

# REPORT

The training was conducted entirely on Google Colab's free tier using a **T4 GPU** which had **15 GB's of usable VRAM** and limited session durations. These constraints definitely imposed restrictions on the batch sizes, model size and training time. Therefore the way around was to use a smaller model and few batch sizes, also usage of other training methods like mixed precision training and gradient checkpointing.

Training was set up for **30 epochs** with early stopping enabled, so in-reality it ran for only **21 epochs**. We didn't have to write the training loop or evaluation methods from scratch as Roboflow took care of everything. So we had to only set up the parameters we required and just run. The various parameters that were used were:

**epochs=30**

**batch\_size=2**

**grad\_accum\_steps=8**

**lr=1e-4**

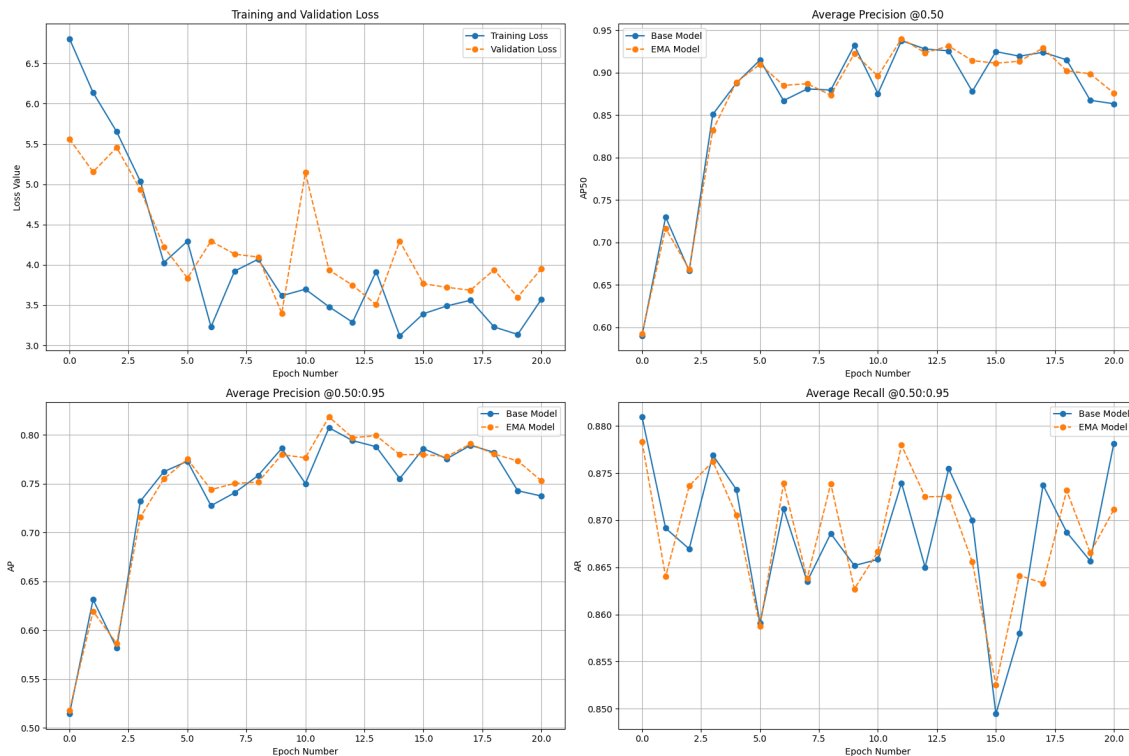
**early\_stopping=True**

**early\_stopping\_patience=9**

**use\_ema=True**

**fp16=True**

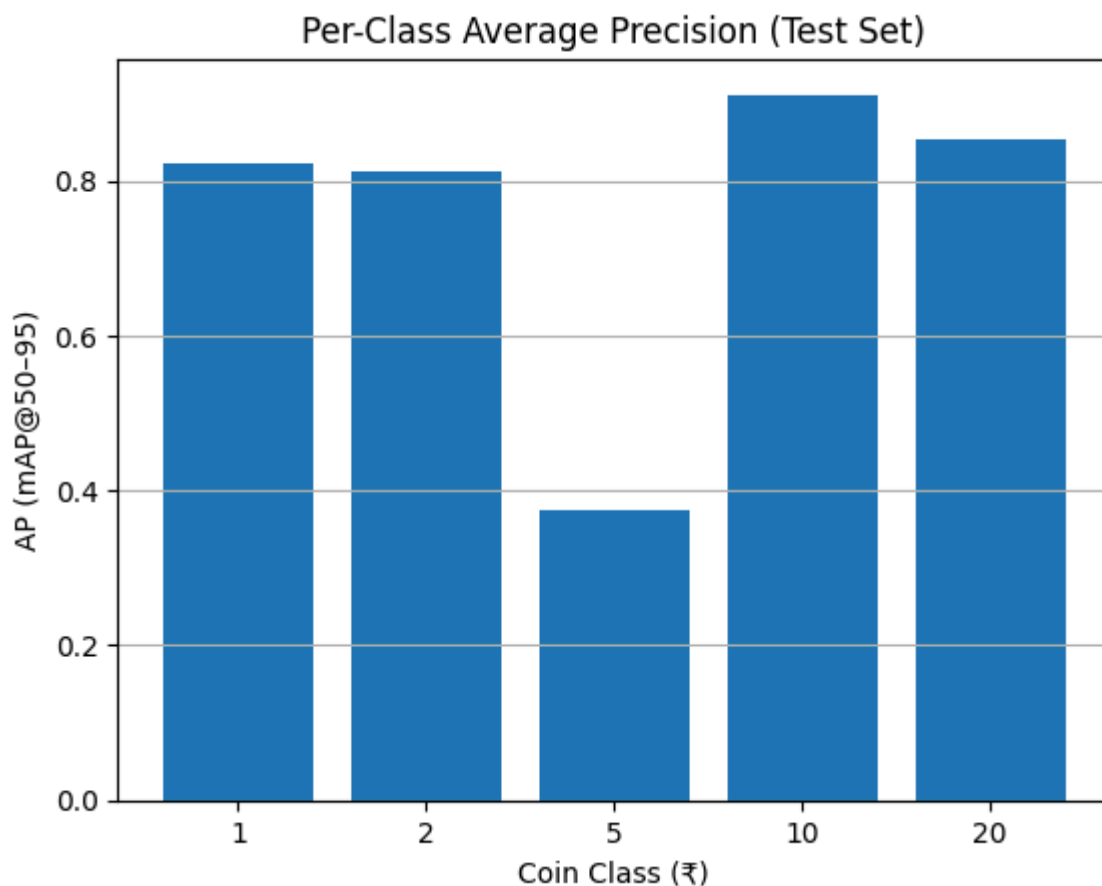
**checkpoint\_interval=5**



## PERFORMANCE ANALYSIS

The model achieved a strong overall performance. On the test set the model reached approximately **0.89 mAP@50** and **0.75 mAP@50–95**.

1. **Classes 1, 10 and 20** performed the **best** with near-perfect precision and recall on the test set.
2. **Class 2 had moderate performance**, while the detection was good but it had occasional confusions.
3. **Class 5 performed the worst** with perfect recall but zero-precision on the test set. This indicated that while the model was able to reliably detect the presence of coins, it frequently misclassified other classes as 5



## FAILURE CASES AND LIMITATIONS

The main reason for failures was class confusion i.e the model was correctly able to detect the coins but the predictions on classes were wrong. The reason can be that the images were not clear enough so it confused some numbers like 2 and 20. Also the presence of a supercategory ('coins') which added an extra index thereby increasing the classification complexity.

## POTENTIAL IMPROVEMENTS

1. Training at a higher resolutions
2. Using larger models
3. Collecting more balanced and high-quality examples, particularly for 5 rupee coins

## LESSONS LEARNED

Even with resource constraints it's possible to find a way around and actually get good performance, so *it's what you use it's about how you use it*.