

# PROJECT ODIN

---

O PTICAL

D ISTRIBUTED

I NTEGRATED AIR DEFENCE

N ETWORK COMPONENT

# Modern Air Warfare

S.A.

# The Air Defence Network

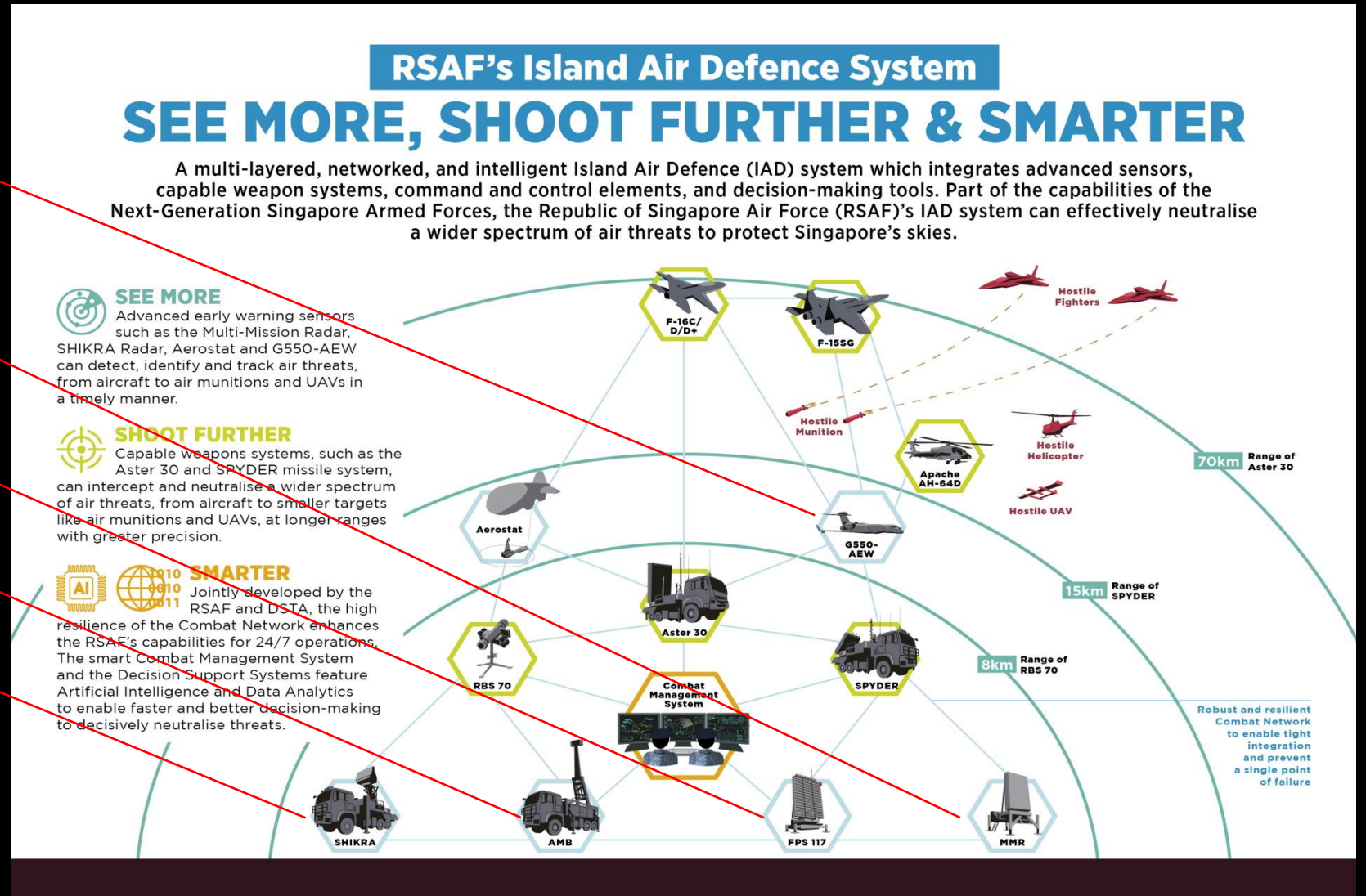
US\$ 330 million

US\$ 14 million

US\$ 14 million

US\$ 3 million

US\$ 23 million



# Problems

- Certain missiles can home on radars
- Aircraft have radar warning receivers
- Cameras and GPUs are cheap relative to radars, and offer a capability boost for minimal expenditure

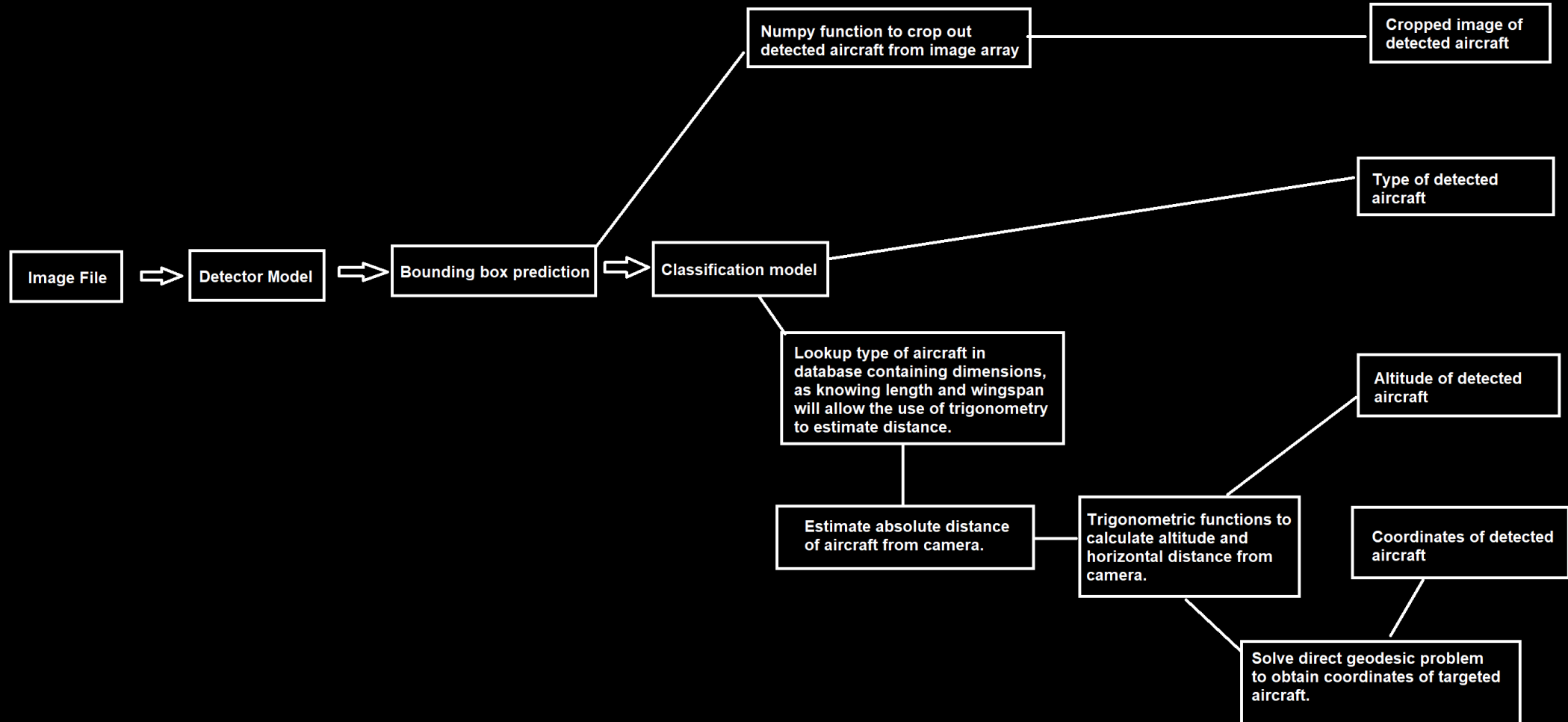
# Objectives

Take an image of an aircraft, location of camera, and information on where it is pointing

And RETURN:

Type of aircraft, coordinates, and altitude.

# Objectives – The Plan



# Project scope

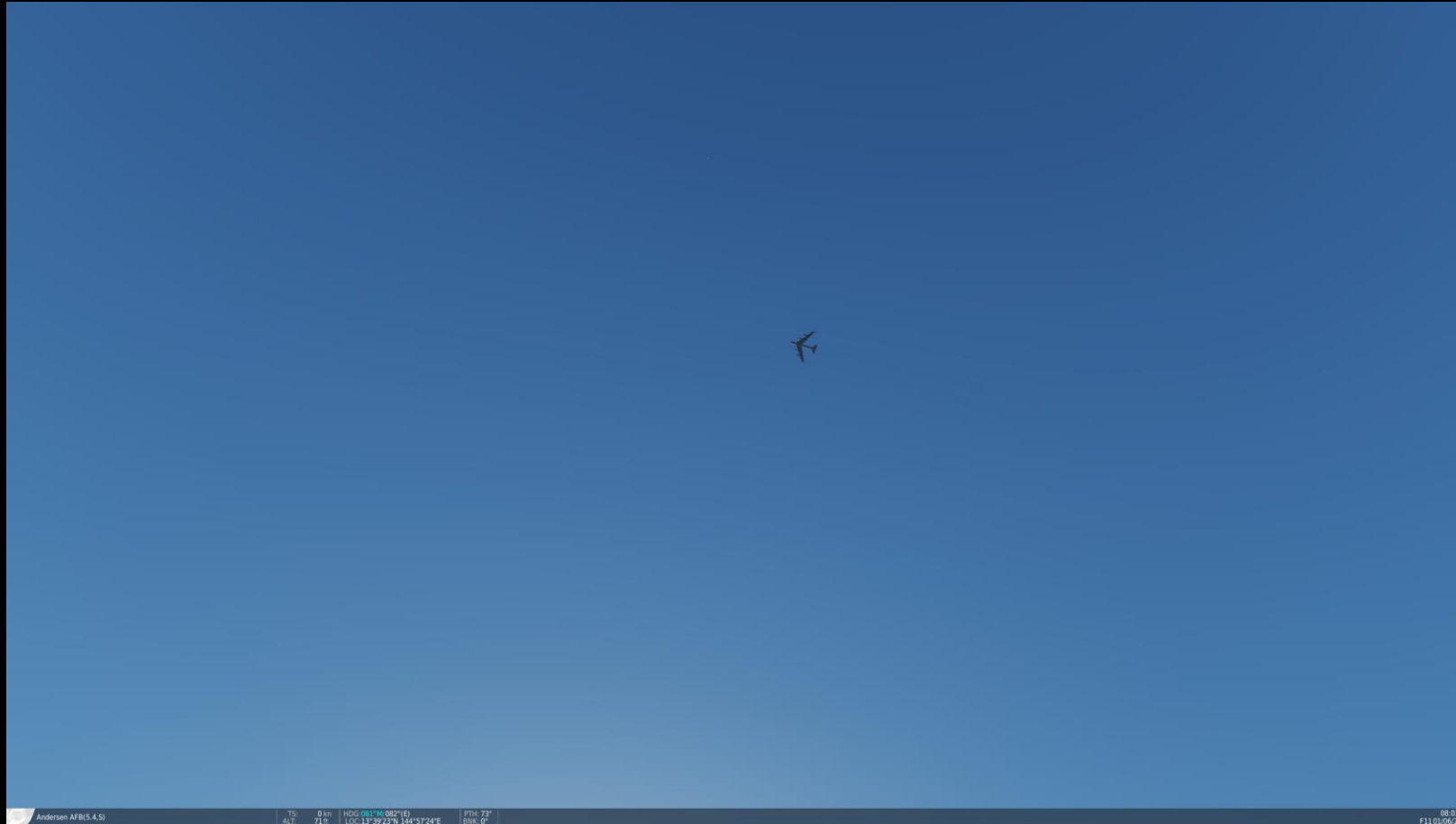
- Detector model
- Classifier model
- Cropping function
- Aircraft dimension database
- Distance calculation function
- Various trigonometric functions
- Geodesic solver function

# Success Metrics

- Detection Model – Precision + Recall
- Classification Model – Accuracy
- Program – from image of sky with aircraft, produce:
  - cropped .jpg image of aircraft;
  - aircraft type (in .json);
  - aircraft coordinates (in .json); and
  - aircraft altitude (in .json).



# We want to go from this:



# To this:



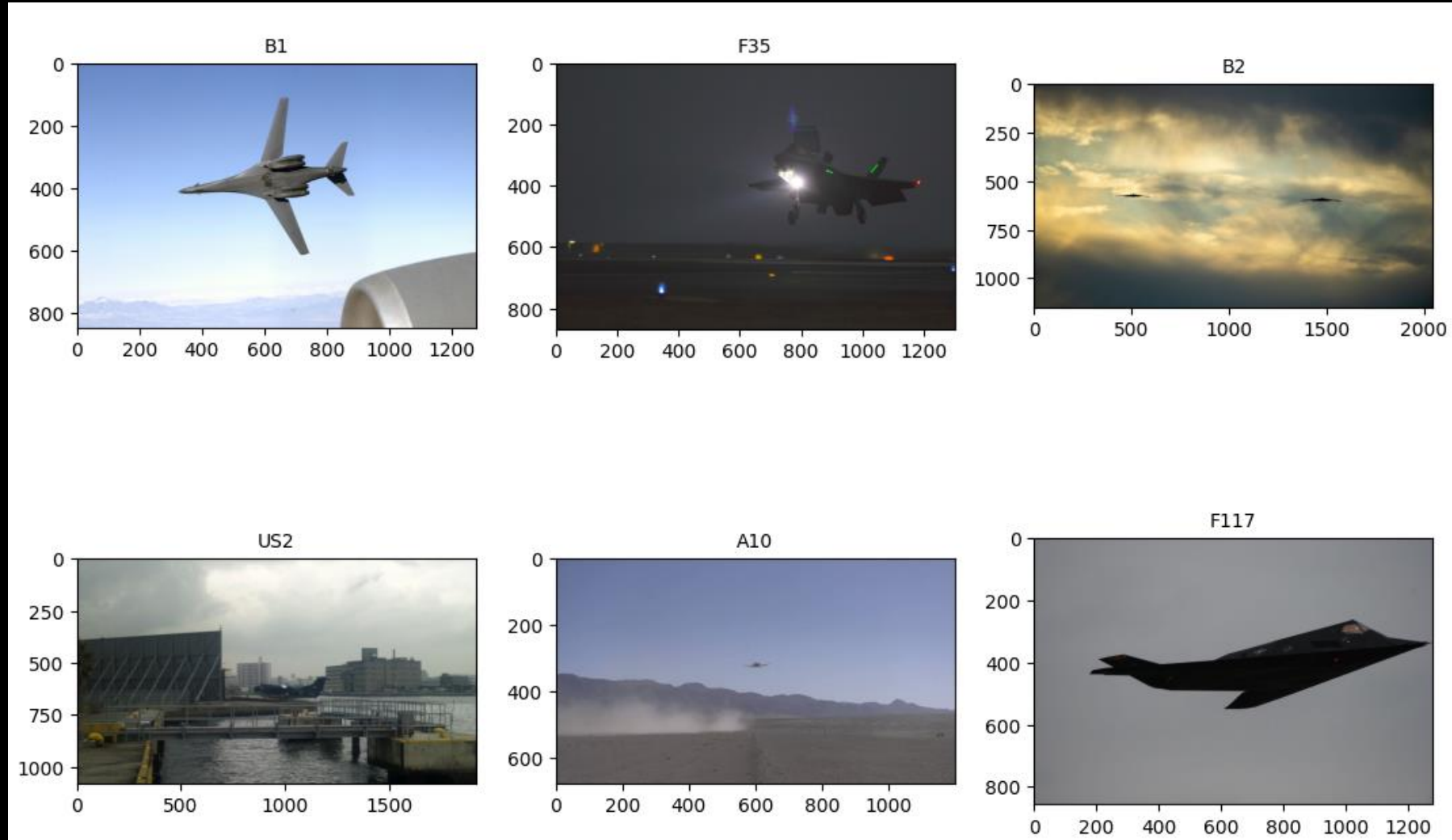
```
{  
  "Coordinates": [13.656864281456352,  
144.96277581585085],  
  "Altitude": 5875.418048686289,  
  "Type": "B52"  
}
```

# Dataset

A collection of annotated photographs of aircraft, annotated with type and bounding box information.

Accompanied by a collection of images of aircraft cropped from the same image set.

# Sample of Complete Images

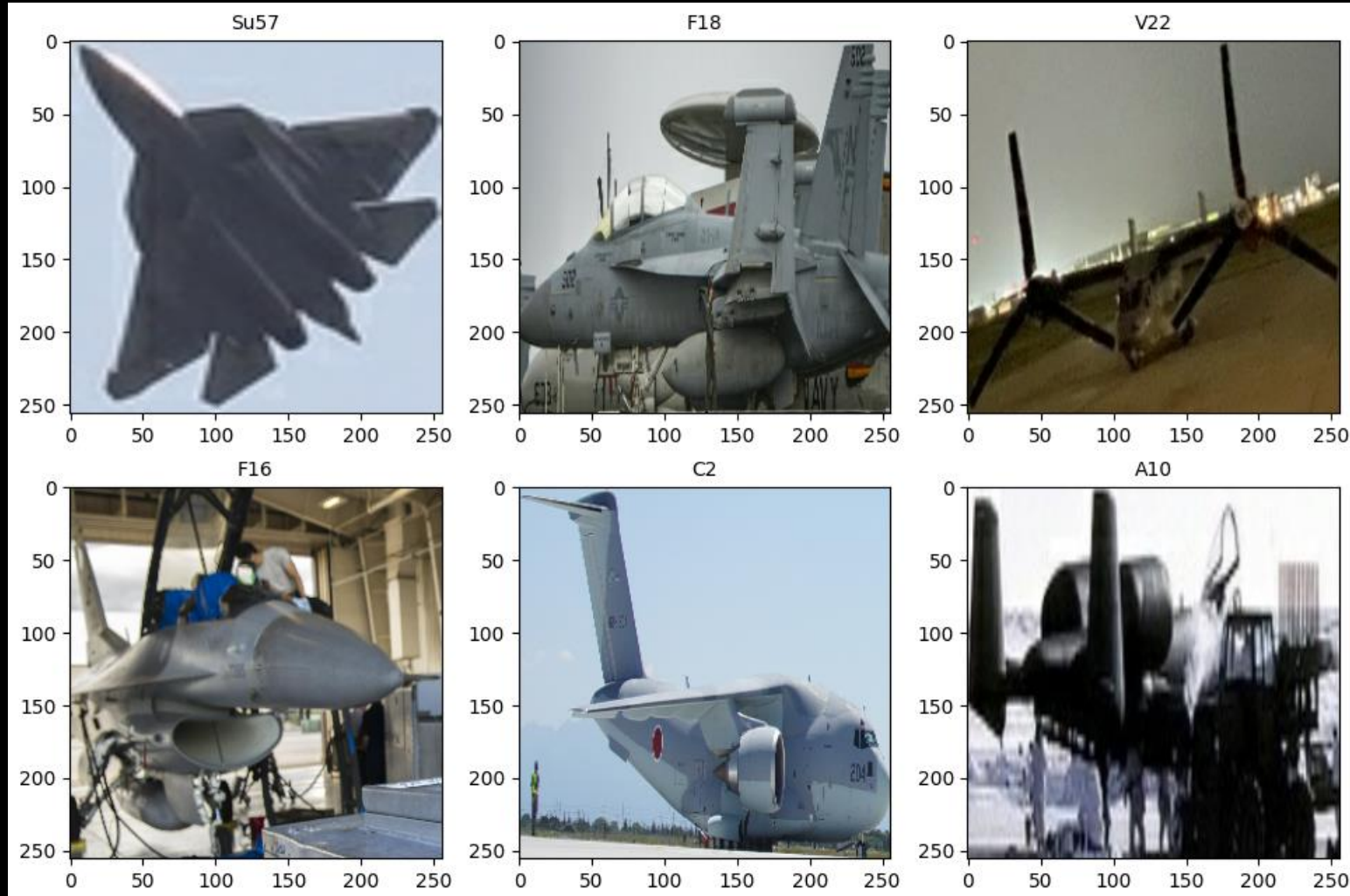


# Annotations

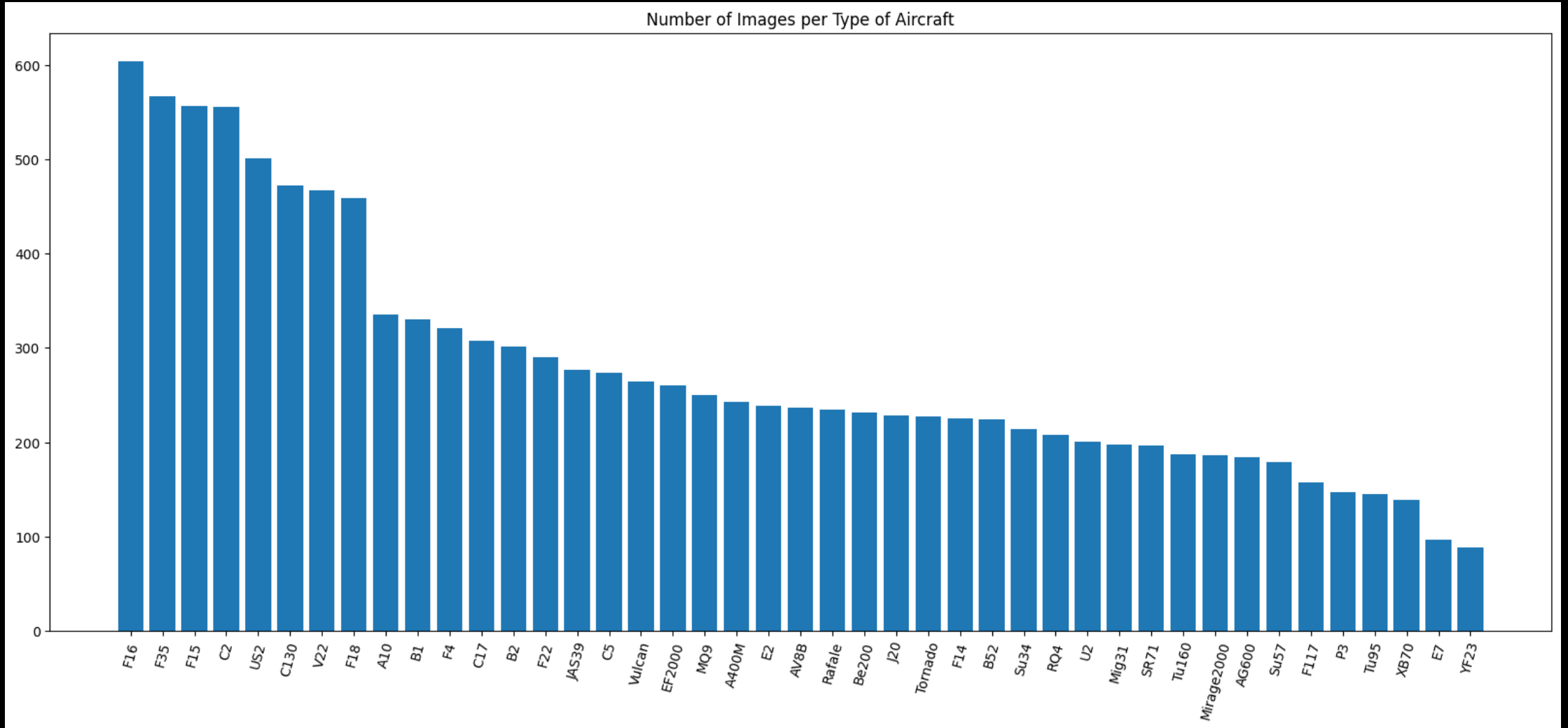
Each image file is accompanied by a .csv file, sharing the same name, containing the relevant annotations:

filename	width	height	class	xmin	ymin	xmax	ymax
0a1b628512650c43246c152c409f9c41	1200	800	US2	773	307	929	354

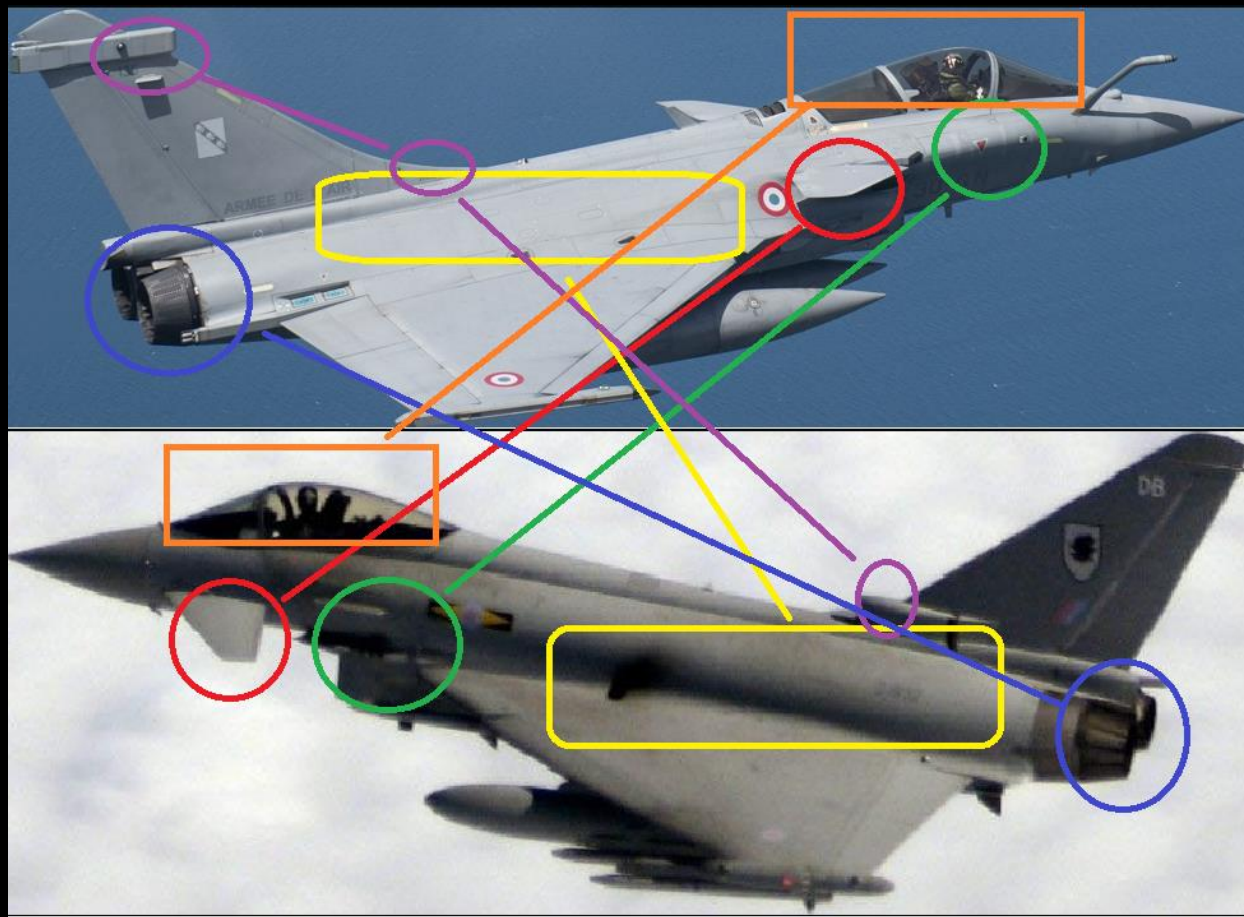
# Sample of cropped images



# Class Distribution

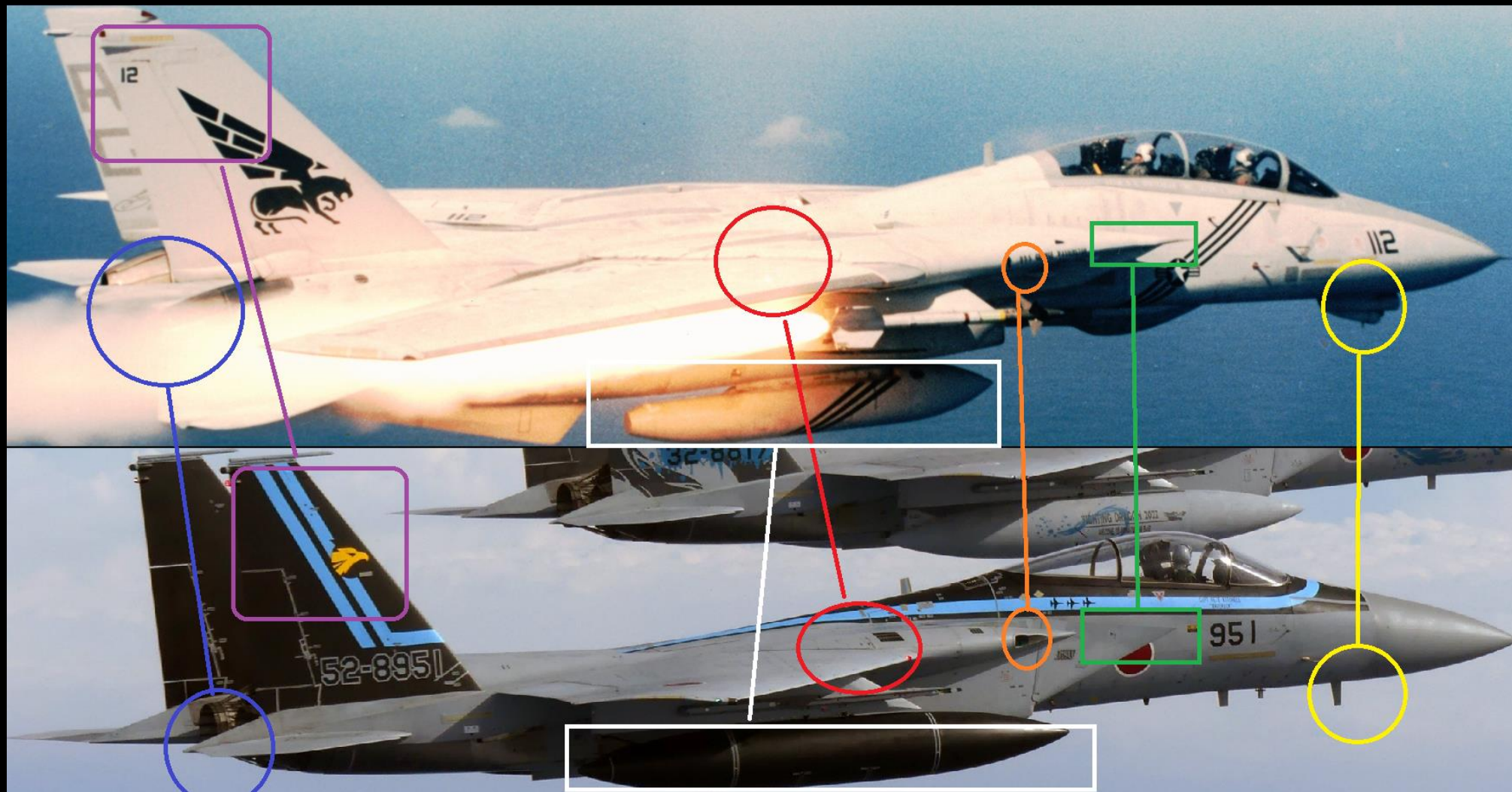


# Feature identification















# Feature identification



# Detection Model – Data Preparation

- The dataset had to be split into train-test-val, and the split had to be stratified by class.
- The data arrived in a single directory, and there is no library which automates the split.

 000aa01b25574f28b654718db0700f72.csv	2/11/2023 7:24 pm	Microsoft Exce...	1 KB
 000aa01b25574f28b654718db0700f72.jpg	2/11/2023 7:24 pm	JPG File	57 KB
 000e7662268a1071827c5a8663e773f9.csv	2/11/2023 7:24 pm	Microsoft Exce...	1 KB
 000e7662268a1071827c5a8663e773f9.jpg	2/11/2023 7:24 pm	JPG File	128 KB
 000ec980b5b17156a55093b4bd6004ab.csv	2/11/2023 7:24 pm	Microsoft Exce...	1 KB
 000ec980b5b17156a55093b4bd6004ab.jpg	2/11/2023 7:24 pm	JPG File	56 KB
 00a174c977e5a4c112422ddbec8d9266.csv	2/11/2023 7:24 pm	Microsoft Exce...	1 KB
 00a174c977e5a4c112422ddbec8d9266.jpg	2/11/2023 7:24 pm	JPG File	7,572 KB
 00a146155690879cedf0b20325b0ba18.csv	2/11/2023 7:24 pm	Microsoft Exce...	1 KB
 00a146155690879cedf0b20325b0ba18.jpg	2/11/2023 7:24 pm	JPG File	317 KB

# Detection Model – Data Preparation

Solution to the splitting problem:

- Iterate through entire directory, build dataframe of filenames and aircraft type.
- Use scikit-learn's train test split to do a stratified split with class as y.
- Iterate through dataframes of train, test, and val filenames to copy files into train, test, and val directories.

# Detection Model – Data Preparation

- Detection to be done using Ultralytics library, which requires a .yaml file to tell the model where to look, and txt files to tell the model where the bounding boxes are.
- When copying files to train, test, and val directories, a .txt files containing bounding box coordinates had to be created as well.

# Modelling results











Model	Epochs	Precision	Recall	mAP50	Remarks
RT-DETR	100	0.953	0.866	0.902	This performance is acceptable
RT-DETR	100	0.936	0.875	0.905	With additional augmentation
RT-DETR	100	0.953	0.849	0.889	With less augmentation
RT-DETR	100	0.96	0.868	0.902	Using cosine LR scheduler
YOLOv8	100	0.89	0.778	0.883	Somewhat worse performance

# Why select that model?

- The models all perform similarly, with the RT-DETR models performing slightly better than the YOLO model.
- This is likely due to the fact that YOLO is a CNN, and we are losing information in the pooling layers.
- Due to the close performance, we should just choose the one with the highest metrics.

# Classification Model

This was much more straightforward as the data arrived neatly sorted into directories with names corresponding to the aircraft types.

Name	Date modified	Type	Size
 A10	2/11/2023 7:20 pm	File folder	
 A400M	2/11/2023 7:20 pm	File folder	
 AG600	2/11/2023 7:20 pm	File folder	
 AV8B	2/11/2023 7:20 pm	File folder	
 B1	2/11/2023 7:20 pm	File folder	
 B2	2/11/2023 7:20 pm	File folder	
 B52	2/11/2023 7:21 pm	File folder	
 Be200	2/11/2023 7:21 pm	File folder	
 C2	2/11/2023 7:21 pm	File folder	
 C5	2/11/2023 7:21 pm	File folder	

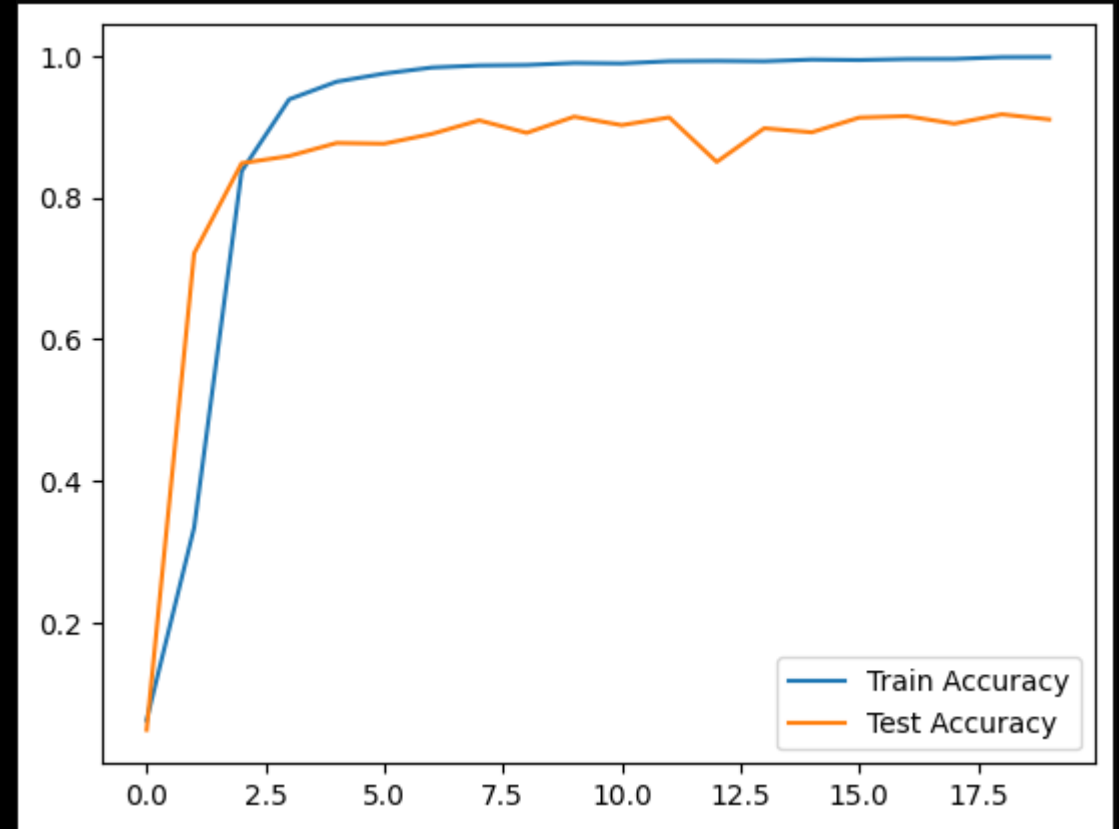
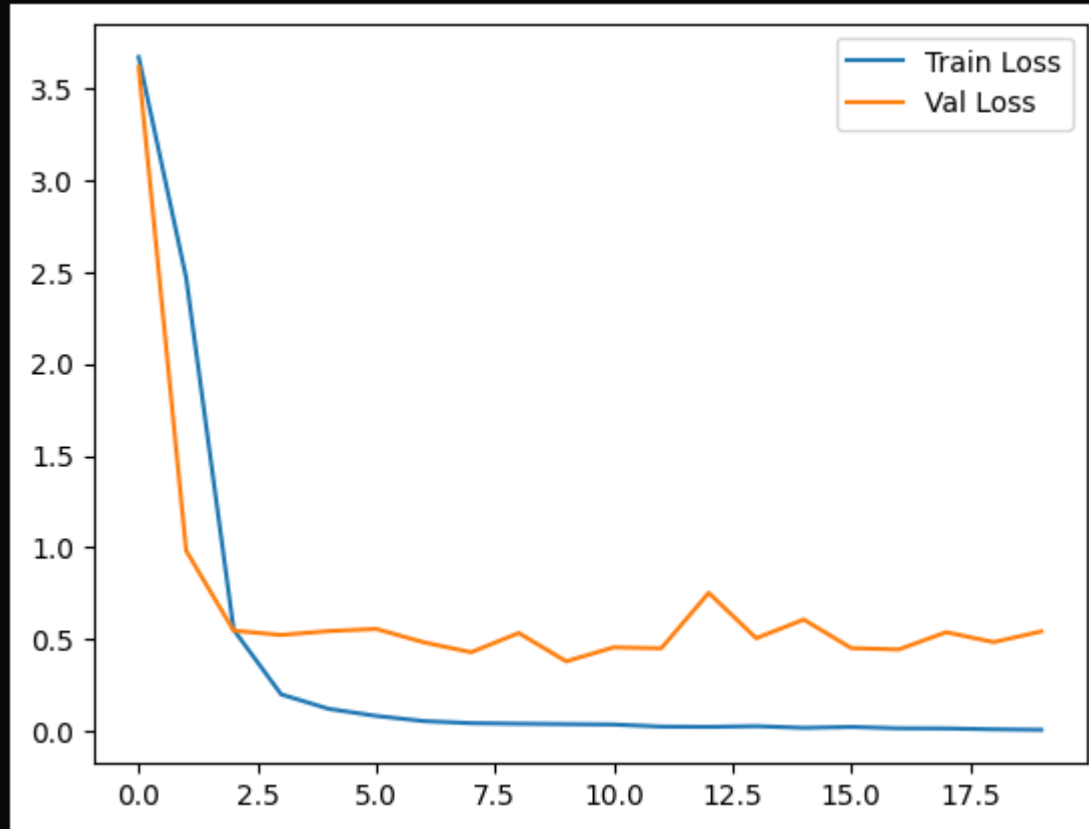
# Classification Model

Model	Optimiser	Train Acc	Train Loss	Val Acc	Val Loss	Fit time
ConvNext Base	AdamW	0.9811	0.0647	0.7831	1.1933	58m 29s
ConvNext Base	RMSProp	0.9760	0.0702	0.7766	1.3348	51m 11s
ConvNext Base	Adam	0.9761	0.0743	0.7724	1.1510	50m 41s
ConvNext Base	Adamax	0.9981	0.0330	0.8031	0.8546	1hr 12m 39s
ConvNext Base	SGD	0.9620	0.2507	0.7802	0.8217	12hr 17m 2s
ConvNext Base	Adadelata	0.9978	0.0091	0.9100	0.5438	76m 13s
ConvNext Base	Adagrad	0.0362	15.5349	0.0384	15.5020	52m 52s
ConvNext Base	Nadam	0.5144	1.6397	0.1886	4.7987	1hr 59m 7s
ConvNext Base	Ftrl	0.0619	3.6134	0.0553	3.6123	2hr 47m 16s

\* Other models such as EfficientNetV2 were tried as well, but the performance was unacceptable



# Classification Model (Adadelata Optimiser)



# Classification Model

99.78% Accuracy on train

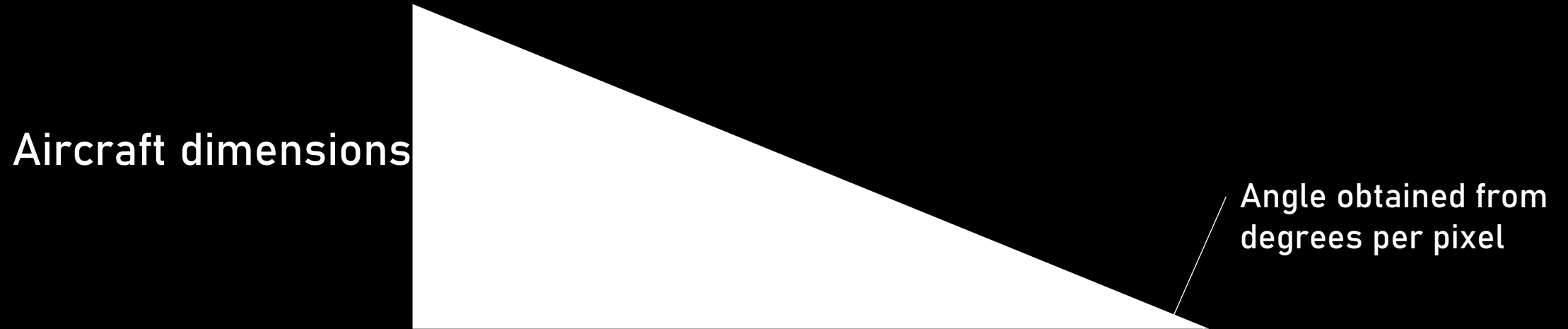
91.00% Accuracy on test

These performance metrics are good enough for the moment.

# Computing Location

- Functions were written to determine actual azimuth and elevation to aircraft, based on position in image frame.
- Next, the application of basic trigonometry.
- A database of aircraft containing wingspan and length was created.

# Trigonometric ratios - Distance




The length of the other two sides can be obtained using the trigonometric ratios.

The same principles can be applied to obtaining altitude and horizontal distance once we have the absolute distance and the true elevation from camera to aircraft.

# Bringing it all together

<https://www.youtube.com/watch?v=MklymKNswYQ>

**\*User interface for demo purposes only**  
– data should be automatically piped  
from sensors (both camera and gimbal)  
to the system



Open Image

Select an image file using the button above.

---

Enter the required information below:

Camera Latitude Degrees:	<input type="text"/>
Camera Latitude Minutes:	<input type="text"/>
Camera Latitude Seconds:	<input type="text"/>
Camera Latitude Hemisphere (N / S):	<input type="text"/>
Camera Longitude Degrees:	<input type="text"/>
Camera Longitude Minutes:	<input type="text"/>
Camera Longitude Seconds:	<input type="text"/>
Camera Longitude Hemisphere (E / W):	<input type="text"/>
Camera Elevation in Degrees:	<input type="text"/>
Camera Azimuth:	<input type="text"/>
Camera altitude in Feet:	<input type="text"/>

Process Image

# Results - Test Image



# Results - Produced by program



```
{  
  "Coordinates":  
    [13.592156214424612,  
     144.94771157856493],  
  "Altitude": 1103.6660373560921,  
  "Type": "F15"  
}
```

\*Aircraft set to fly at 1,000 ft

# Camera Location



Andersen AFB TS: 0.0m HDG: 247° (247° WSW) FTH: -18°  
ALT: 1850ft LOC: 11° 29' 31"N 144° 5' 04"E SINK: 0° 00' 00" 08:00:50  
F110L06/2500



# Camera Location





# Computed Aircraft Location



# Conclusion

The system works – the computed location is good enough to cue a track radar on target for weapons employment.

# Recommendations

Higher metrics would be preferable for greater confidence from users. Further development is therefore necessary.

# Recommendations

- Further training of both models on additional aircraft types (especially red air), including civilian airliners, so ODIN can be set to ignore them.
- Additional training on a much larger dataset – this can be obtained by grabbing frames from airshow videos.
- It may be worth overfitting the models, as there are only so many angles one can look at an aircraft from.

# Potential further development

- These will all add costs, but they are worth considering:
  - Train models on infrared images to support night-time operations and use infrared imaging hardware in addition to standard cameras.
  - (This adds an active element which could be detected by hostile parties) Add a laser rangefinder and cue it on target to get precise ranging information. The benefit of this is that the system can directly cue IR/Optically-guided munitions on target.

Questions?