Experiments

Comparisons and motivations behind the main design choices. Includes their training plots and respective hyperparameters and metrics.
Test results at the bottom

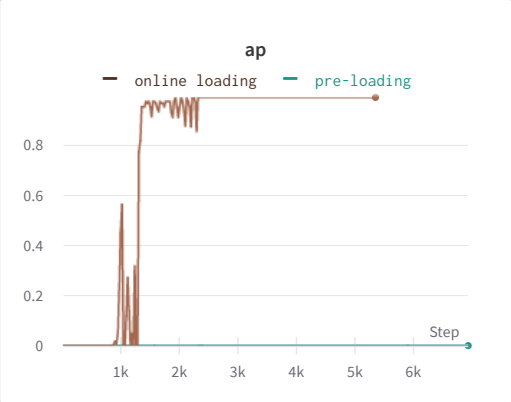
Sri Datta Budaraju

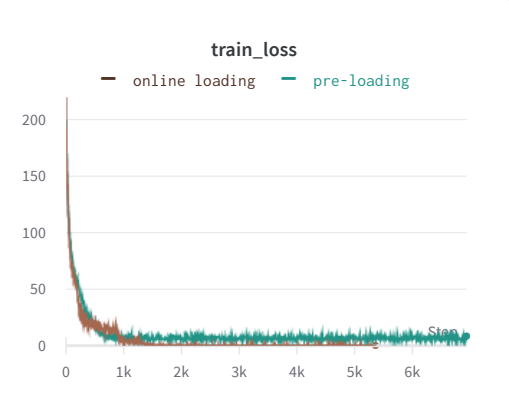
Loading procedure

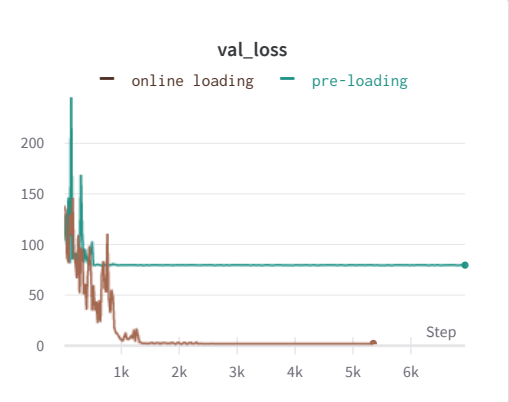
How different is taking 5k/1k train/Val images all at once to train from generating them for every epoch?

- It is important to note that the validation changes as well. Since the dataset's function to retrieve a sample calls the image generator

Since there is no pattern in the scale or orientation other than the shape of the object as we would find in real datasets, it might be hard to generalize to unseen scale and poses.
This might be the reason that the training loss is not significantly different but the validation loss is.







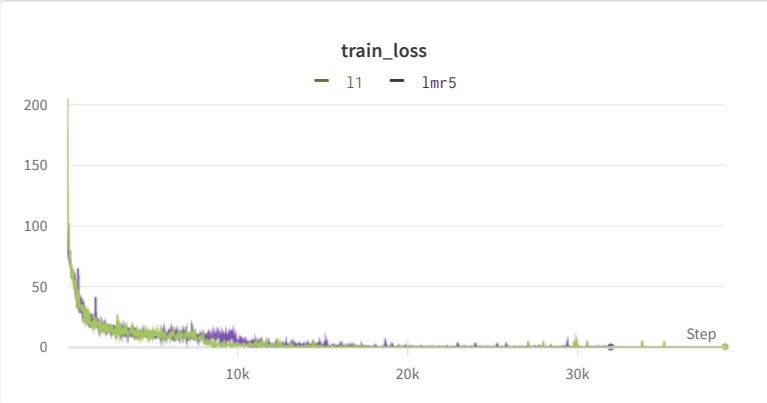
Loss Function

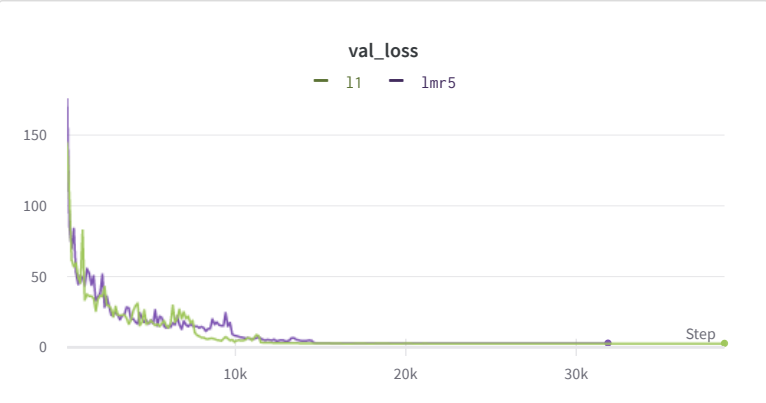
The common loss function used in rotated object detection papers is l1 loss. Modulated Rotation Loss (LMR) is a loss function that tries to handle the discontinuity in the loss due to the sudden and periodic shifts in angles, hight, and width.

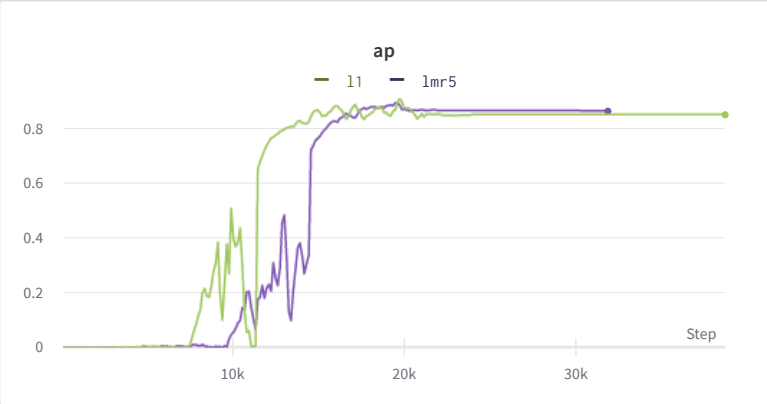
The training parameters are mentioned below.

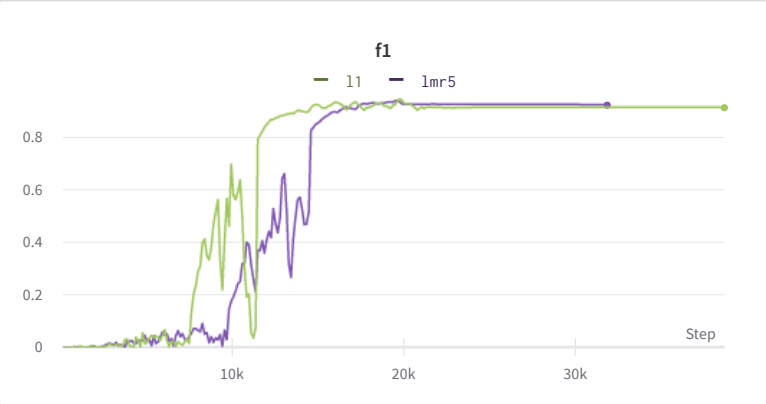
There is a notable improvement in the evaluation metrics using LMR. We use 5 parameters for loss as given in the baseline. The LMR papers present considerable improvements using 8 parameters

Paper - Learning Modulated Loss for Rotated Object Detection [https://arxiv.org/pdf/1911.08299.pdf]









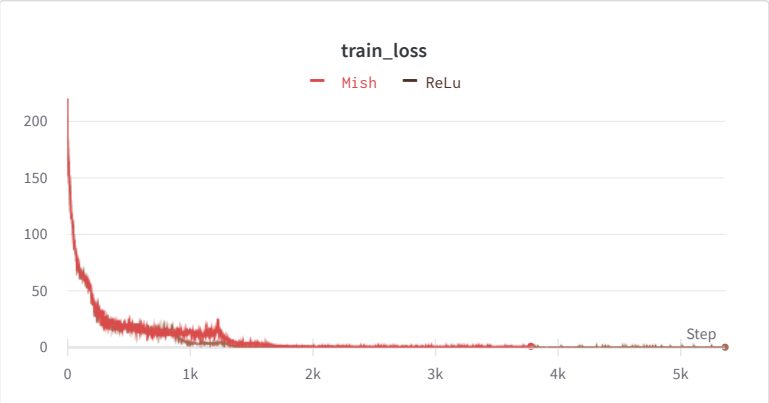
Run set 2

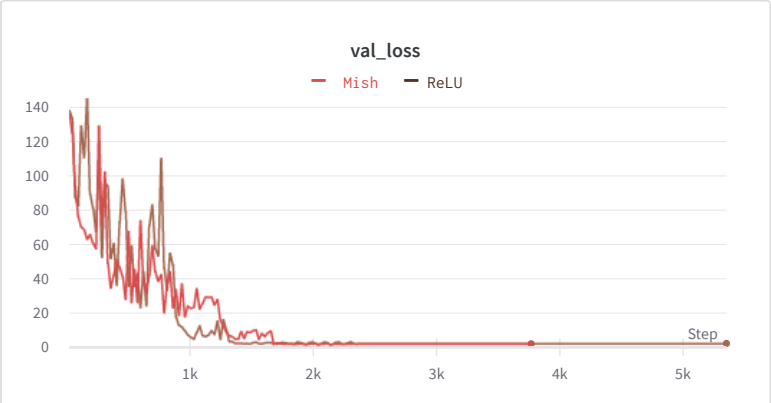
Name (2 visualized)	batch_size	train_len	val_len	epoch	train_loss	val_loss	ap	f1
l1	256	32000	8000	300	5.61	2.837	0.85	0.9131
lmr5	256	32000	8000	248	0.3673	2.993	0.863	0.9227

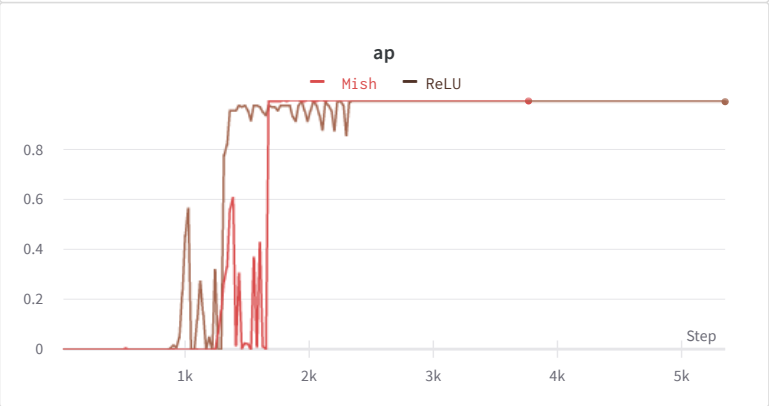
1-2 of 2

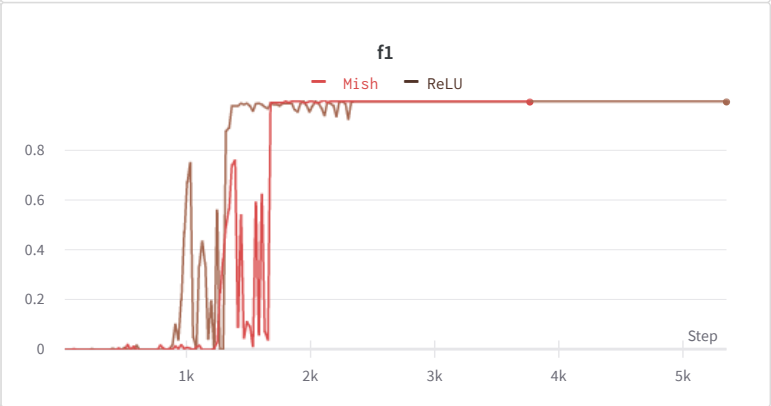
Activation Function

experimented to see if the activation function could make some difference and tried Mish as it seems to do better than ReLU or Swish on CV tasks. And it did to some extent. So Mish is used for the other experiments as well.









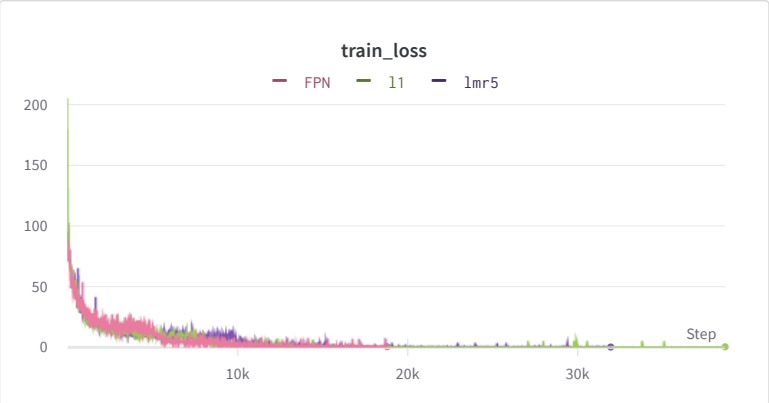
Run set 2

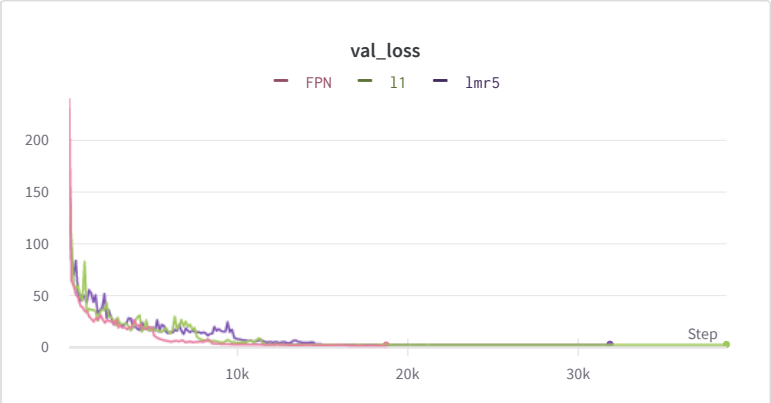
Name (2 visualized)	epoch	ap	epoch	train_len	val_len	f1	prec	train_loss	val_loss
Mish	158	0.9946	157	5000	1000	0.9945	0.9938	0.1501	2.05
online loading	224	0.9917	223	5000	1000	0.9949	0.9947	0.3892	2.249

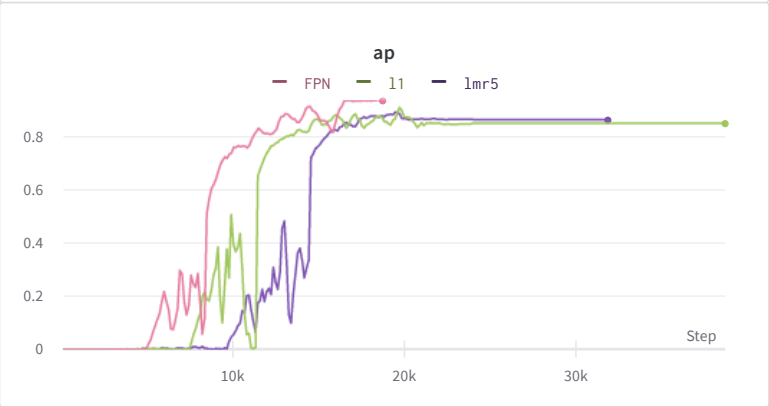
1-2 of 2

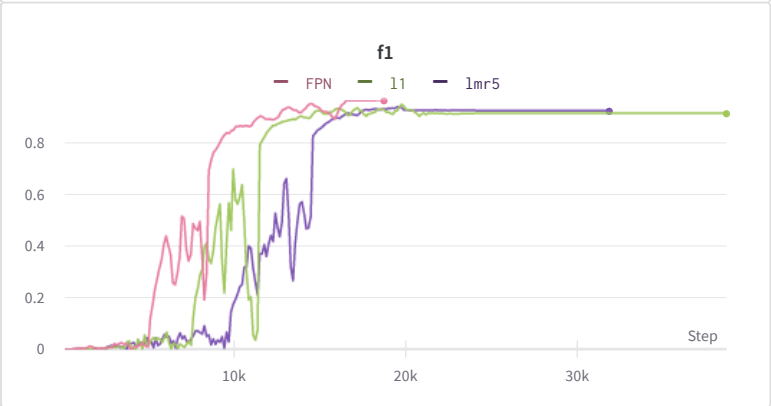
Feature Pyramid Network

FPN network has done significantly better and boosted the AP from 86 to 93% when trained on 30K/8K train/val samples. FPN converged faster, the results are after ~150 epochs of training taking ~3hrs on Tesla T4 GPU, while runs, 'l1', 'lmr5' using the basic model took more time to reach lower results. It is also important to note that the basic models where trained on P100s, a considerable difference has been noticed when changing the GPUs at a very later stage in the experiments. With Respect To FPN, the training was stuck especially, the classification head at a very early stage in training and never or very slowly converges. The reported results are reproducible only with Tesla K80/T4 GPUs. It is strongly suggested to train on Colab for reproduction.









Run set 3

Name (3 visualized)	epoch	ap	epoch	train_len	val_len	f1	prec	train_loss	val_loss
FPN	-	0.9359	143	32000	8000	0.9633	0.9622	0.5046	2.201
l1	300	0.85	300	32000	8000	0.9131	0.9119	5.61	2.837
lmr5	248	0.863	247	32000	8000	0.9227	0.9217	0.3673	2.993

1-3 of 3

Test Results

Results after loading the checkpoints (using main.py inference file)

AP on 1000 samples: 0.9178300024569035
AP on 2000 samples: 0.9316318625211717
AP on 8000 samples: 0.9359464694932108