Madras Institute of Technology
Anna University, Chromepet, Chennai- 600044

SCENE CLASSIFICATION AND TARGET DETECTION USING VISION TRANSFORMER ARCHITECTURE

PRESENTED BY:

M. VIJAYA PRAKASH (2022603026)
M.E. Avionics

UNDER THE GUIDANCE OF:

Dr. G. ANITHA
Professor

INTRODUCTION

- State-of-the-art approaches in quadrotor control split the problem into multiple cascaded subproblems, exploiting the different time scales of the rotational and translational dynamics. The work focuses on the mathematical model of a quadrotor's dynamics is derived, using Newton's and Euler's laws.
- Implantation of Vision Transformer model with self-attention algorithm will be performed for target object recognition as real-time mission objective.
- The implementation will be efficient enough to compute the full linearization solution for ViT model using a common ARM processor developer board.

OBJECTIVE FOR DESIGNING ViT BASED MODEL

- We propose an implementation of a Vision Transformer based model, which:
  1. collects the target object data and perform preprocessing for self-attention model;
  2. is trained with relative to the pre-trained model such as ImageNET;
  3. handles time-varying dynamic real time implementation.

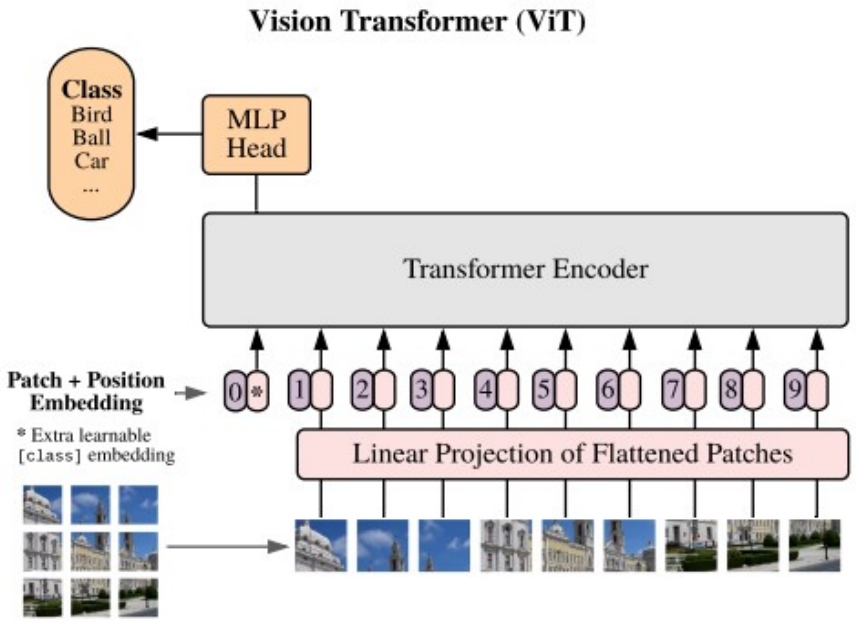
- Research Gap:
  1. Until today, Vision Transformer has often only been used to solve individual photos, but never to provide a time-varying dynamic real time image processing.
  2. Investigate the transfer learning capabilities of Vision Transformers. Explore whether pre-trained ViT models on outdoor datasets can be effectively fine-tuned for indoor navigation tasks, considering the domain gap between indoor and outdoor environments.

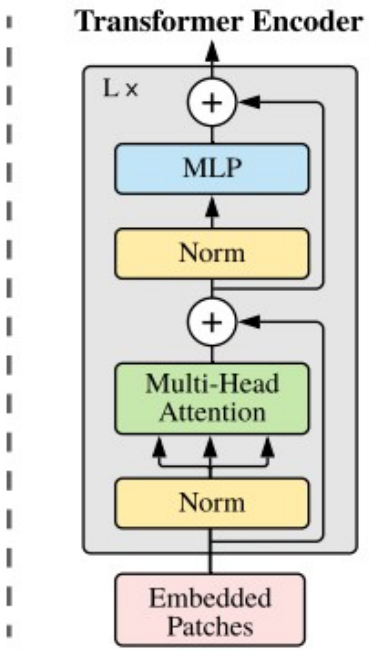
- Execution Workplan:
  1. Plan of execution will be developing ViT model for indoor location by collecting target dataset. It vastly simplifies the tuning of the training, since the tuning parameters are costs that directly related to the accuracy of the training model. To achieve this, we leverage the self-attention model to implement an independent self-learning vision transformer model.

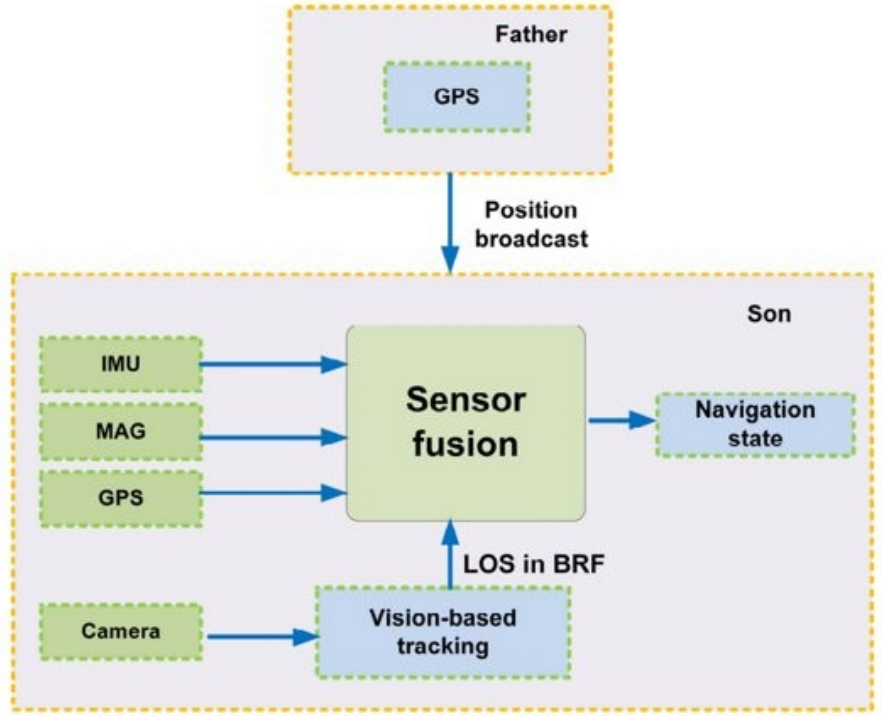
REFERENCE PAPERS

S.No	TITLE	LIST OF AUTHOR	DESCRIPTION
1	Vision based Position Control for MAVs using one single Artificial Landmark *(ETH Zurich)*	• Daniel Eberli • Davide Scaramuzza • Stephan Weiss • Roland Siegwart	This paper presents a real-time vision-based algorithm for 5 degrees of freedom pose estimation and set-point control for a Micro Aerial Vehicle (MAV). The camera is mounted on-board a quadrotor helicopter. Camera pose estimation is based on the appearance of two concentric circles which are used as landmark.
2	Recurrent Vision Transformers for Object Detection with Event Cameras *(Robotics and Perception Group, University of Zurich, Switzerland)*	• Mathias Gehrig • Davide Scaramuzza	Event cameras provide visual information with sub millisecond latency at a high-dynamic range and with strong robustness against motion blur. These unique properties offer great potential for low-latency object detection and tracking in time-critical scenarios.
3	An image is worth 16x16 words: Transformers for Image Recognition at scale *(Google Research, Brain Team)*	• Alexey Dosovitskiy • Lucas Beyer • Alexander Kolesnikov • Dirk Weissenborn • Xiaohua Zhai • Thomas Unterthiner • Mostafa Dehghani • Matthias Minderer • Georg Heigold • Sylvain Gelly • Jakob Uszkoreit • Neil Houlsby	While the Transformer architecture has become the de-facto standard for natural language processing tasks, its applications to computer vision remain limited. In vision, attention is either applied in conjunction with convolutional networks, or used to replace certain components of convolutional networks while keeping their overall structure in place. They show that this reliance on CNNs is not necessary, and a pure transformer applied directly to sequences of image patches can perform very well on image classification tasks. When pre-trained on large amounts of data and transferred to multiple mid-sized or small image recognition benchmarks (ImageNet, CIFAR-100, VTAB, etc.), Vision Transformer (ViT) attains excellent results compared to state-of-the-art convolutional networks while requiring substantially fewer computational resources to train.

S.No	TITLE	LIST OF AUTHOR	DESCRIPTION
4	Fixing the train-test resolution discrepancy *(Facebook AI Research)*	• Hugo Touvron, • Andrea Vedaldi, • Matthijs Douze, • Herve J ´ egou ´	Data-augmentation is key to the training of neural networks for image classification. This paper first shows that existing augmentations induce a significant discrepancy between the size of the objects seen by the classifier at train and test time: in fact, a lower train resolution improves the classification at test time. They then propose a simple strategy to optimize the classifier performance, that employs different train and test resolutions. It relies on a computationally cheap fine-tuning of the network at the test resolution. This enables training strong classifiers using small training images, and therefore significantly reduce the training time.
5	Event-based Vision: A Survey *(Robotics and Perception Group, University of Zurich, Switzerland)*	• Guillermo Gallego, • Tobi Delbruck, • Garrick Orchard, • Chiara Bartolozzi, • Brian Taba, • Andrea Censi, ¨ Stefan Leutenegger, • Andrew J. Davison, • Jorg Conradt, • Kostas Daniilidis, • Davide Scaramuzza	Event cameras are a type of bio-inspired sensor that differs from traditional cameras by asynchronously measuring per-pixel brightness changes, providing a stream of events encoding time, location, and sign of these changes. They offer high temporal resolution, very high dynamic range, low power consumption, and reduced motion blur, making them suitable for challenging scenarios in robotics and computer vision. This paper offers a comprehensive overview of event-based vision, covering the working principles, available sensors, and various applications. It discusses tasks ranging from low-level vision (feature detection, tracking, optic flow) to high-level vision (reconstruction, segmentation, recognition). The paper also explores processing techniques, including learning-based approaches and specialized processors like spiking neural networks. While highlighting the potential of event cameras, it acknowledges challenges and emphasizes opportunities for a more efficient, bio-inspired approach to machine perception and interaction with the world.





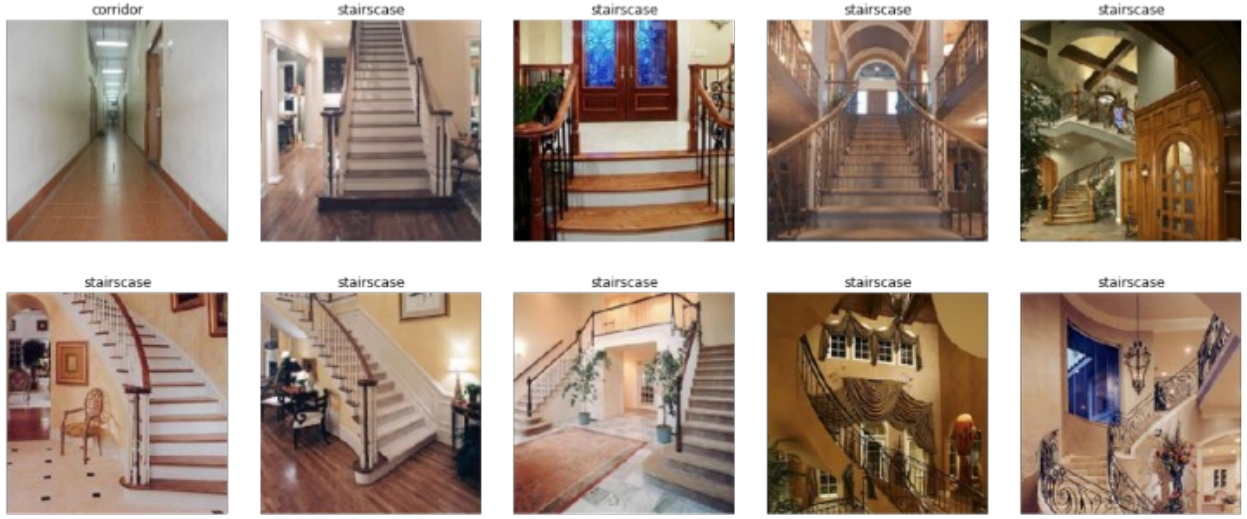


VISION TRANSFORMER

We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder.

In order to perform classification, we use the standard approach of adding an extra learnable “classification token” to the sequence.[1]

TRAINING DATASET COLLECTED



Train the Model

```
# Train the model
for epoch in range(EPOCHS):
  for step, (x, y) in enumerate(train_loader):
    # Change input array into list with each batch being one element
    x = np.split(np.squeeze(np.array(x)), BATCH_SIZE)
    # Remove unecessary dimension
    for index, array in enumerate(x):
      x[index] = np.squeeze(array)
    # Apply feature extractor, stack back into 1 tensor and then convert to tensor
    x = torch.tensor(np.stack(feature_extractor(x)['pixel_values'], axis=0))
    # Send to GPU if available
    x, y  = x.to(device), y.to(device)
    b_x = Variable(x)   # batch x (image)
    b_y = Variable(y)   # batch y (target)
    # Feed through model
    output, loss = model(b_x, None)
    # Calculate loss
    if loss is None:
      loss = loss_func(output, b_y)
      optimizer.zero_grad()
      loss.backward()
      optimizer.step()

    if step % 50 == 0:
      # Get the next batch for testing purposes
      test = next(iter(test_loader))
      test_x = test[0]
      # Reshape and get feature matrices as needed
      test_x = np.split(np.squeeze(np.array(test_x)), BATCH_SIZE)
      for index, array in enumerate(test_x):
        test_x[index] = np.squeeze(array)
      test_x = torch.tensor(np.stack(feature_extractor(test_x)['pixel_values'], axis=0))
      # Send to appropirate computing device
      test_x = test_x.to(device)
      test_y = test[1].to(device)
      # Get output (+ respective class) and compare to target
      test_output, loss = model(test_x, test_y)
      test_output = test_output.argmax(1)
      # Calculate Accuracy
      accuracy = (test_output == test_y).sum().item() / BATCH_SIZE
      print('Epoch: ', epoch, '| train loss: %.4f' % loss, '| test accuracy: %.2f' % accuracy)
```

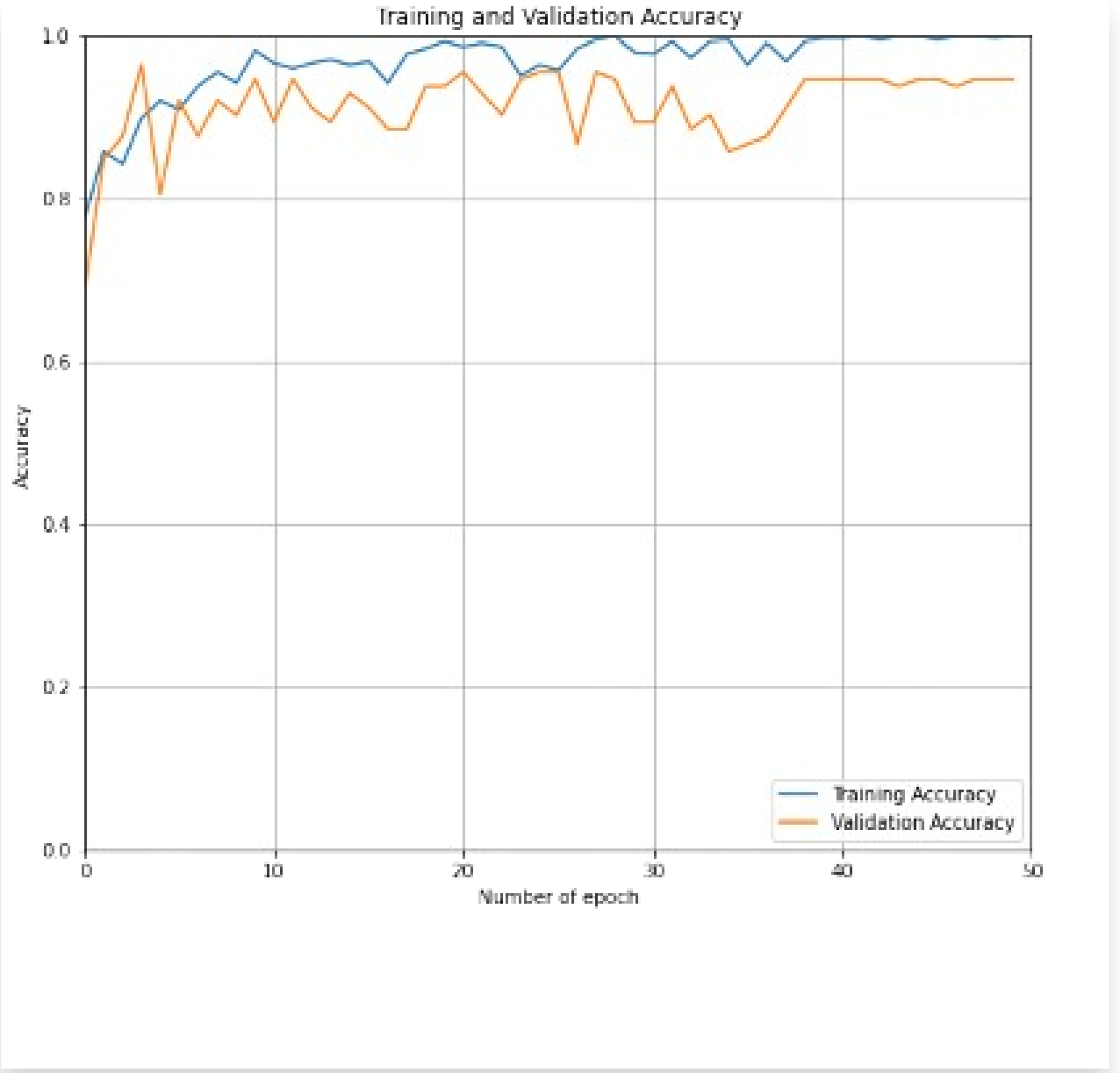
Testing the Model

```
import matplotlib.pyplot as plt
import numpy as np

EVAL_BATCH = 1
eval_loader  = data.DataLoader(valid_ds, batch_size=EVAL_BATCH, shuffle=True, num_workers=4)
# Disable grad
with torch.no_grad():

  inputs, target = next(iter(eval_loader))
  # Reshape and get feature matrices as needed
  print(inputs.shape)
  inputs = inputs[0].permute(1, 2, 0)
  # Save original Input
  originalInput = inputs
  for index, array in enumerate(inputs):
    inputs[index] = np.squeeze(array)
  inputs = torch.tensor(np.stack(feature_extractor(inputs)['pixel_values'], axis=0))

  # Send to appropriate computing device
  inputs = inputs.to(device)
  target = target.to(device)

  # Generate prediction
  prediction, loss = model(inputs, target)

  # Predicted class value using argmax
  predicted_class = np.argmax(prediction.cpu())
  value_predicted = list(valid_ds.class_to_idx.keys())[list(valid_ds.class_to_idx.values()).index(predicted_class)]
  value_target = list(valid_ds.class_to_idx.keys())[list(valid_ds.class_to_idx.values()).index(target)]

  # Show result
  plt.imshow(originalInput)
  plt.xlim(224,0)
  plt.ylim(224,0)
  plt.title(f'Prediction: {value_predicted} - Actual target: {value_target}')
  plt.show()
```

Note: The Vision Transformer model can be used as a regular PyTorch code.

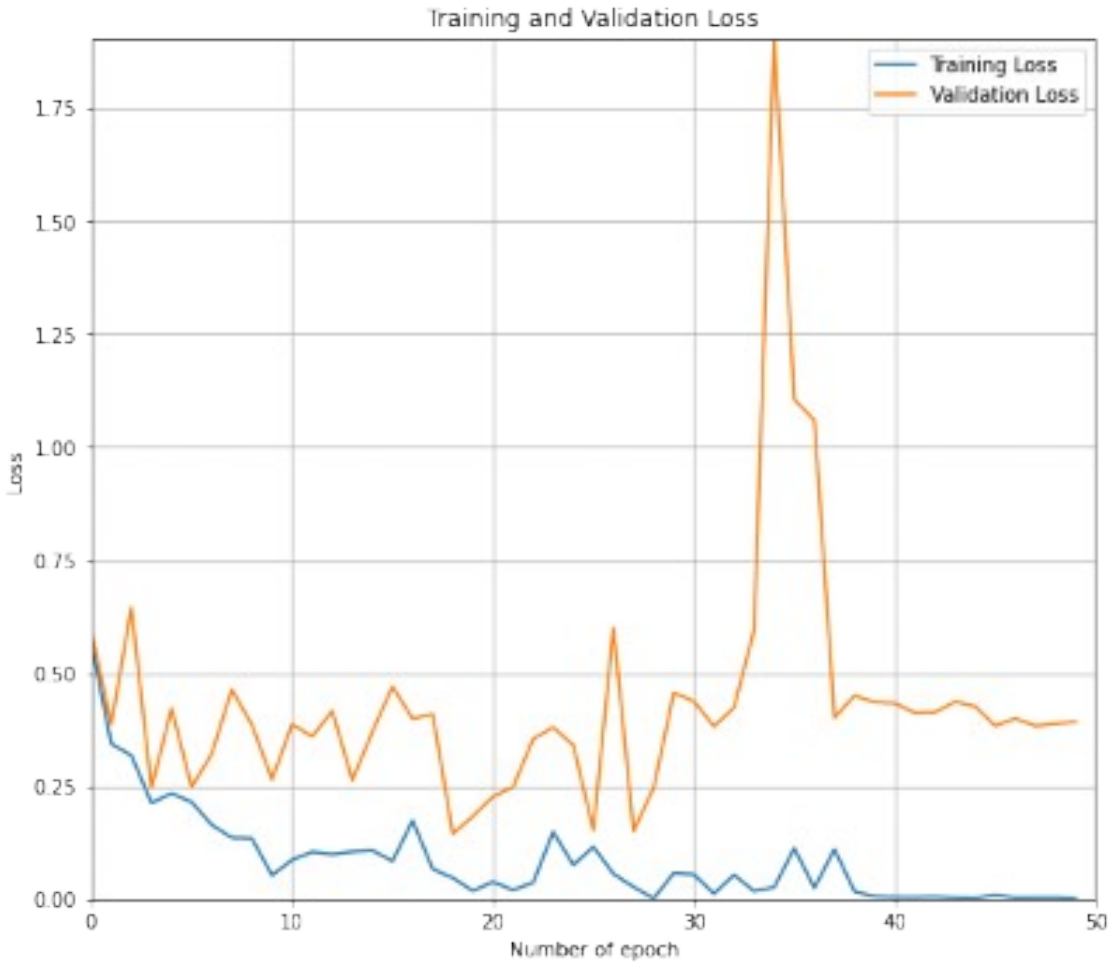
Number of Epoch vs Accuracy

- The Model is trained on training dataset of 480 images for 50 epochs and validated using validation set of 120 images.
- The accuracy and loss function for both training and validation set is plotted for the number of epochs. The model has converged after 40 epochs. Since the training accuracy had not increased further after that.



Number of Epoch vs Loss

From we can see that as the number of epoch increases the validation loss starts increases (between 30-40). This shows that the model needs to be trained with a bigger dataset. The validation loss can be reduced in the future by training this model with different datasets.



Evaluating Metrics

- For predictions, there are four important terms: - True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). TP and TN represent the cases when the model correctly predicts corridor and staircase correctly, whereas FP and FN are the cases when the opposite results are obtained.
- The Precision metric shows what percent of predictions are correct.
- Recall describes what percent of positives are correctly identified.
- The F-beta score is the percent of positive predictions that are correct.

		Actual	
		A	**B**
Predicted	**A**	True A (TP)	False A, Actually B (FP)
	B	False B, Actually A (FN)	True B (FP)

$$Accuracy = \frac{Number\ of\ correct\ predictions\ (TP+TN)}{Total\ Number\ of\ predictions\ (TP+TN+FP+FN)}$$

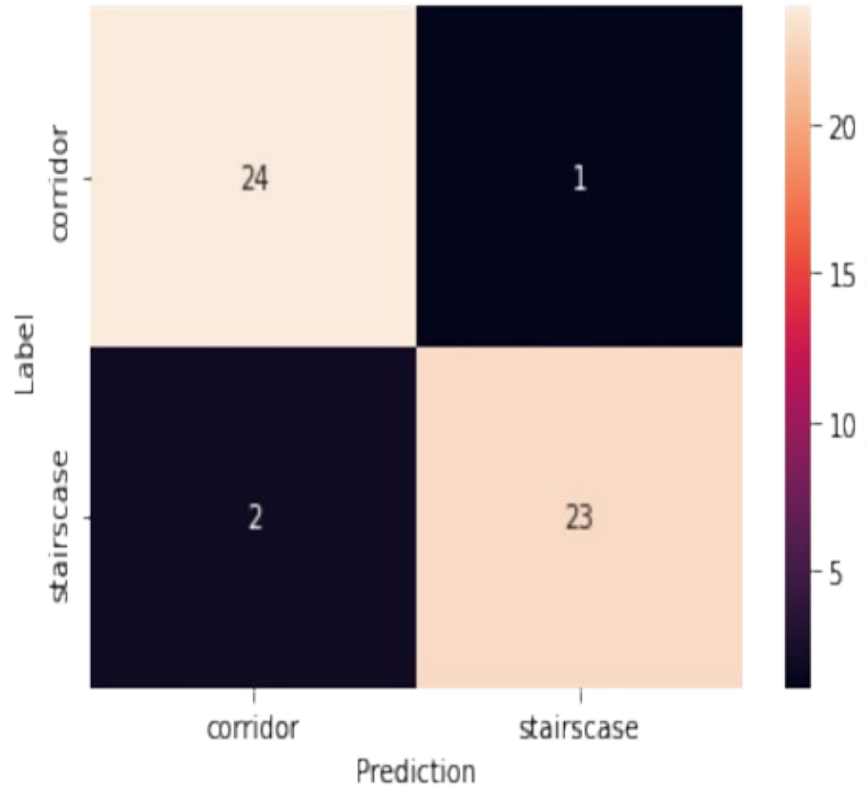
$$Precision = \frac{TP}{TP+FP}$$

$$Fbeta_{Measure} = \frac{(1+beta^2)*Precision*Recall}{beta^2*Precision+Recall}$$

$$Recall = \frac{TP}{TP+FN}$$

Confusion Matrix and Model Score

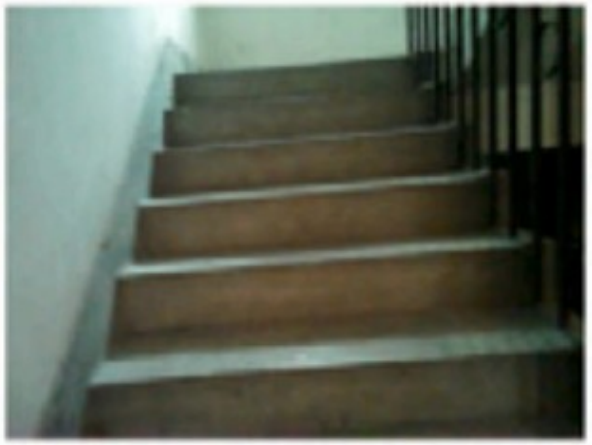
- Table shows the score of the model in various metrics.
- The precision for the corridor class is the number of correctly predicted corridor photos (24) out of all predicted corridor photos (24+1=25), which amounts to 24/25=96%. So, 96% of the photos that our model classifies as corridor are actually corridor.
- On the other hand, the recall for corridor is the number of correctly predicted corridor photos (24) out of the number of actual corridor photos (24+2=26), which is 24/26=92.3%.
- This means that our classifier classified 92.3% of the corridor photos as corridor. The obtained accuracy for corridor is 96% and for staircase is 92% and the overall accuracy for model is 94%.
- The accuracy can be improved in future with a greater number of training images of varying dimensions.



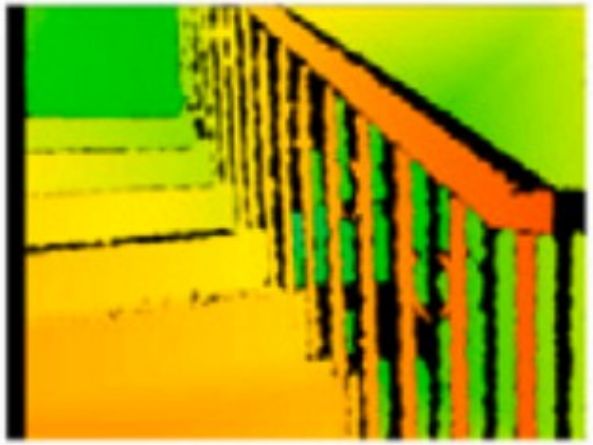
	Precision	*Recall*	*F1 Score*	*F2 Score*	*F0.5 Score*
Corridor	0.96	0.923	0.941	0.93	0.952
Staircase	0.92	0.958	0.939	0.95	0.927

ViT STAIRCASE DETECTION

Captured Image



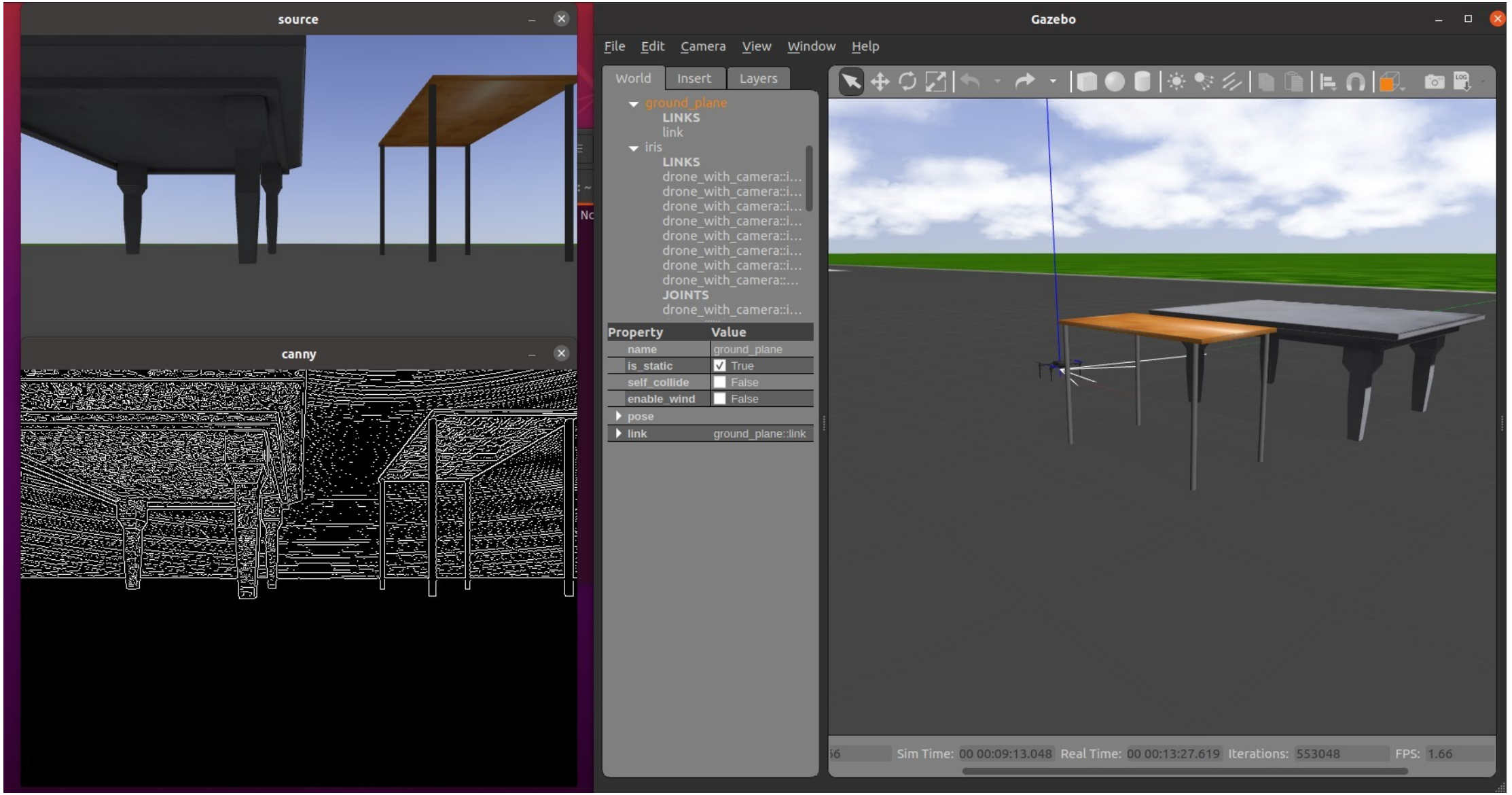
Segmented Image



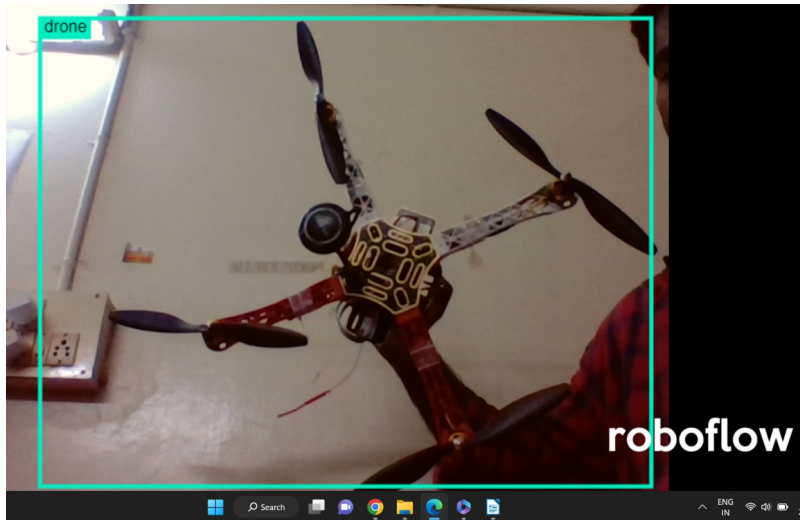
Saliency Map Generation [3]

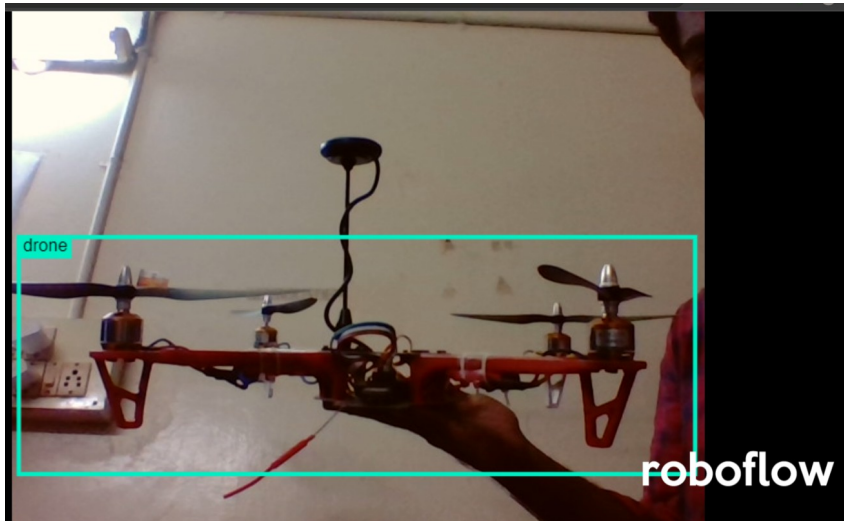
Saliency map generation techniques are at the forefront of explainable AI literature for a broad range of machine learning applications. It aims at resizing an image by expanding or shrinking the noninformative regions. Therefore, retargeting algorithms rely on the availability of saliency maps that accurately estimate all the salient image details.[3]

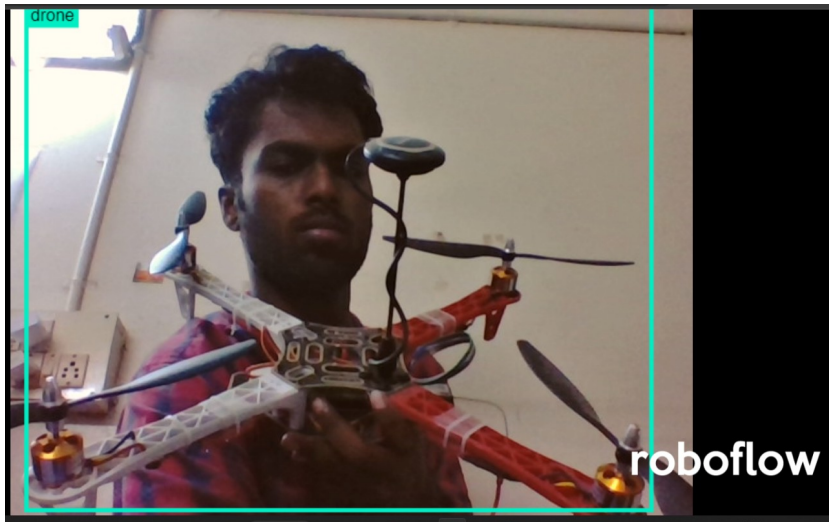
COMPUTR VISION IN GAZEBO SIMULATION

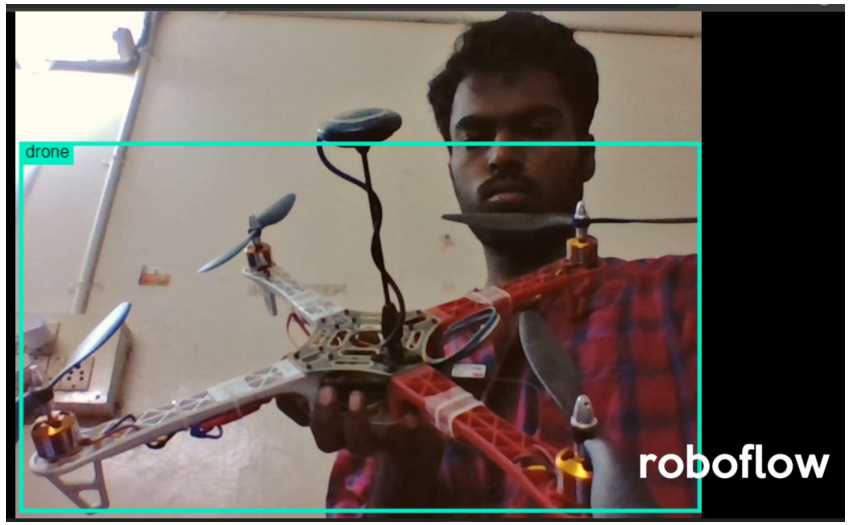


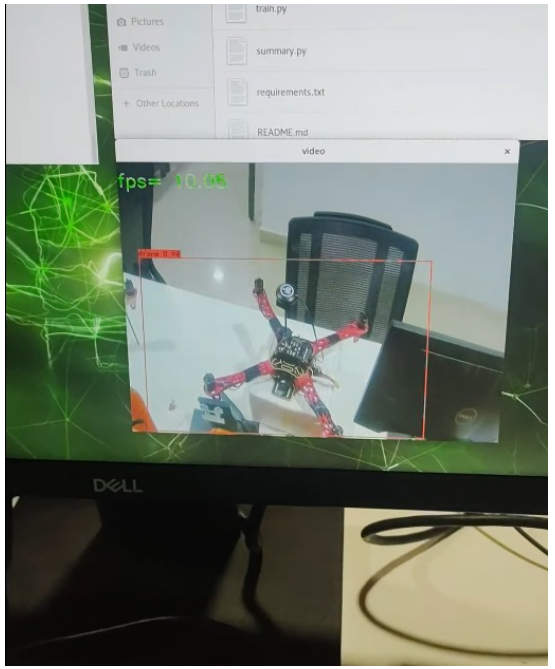
DRONE DETECTION USING CNN ARCHITECTURE

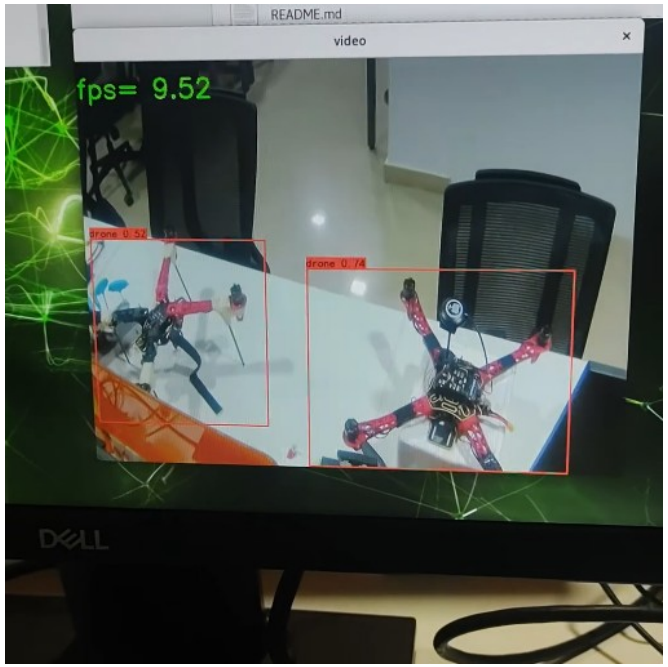


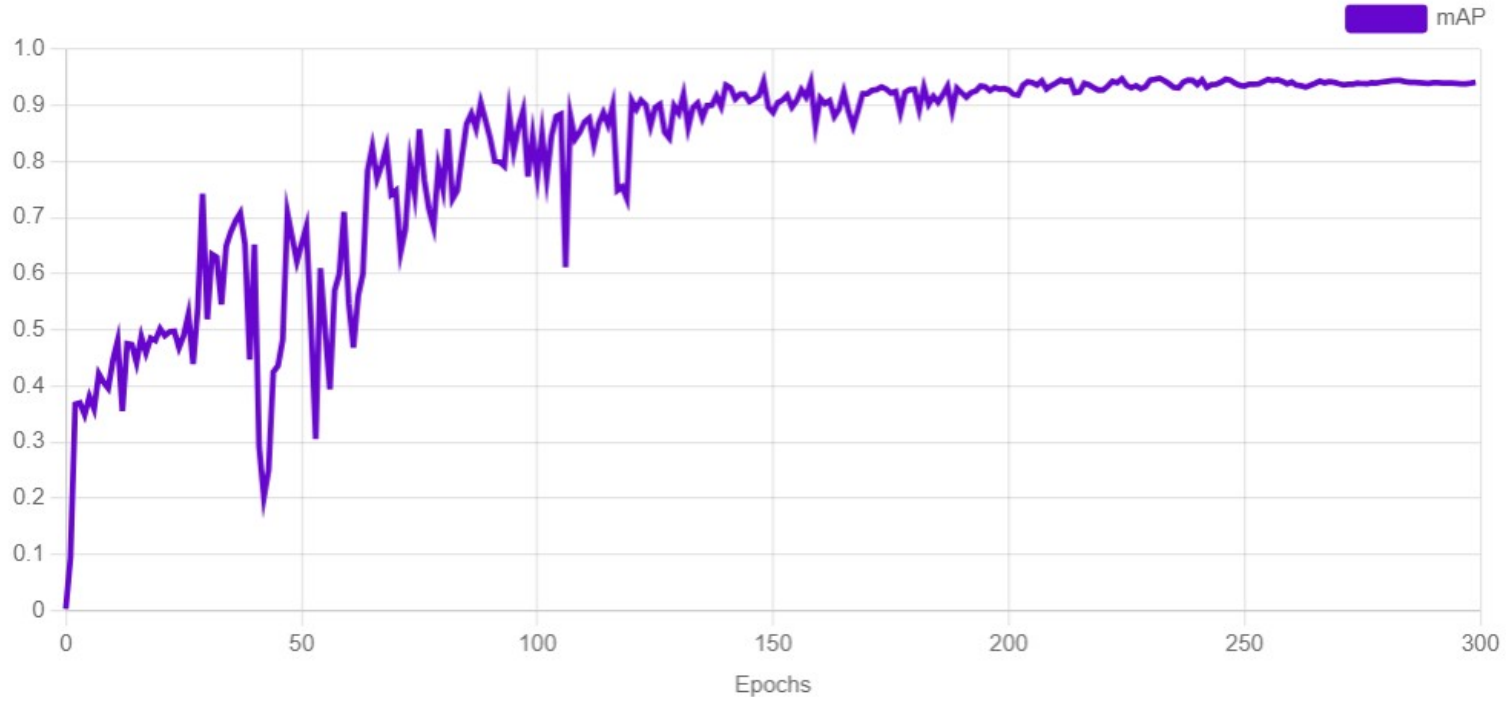


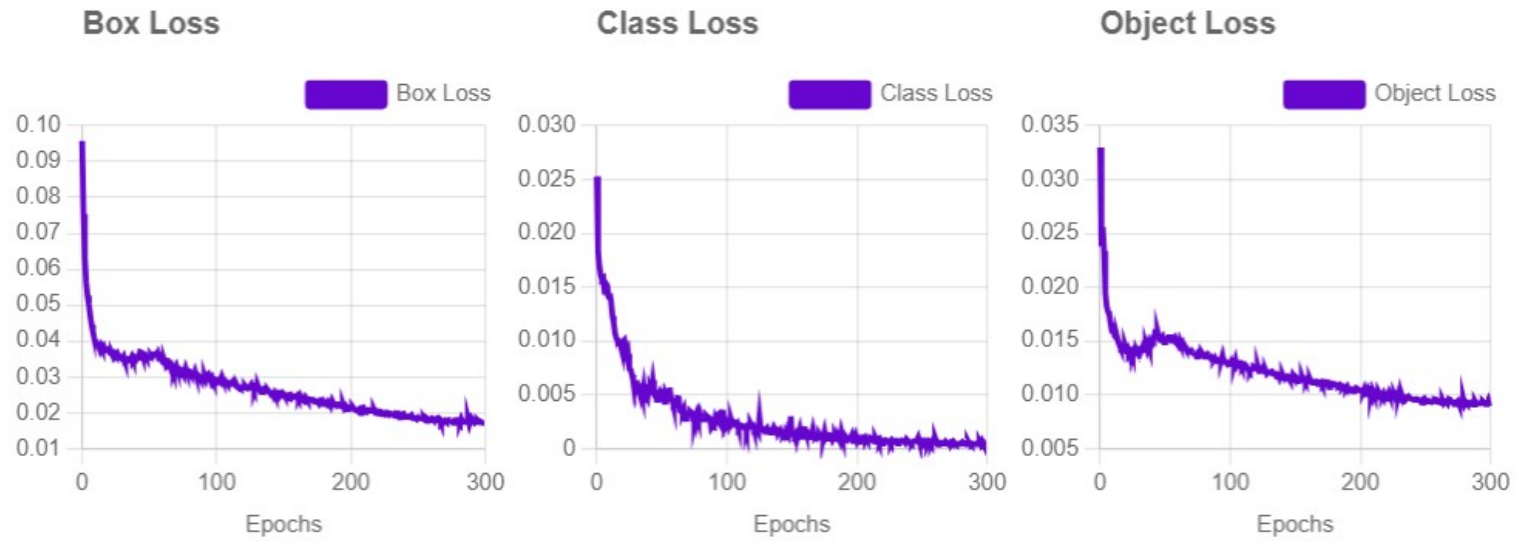












AFTER INCREASING DATASETS

- The Model is trained on training dataset of 3550 images for 300 epochs and validated using validation set of 1065 images.
- The accuracy and loss function for both training and validation set is plotted for the number of epochs. The model has converged after 130 epochs. Since the training accuracy had significantly increased
after previous training

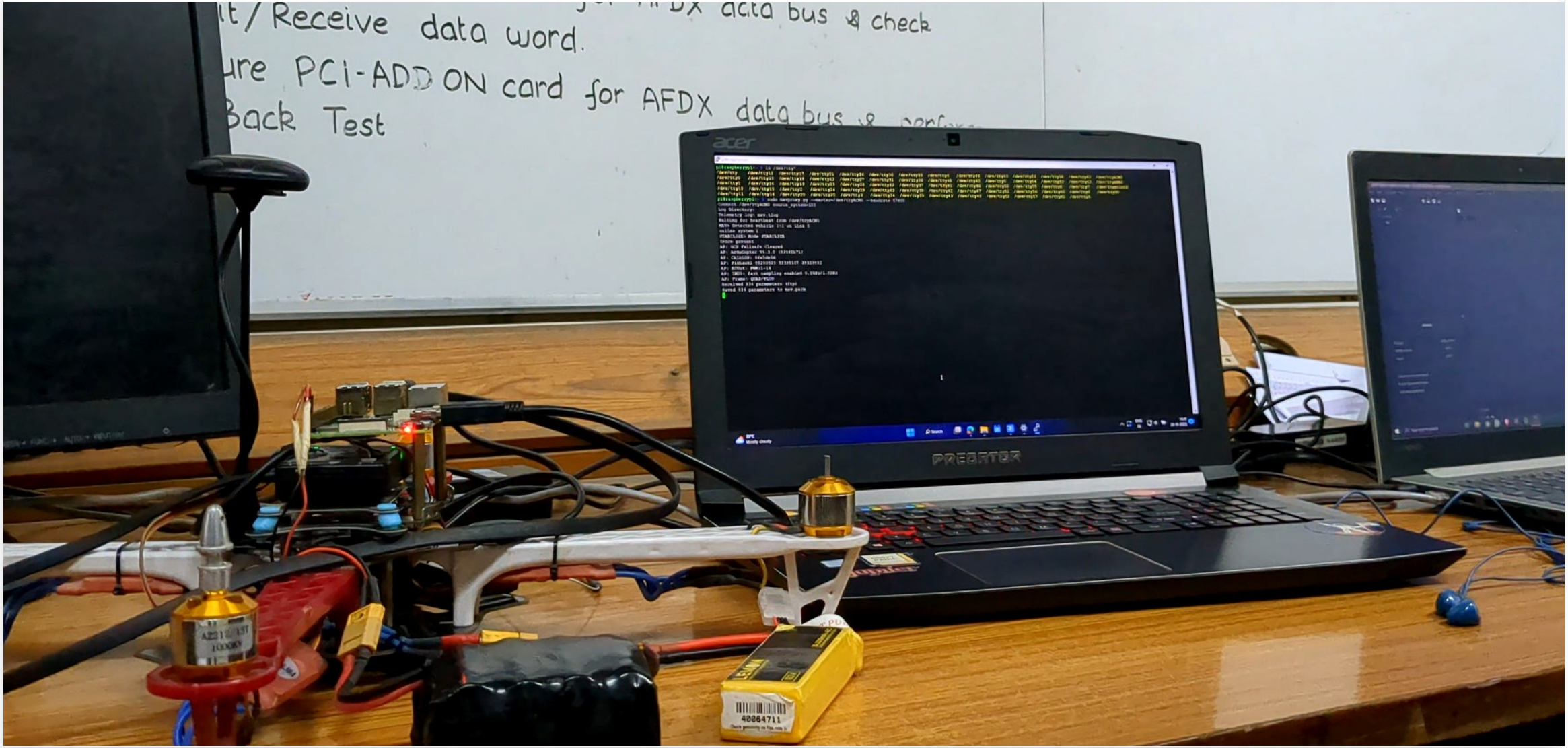
REAL-TIME DRONE DETECTION USING NLP ARCHITECTURE

CNN vs ViT

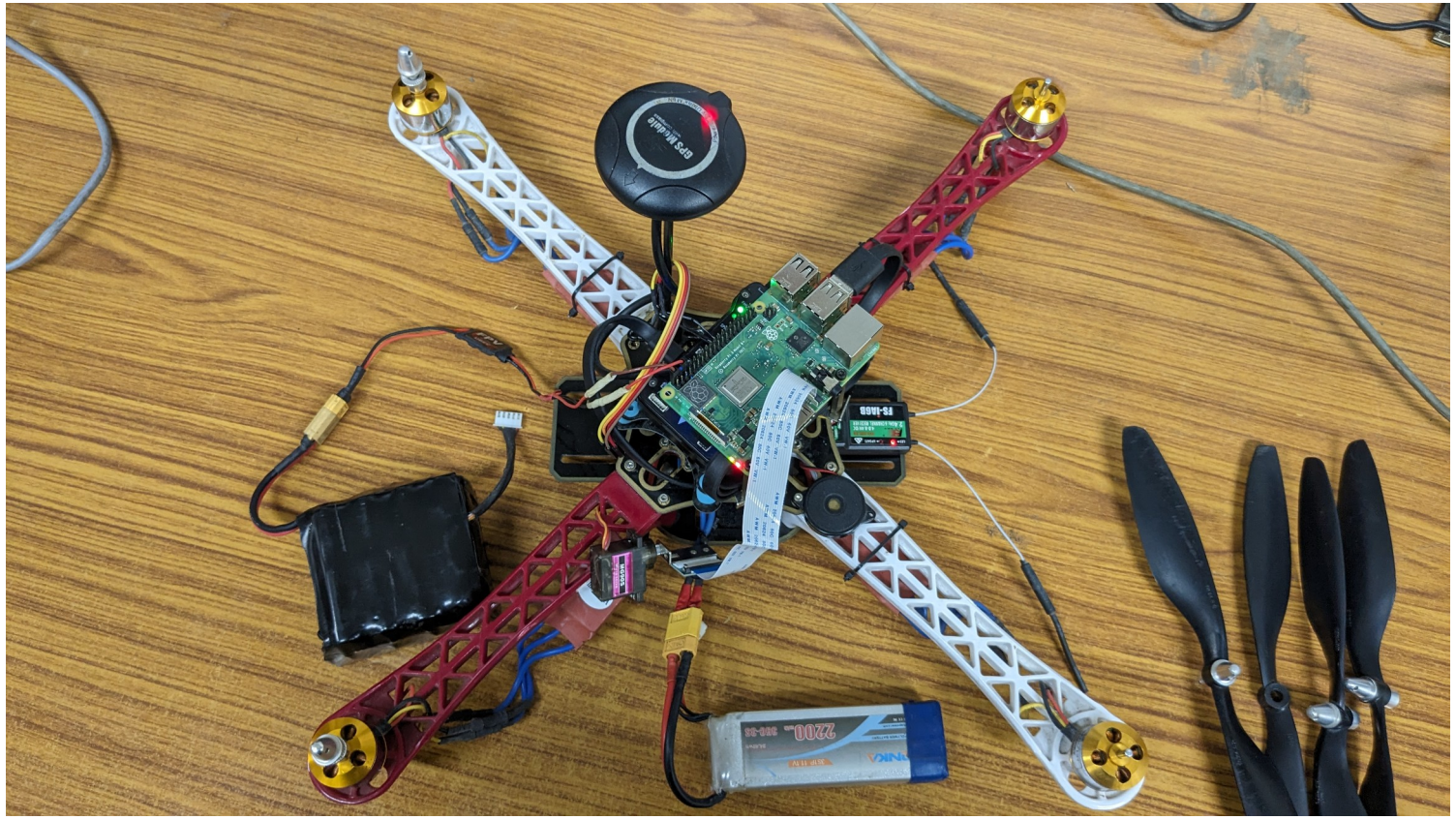
	CNN	VISION TRANSFORMER
ARCHITECTURE	Multiple Layer	Transformer
INPUT REPRESENTATION	Images as Grid	Images as Patches
LOCAL VS GLOBAL INFO	Local Spatial	Self-Attention
PARAMETER EFFICIENCY	More Efficient	Parameter Heavy
TRAINING DATA	Smaller Scale	Larger Scale

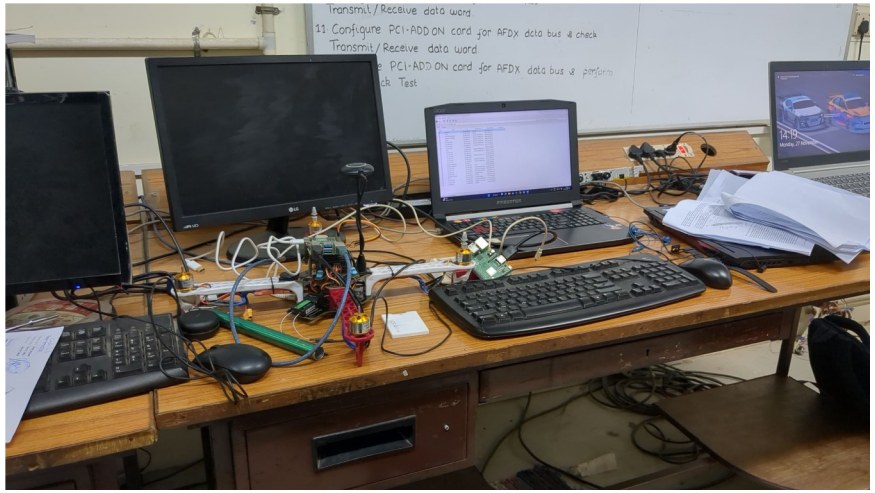
MAVLINK

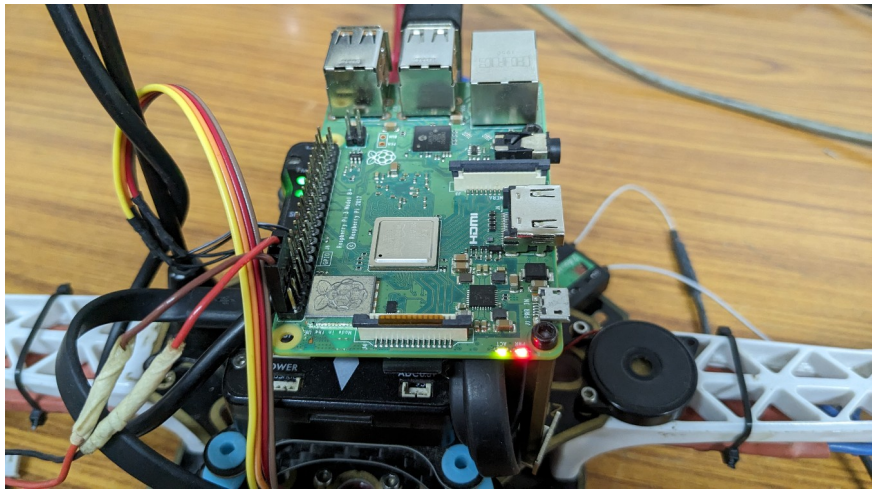
- The Mavlink message is basically a stream of bytes that has been encoded by GCS (Ground Station Control) and is sent to the flight controller via USB serial OR Telemetry (both cannot be used at the same time. If they are plugged in at the same time, USB is given preference and Telemetry is ignored).
- message length = 17 bits (**6 bytes header** + **9 bytes payload** + **2 bytes checksum**)
- My keen interests are - **System ID** *(source of message)*, **Component ID** *(subsystem within the system)*, **Message ID** *(What is this message about)*, **Payload** *(the actual data)*
- The payload data is extracted from the message and put into a packet. A packet is a data structure based on a ‘type of information’.
- The packet is put into an ‘appropriate data structure’. There are plenty of data structures e.g. for Attitude (pitch, roll, yaw orientation), GPS, RC channels, etc. that groups similar things together, to be it modular, understandable. These data structures are ‘perfectly 100% alike’ at the sending and receiving ends (i.e. at GCS and Copter side).


t/Receive data word.
ure PCi-ADD ON card for AFDX data bus
Back Test
acer
PREDATOR

HARDWARE SETUP



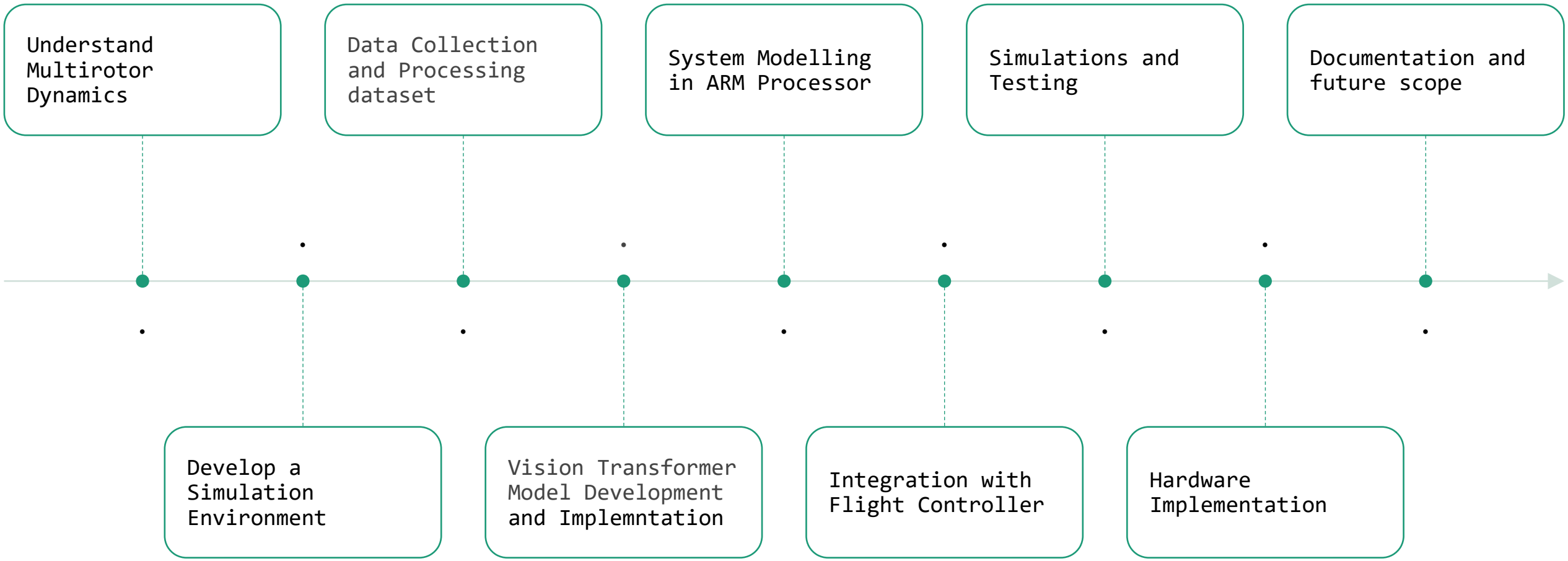




WORKFLOW

- Literature Survey
- Data Collection and Processing dataset
- Vision Transformer Model Development
- Implementation on SELF-Attention algorithm
- Real-time comparison with CNN architecture (CNN vs NLP)
- Interpretation of Results
- Documentation

WORKFLOW


Understand Multirotor Dynamics
Develop a Simulation Environment
Data Collection and Processing dataset
Vision Transformer Model Development and Implemntation
System Modelling in ARM Processor
Integration with Flight Controller
Simulations and Testing
Hardware Implementation
Documentation and future scope

REFERENCE

1. Tutorial 6: Transformers and Multi-Head Attention — UvA DL Notebooks v1.2 documentation (uvadlc-notebooks.readthedocs.io)
2. How to Train the Hugging Face Vision Transformer On a Custom Dataset (roboflow.com)
3. Visual Saliency Transformer
4. Salient Object Detection: A Discriminative Regional Feature Integration Approach
5. Indoor versus outdoor scene recognition for navigation of a micro aerial vehicle using spatial color gist wavelet descriptors

THANK YOU