**K19-1061 Furqan Tariq**

**Introduction:**

This project is implementing an object detection model using the VOC dataset. It is using pre-trained ResNet-50, VGG-16, Inception-v3, MobileNet-v2 and Densenet-121 models from torchvision and fine-tuning it for the specific task of object detection.

**Working:**

The project includes data loading, preprocessing, model definition, training, and evaluation. The dataset is loaded using the `VOCDetection` class from torchvision and is transformed using `transforms.Compose`. The images are resized, normalized, and transformed to tensors. The labels are transformed to one-hot vectors.

The model are defined using their default properties and a final linear layer with the same number of outputs as the number of classes in the dataset. The loss function used is `nn.BCEWithLogitsLoss()`, which is a binary cross-entropy loss function that is applied to each output independently. The optimizer used is `optim.Adam()` and the learning rate scheduler is `lr\_scheduler.StepLR()`. The training is done for a specific number of epochs and the model weights that result in the best mean average precision (mAP) score are saved.

During training, the model is run for each epoch on both the training and validation datasets. The model is set to either training or evaluation mode depending on the phase. The running loss, micro F1 score, sample F1 score, and average precision (AP) score are calculated and stored for each iteration. The F1 score is used to evaluate the model's precision and recall for each class, and the AP score is used to evaluate the model's accuracy in predicting object instances across all classes.

After training, the model is evaluated on the test dataset using the best saved model weights. The same metrics (loss, micro F1 score, sample F1 score, and AP score) are calculated and displayed using the `classification\_report()` and `ConfusionMatrixDisplay()` functions from scikit-learn.

The project includes visualization of the metrics using `matplotlib` and logging of the metrics using `torch.utils.tensorboard`.

We tried to apply Early and Late Fusion techniques to these models after their training but due to some complications, we couldn’t succeed in that but we have submitted our approach for that purpose as a separate file.

Overall, this code provides a comprehensive pipeline for object detection using a pre-trained CNN architectures and the VOC dataset.

**Results:**

*\*Results mentioned below are taken from the submitted notebook.\**

* **ResNet-50:**

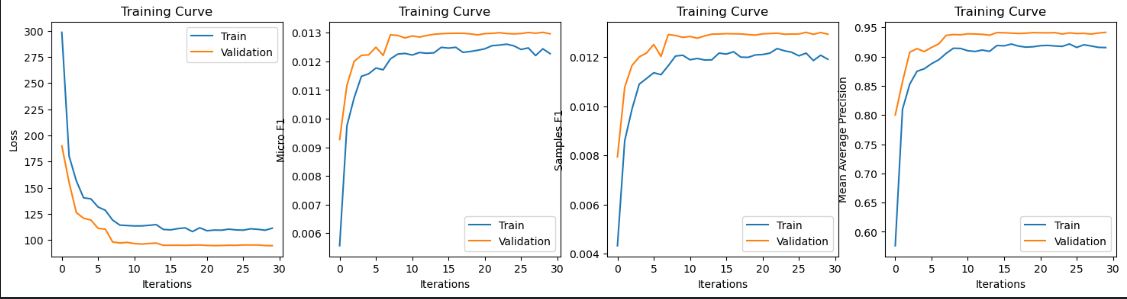
Train Loss: 111.3528 Mean Average Precision: 0.9155 Mean Micro F1: 0.0123 Mean Samples F1: 0.0119

Trainval Loss: 94.6355 Mean Average Precision: 0.9416 Mean Micro F1: 0.0130 Mean Samples F1: 0.0129

Best val Mean Average Precision: 0.941614

Test:

Mean Loss: 102.8458 Mean Average Precision: 0.9257 Mean Micro F1: 0.0125 Mean Samples F1: 0.0124



* **VGG-16:**

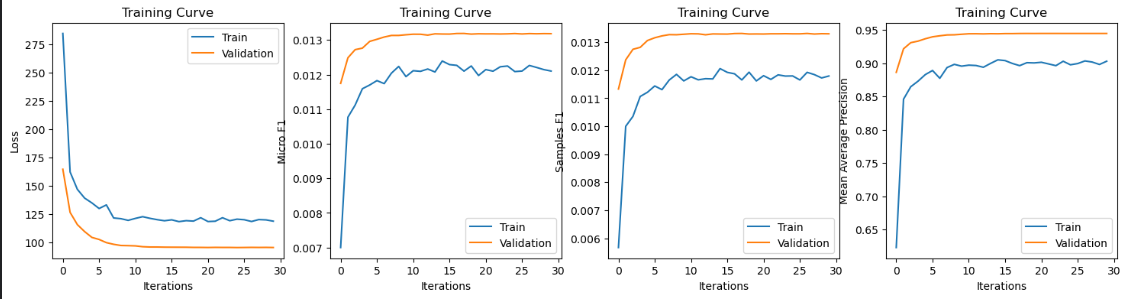
Train Loss: 118.9045 Mean Average Precision: 0.9030 Mean Micro F1: 0.0121 Mean Samples F1: 0.0118

Trainval Loss: 95.6654 Mean Average Precision: 0.9447 Mean Micro F1: 0.0132 Mean Samples F1: 0.0133

Best val Mean Average Precision: 0.944709

Test:

Mean Loss: 107.6070 Mean Average Precision: 0.9225 Mean Micro F1: 0.0127 Mean Samples F1: 0.0127



* **Inception-v3:**

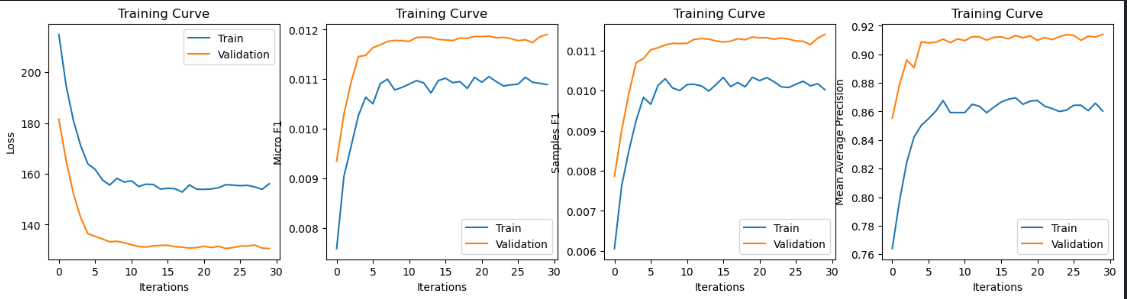
train Loss: 156.1003 Mean Average Precision: 0.8603 Mean Micro F1: 0.0109 Mean Samples F1: 0.0100

trainval Loss: 130.5756 Mean Average Precision: 0.9140 Mean Micro F1: 0.0119 Mean Samples F1: 0.0114

Best val Mean Average Precision: 0.913974

Test:

Mean Loss: 133.9928 Mean Average Precision: 0.8973 Mean Micro F1: 0.0117 Mean Samples F1: 0.0111



* **MobileNet-v2:**

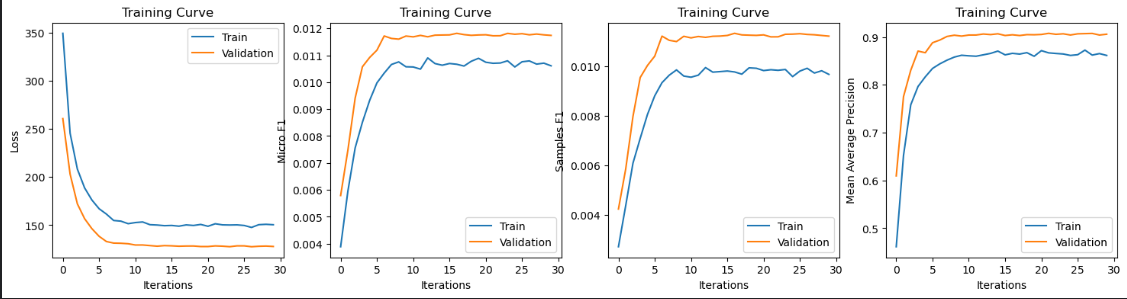
Train Loss: 150.4928 Mean Average Precision: 0.8615 Mean Micro F1: 0.0106 Mean Samples F1: 0.0097

Trainval Loss: 127.9187 Mean Average Precision: 0.9059 Mean Micro F1: 0.0117 Mean Samples F1: 0.0112

Best val Mean Average Precision: 0.907792

Test:

Mean Loss: 130.3331 Mean Average Precision: 0.8973 Mean Micro F1: 0.0116 Mean Samples F1: 0.0110



* **DenseNet-121:**

Train Loss: 137.9545 Mean Average Precision: 0.8956 Mean Micro F1: 0.0112 Mean Samples F1: 0.0105

Trainval Loss: 125.6817 Mean Average Precision: 0.9208 Mean Micro F1: 0.0120 Mean Samples F1: 0.0115

Best val Mean Average Precision: 0.920849

Test:

Mean Loss: 127.8764 Mean Average Precision: 0.9060 Mean Micro F1: 0.0118 Mean Samples F1: 0.0113

