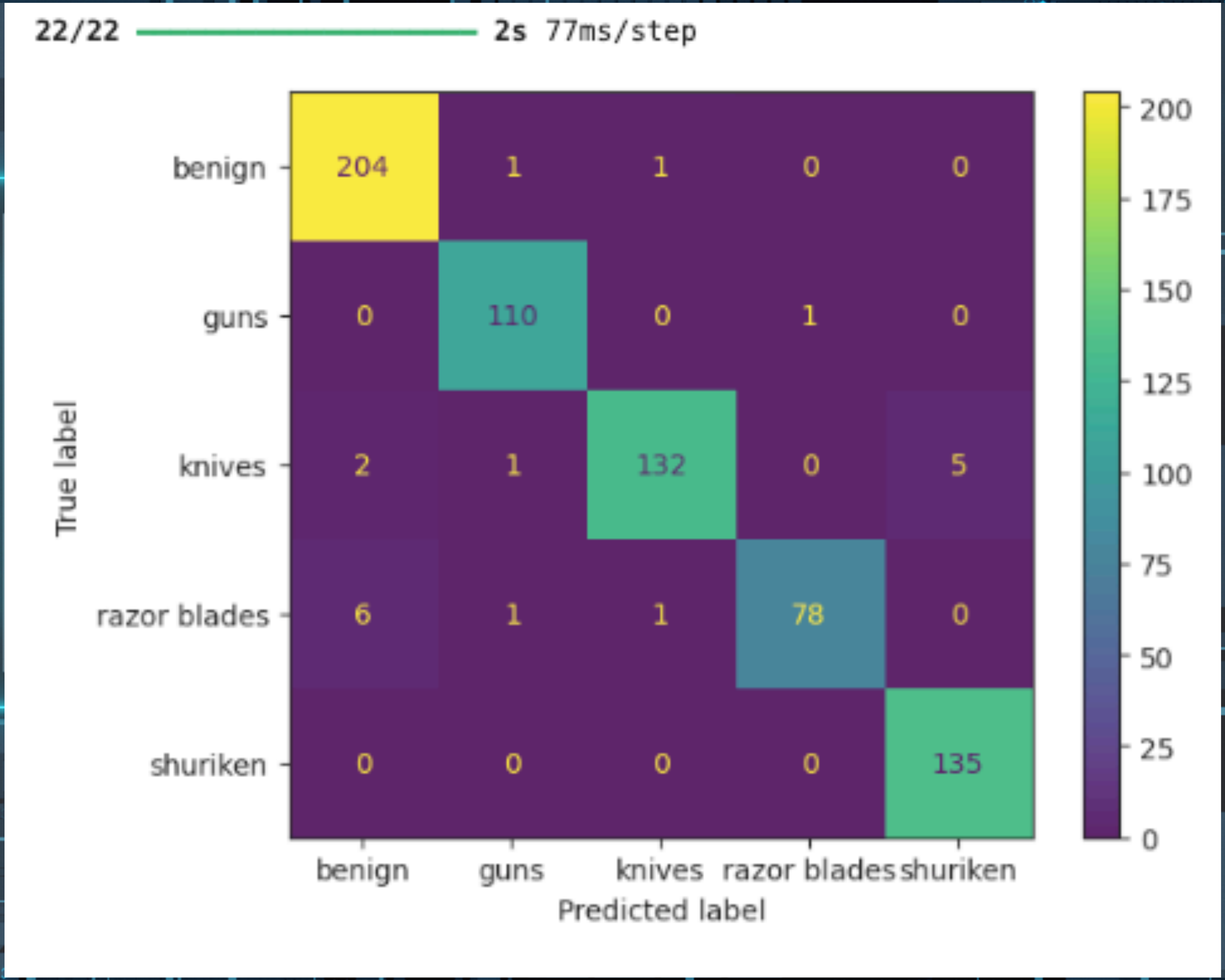
```python
# define input shape
input_shape = X_test_np[0].shape
# Define CNN model
model = keras.Sequential([
    layers.Input(shape=input_shape),
    layers.Conv2D(32, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(5, activation='sigmoid')  # 5 classes of images
])

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
batch_size = 32#64
epochs = 6
history = model.fit(
    X_train, y_train,
    validation_data=(X_test, y_test),
    epochs=epochs
)
```

```
Epoch 1/6
424/424 ━━━━━━━━━━━━━━━━━━━━ 128s 301ms/step - accuracy: 0.6809 - loss: 0.9287 - val_accuracy: 0.9277 - val_loss: 0.2360
Epoch 2/6
424/424 ━━━━━━━━━━━━━━━━━━━━ 123s 289ms/step - accuracy: 0.9383 - loss: 0.1794 - val_accuracy: 0.9528 - val_loss: 0.1657
Epoch 3/6
424/424 ━━━━━━━━━━━━━━━━━━━━ 123s 289ms/step - accuracy: 0.9763 - loss: 0.0733 - val_accuracy: 0.9292 - val_loss: 0.3349
Epoch 4/6
424/424 ━━━━━━━━━━━━━━━━━━━━ 124s 293ms/step - accuracy: 0.9832 - loss: 0.0475 - val_accuracy: 0.9336 - val_loss: 0.2987
Epoch 5/6
424/424 ━━━━━━━━━━━━━━━━━━━━ 126s 296ms/step - accuracy: 0.9858 - loss: 0.0463 - val_accuracy: 0.9499 - val_loss: 0.2420
Epoch 6/6
424/424 ━━━━━━━━━━━━━━━━━━━━ 126s 298ms/step - accuracy: 0.9918 - loss: 0.0286 - val_accuracy: 0.9720 - val_loss: 0.1564
```

Evaluation and Fine Tuning







X-Ray Model in Action

User choose from one of 678 test images/data

User inputs an image number

Model predicts if the image is benign, gun, knife, razor blade or shuriken (throwing star)

```
number of images to chose from = 678
```

```
# call to functions to plot predictions based on any of the test images selected by user
# this is the input from user
input = 0
# call to functions to plot image #= input and to plot probability
model_predictions(input)
```

[89] 0.0s

```
There are only 678 images to chose from.
```

```
# doesn't work properly
# #create gradio app
# with gr.Blocks() as app:
#     # fill row with text boxes
#     with gr.Row():
#             input = gr.Textbox(label="image number")
#     # fill next row
#     with gr.Row():
#         # fill columns
#         with gr.Column(scale=3, min_width=800):
#             #create enter button
#             enter_btn = gr.Button("enter")
#             # create output textbox filled with dataframe
#             output = gr.Image(label="")
#     # define function of enter button
#     enter_btn.click(fn=model_predictions, inputs=input,
#                     outputs=output, api_name="display")
# # start app
# app.launch()
```

[17]

```
Running on local URL:  http://127.0.0.1:7861

To create a public link, set `share=True` in `launch()`.
```

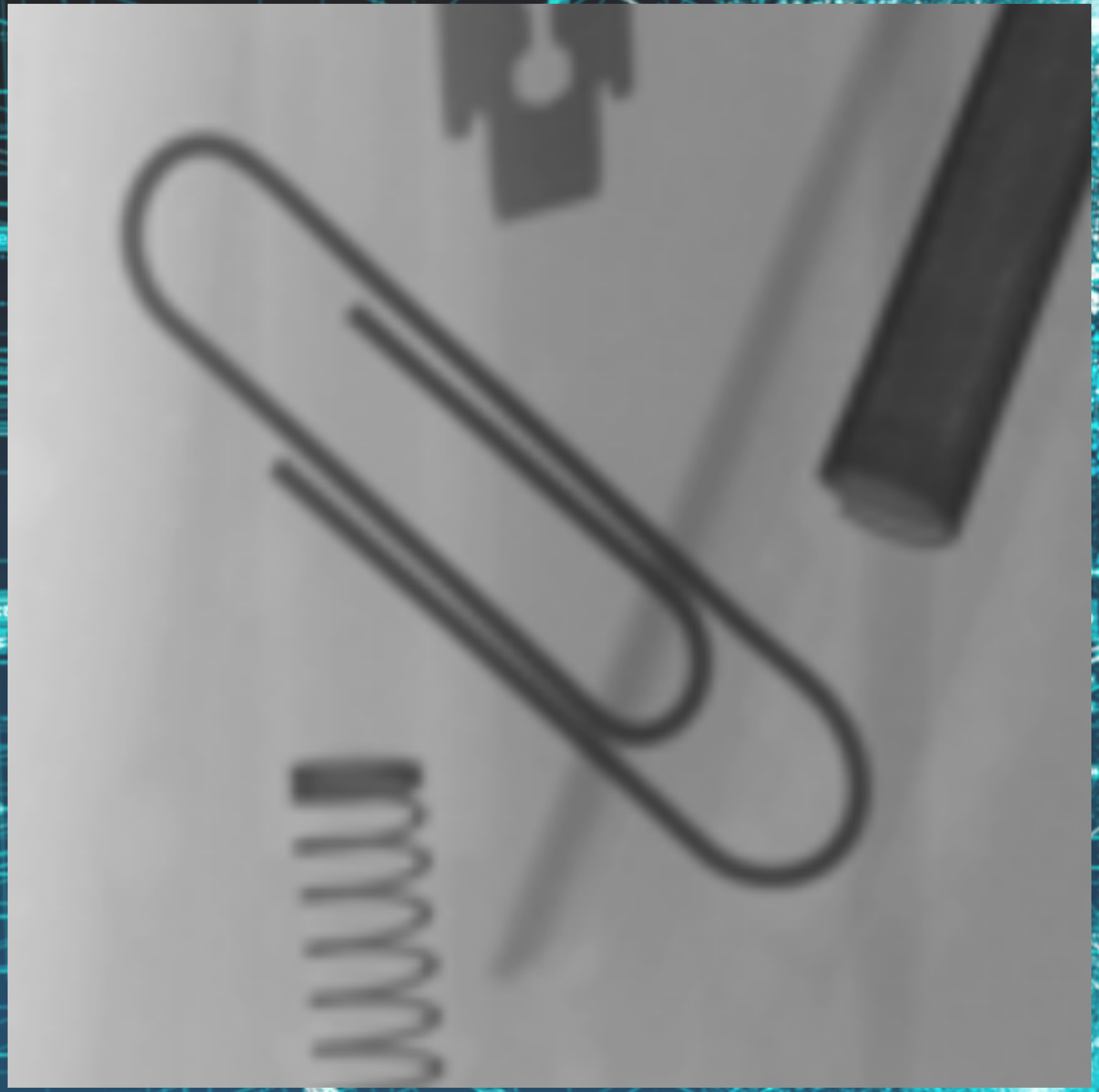
Primary Challenge

Data from trusted sources

The first two datasets had malicious code that targeted the MacBook operating system.

Nick and Ryan had to reset and restore their systems from the data downloaded from Google Drive links in Github repositories.

Other Challenges



Sorting the images (Razor, Knife, or Benign?)

Smoothing out of images

Crop or not? Large images take a long time to process versus fuzziness/lower resolution from images that are too small

File size workarounds (Github size; Shuffles with local drives)

The Now...

THE WALL STREET JOURNAL.

Billions in Dirty Money Flies Under the Radar at World's Busiest Airports

The Heathrow-to-Dubai flights have two big money-laundering features: One airport doesn't scan outbound luggage for cash and the other welcomes sacks of it

Future Considerations

Usability with Gradio

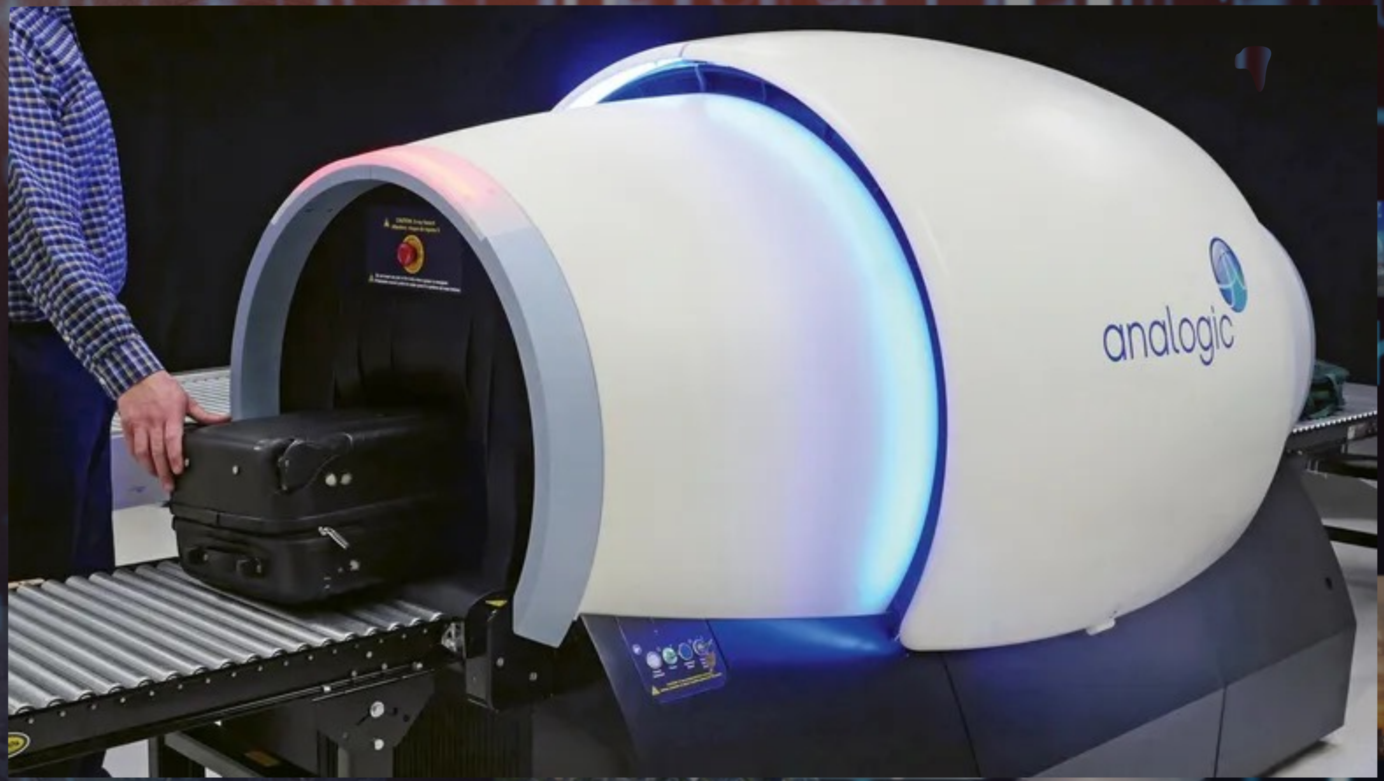
Scattered X-Rays for material composition

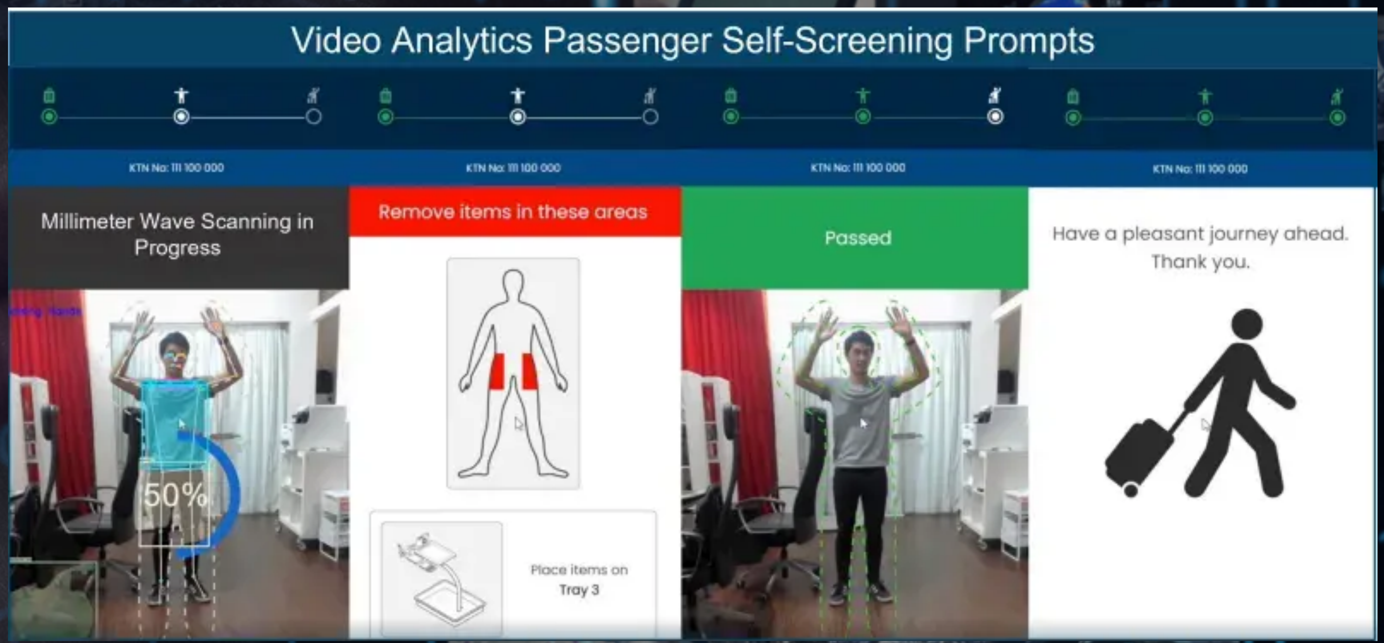
3D CT images

Bio-sensing

Prompts for self-screening lanes

Voxel Radar: In-motion panel sensors





Thank You!

The Project X-Ray Team