I chose Faster RCNN with a ResNet50 backbone and a Feature Pyramid Network. I picked this model because it has a good balance between power and stability. It’s deep enough to learn small details, like differences between cats and dogs, but not so complex that it becomes hard to manage on a small dataset.

Another reason I liked this model is that it's easier to understand and debug. You can see how it makes decisions, where it finds objects, and where it makes mistakes. Since I wasn’t focused on running the model in real time, I wanted something that would help me study the model's behavior and learn from it and Faster RCNN was a great fit for that.

There were a lot of challenges faced while working on this project, at first, the validation accuracy looked perfect. But when I tested the model on new images, it didn’t perform well. Sometimes the boxes were in the wrong place, or they didn’t show up at all. The model also gave very low confidence scores even when the prediction was correct. This showed that I couldn’t trust the validation results alone. I had to actually visualize the predictions to see the real issues.
Another challenge was working with Albumentations. It’s a great tool, but if the image and the bounding boxes aren’t transformed exactly the same way, everything breaks. For example, at one point my boxes were floating in random parts of the image because the flip was applied to the image but not the boxes.
The dataset also had some problems. Some bounding boxes were very tight around the object, others were too loose. Even label names like "Cat" instead of "cat" caused confusion.
Lastly, normalization helped during training, but it made the images look strange. I had to undo the normalization to visualize the results properly, which added an extra step and made debugging harder.

To improve my work, the first thing I’d do is simply train longer and with more data. I only used around 250 images and trained for 2 epochs, which isn’t really enough to get solid results but that’s mainly because I’m limited by hardware. Training on CPU is super slow, and running more epochs wasn’t practical. If I had access to a GPU or more time, I’d definitely increase both the dataset size and training duration.

Another thing I noticed is that the model sometimes gave very low confidence scores, even when it was clearly correct. I think with more data and better training, this would improve on its own, but if not, I might look into techniques like label smoothing or confidence calibration to help with that.

I’d also like to add some realistic noise to the training images, like slight cropping or changing brightness, to help the model learn how to handle imperfect input. Right now, I think it’s learning too “clean” and overfitting to what it sees.

Lastly, I’m open to trying something different like YOLOv8 in the future maybe freeze most of the layers and just fine-tune the head on my dataset. But honestly, I’d only explore that if I can improve my hardware setup first.