AML Assignment-1 Report

Note: <u>All experiments results can be seen here.</u> Also, please filter the option to see the results for specific experiment details.

Code Explanation:

- Checking Available GPU: This code block checks if a GPU is available and prints its information using the nvidia-smi command.
- Importing Requirements: The necessary libraries and packages are imported. This includes PyTorch for deep learning, matplotlib for visualization, numpy for numerical operations, and some additional libraries like torchmetrics and torchinfo.
- Device Agnostic Code: It checks whether a GPU is available and sets the device variable accordingly to either "cuda" (GPU) or "cpu" (CPU).
- Getting Dataset: The code prepares the dataset for training and testing. It uses the CIFAR-10 dataset in this example, but there's commented-out code for another dataset (Flowers102) as well. Data transformations are defined, and both training and testing datasets are created.
- Model Definition: A custom neural network model called Net is defined. It consists of convolutional layers, dropout layers, and fully connected layers. The model architecture is defined in the __init__ and forward methods. There's also a stochastic_pred method that applies dropout at test time for uncertainty estimation.
- Hyperparameters: Hyperparameters such as the number of epochs, learning rate, batch sizes, and others are defined.
- Initial Training Setup: It sets up data loaders for the initial labeled and test datasets, defines the optimizer (Adam), learning rate scheduler, loss function (NLLLoss), and accuracy function.
- Active Learning Loop: The main active learning loop starts here. It runs for a specified number of episodes (MAX_EPISODES). For each episode:
    - a. It logs into Weights and Biases (WandB) for experiment tracking.
    - b. Active samples are selected from the unlabeled dataset using a specified sampling method (e.g., random or coreset).
    - c. A new neural network model is created from scratch (Net), and the optimizer, loss function, and accuracy function are reinitialized.
    - d. The model is trained on the updated labeled dataset for a specified number of epochs using the train function.
    - e. At the end of each episode, the model, optimizer, loss, and accuracy are saved with a unique identifier ("episode_1", "episode_2", etc.).

Results:

- # parms (dataset used in paper)
    - EPOCHS=10
    - LR=0.001
    - GAMMA=0.1
    - INIT_SIZE=100
    - AL_BSIZE=100
    - # SAMPLE_METHOD="coreset"
    - OUT_DIR="output"
    - DATASET_NAME="Cifar-10"
    - MAX_EPISODES=10
    - BATCH_SIZE=32
    - TEST_BATCH_SIZE=1000
    - DEVICE = device

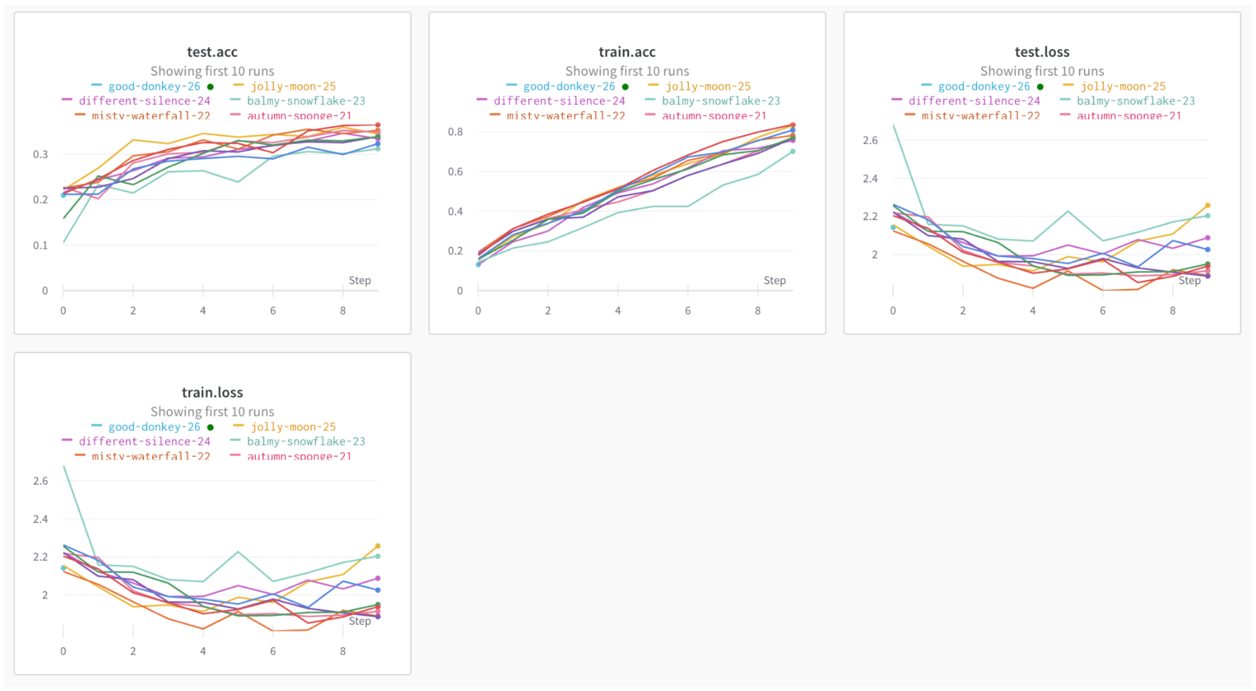

Fig: Results for cifar 10 datasets on coreset method.

- # parms (dataset used in paper)
    - EPOCHS=10
    - LR=0.001
    - GAMMA=0.1
    - INIT_SIZE=100
    - AL_BSIZE=100
    - # SAMPLE_METHOD="random"
    - OUT_DIR="output"
    - DATASET_NAME="Cifar-10"
    - MAX_EPISODES=10
    - BATCH_SIZE=32
    - TEST_BATCH_SIZE=1000
    - DEVICE = device

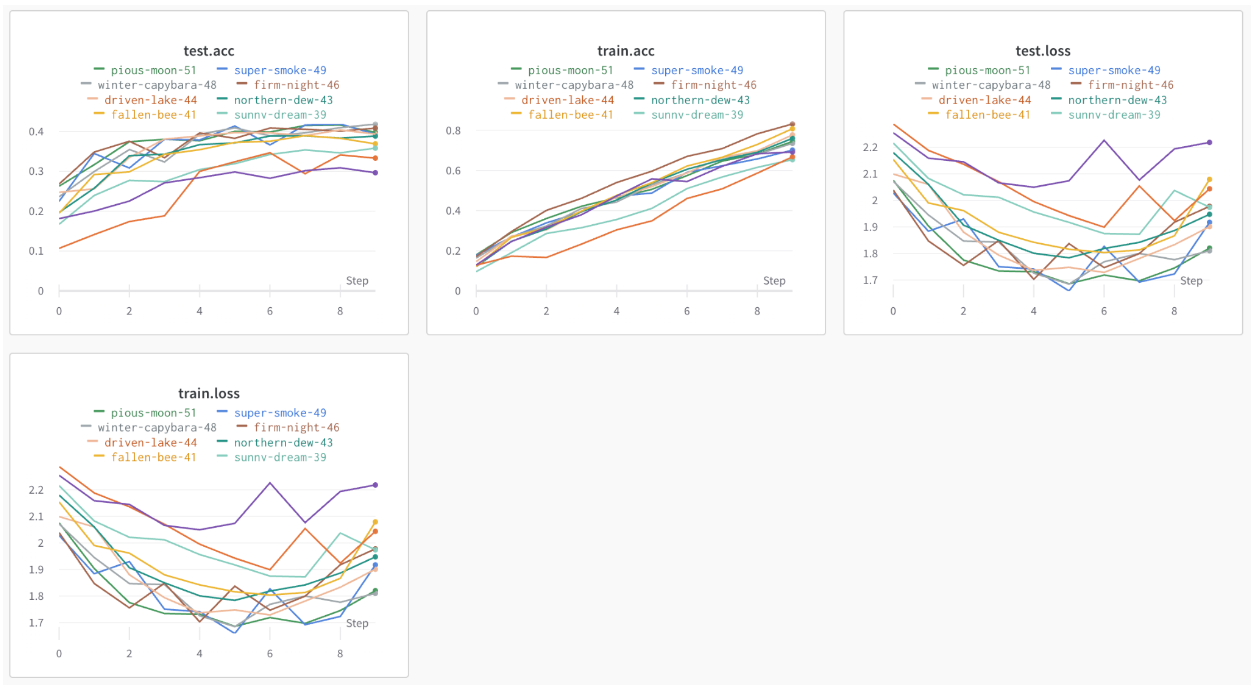

Fig: Results for cifar 10 datasets on random method.

- # parms (dataset not used in paper)
  - EPOCHS=10
  - LR=0.001
  - GAMMA=0.1
  - INIT_SIZE=100
  - AL_BSIZE=100
  - # SAMPLE_METHOD="coreset"
  - OUT_DIR="output"
  - DATASET_NAME="Flowers102"
  - MAX_EPISODES=10
  - BATCH_SIZE=32
  - TEST_BATCH_SIZE=1000
  - DEVICE = device

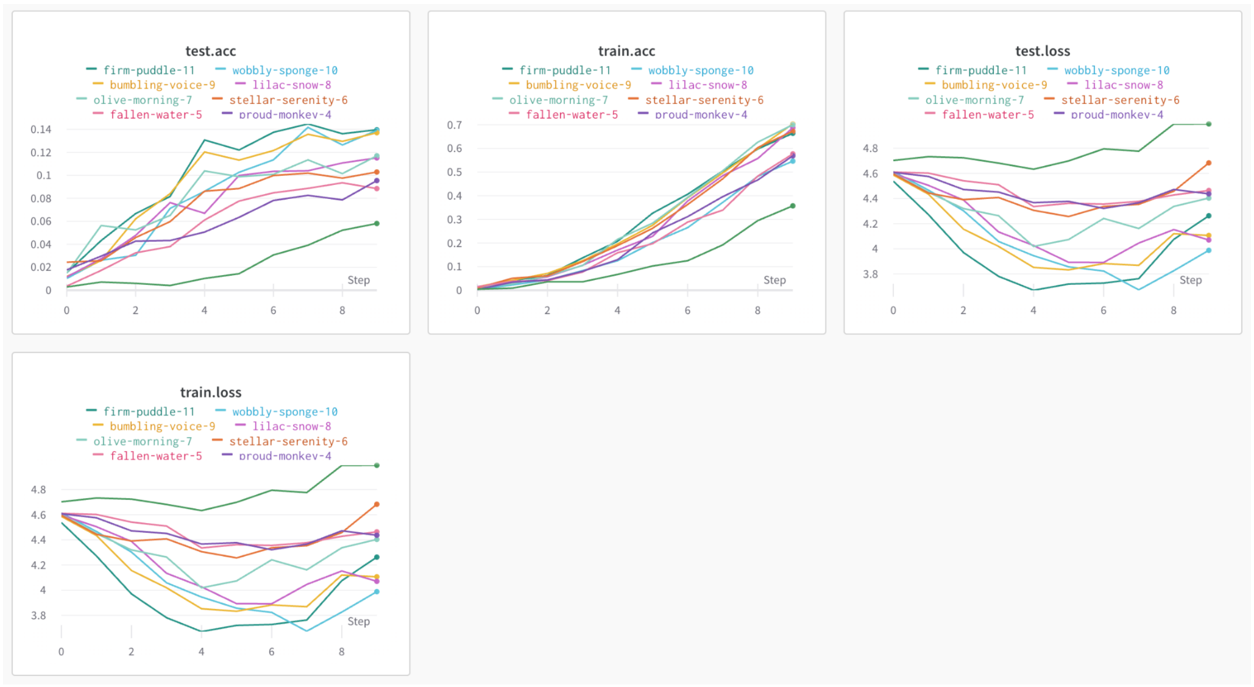

Fig: Results for Flowers102 datasets on coreset method.

- # parms (dataset not used in paper)
    - EPOCHS=10
    - LR=0.001
    - GAMMA=0.1
    - INIT_SIZE=100
    - AL_BSIZE=100
    - # SAMPLE_METHOD="random"
    - OUT_DIR="output"
    - DATASET_NAME="Flowers102"
    - MAX_EPISODES=10
    - BATCH_SIZE=32
    - TEST_BATCH_SIZE=1000
    - DEVICE = device

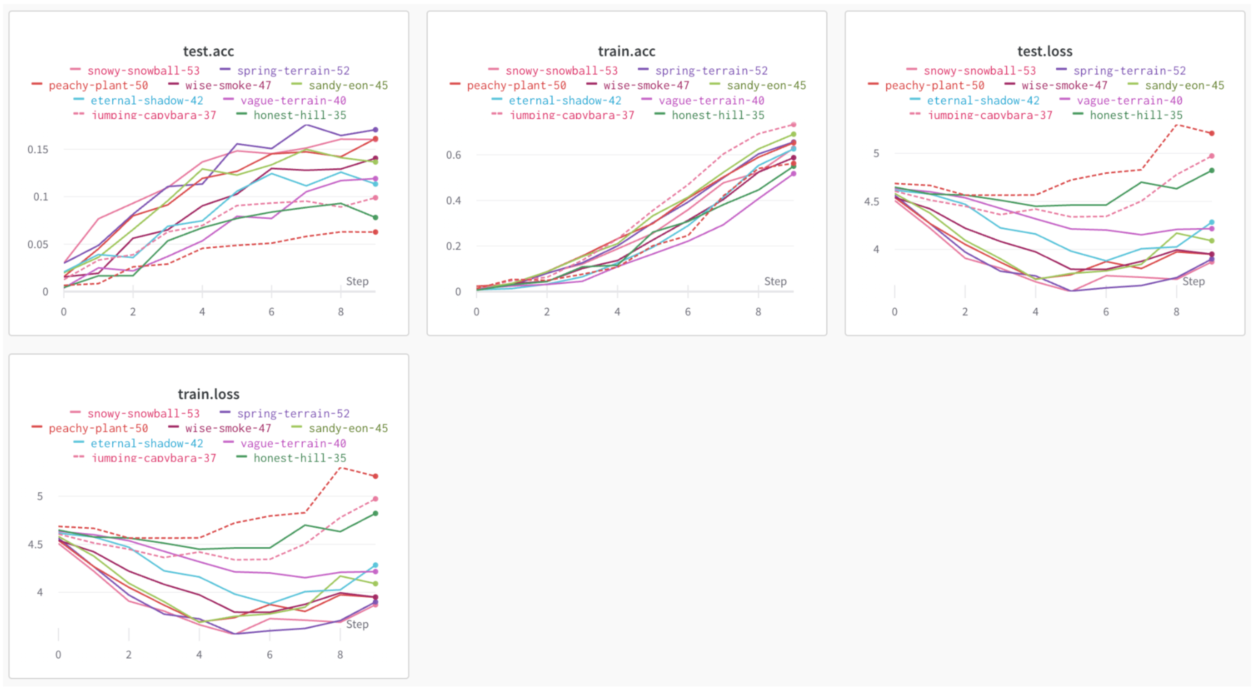

Fig: Results for cifar 10 datasets on random method.

Analysis of Result:

- During Each episode, the oracle gives 100 (according to parms) newly annotated samples that are also included in the training on the model, leading to better mode accuracy.
- This is because of sample fact: more data => better model.(true in this case since very little data is being annotated)
- As per our experiment setup, we will have 10 episodes with 100 samples each, resulting in 1000 annotated samples for the last trained model.

- Since the models have been training from scratch & training for fewer epochs, this explains the reason for the stats shown in the graphs.
- The monotonic rate of better performance proves the hypothesis that coreset approach for active learning.
- The code shows similar results as of the papers' for my implementation of the paper since the code of the official repo is ancient and not compatible with the current versions of libs.
- The results are similar, not the same because we have reduced the number of episodes and other hyperparms for paper replication because of the compute resources.
- The results can be proven to be in the same ranges using the interpolation.

References:

- https://github.com/svdesai/coreset-al
- https://www.kaggle.com/code/puru98/federated-learning-pytorch
- https://medium.com/pytorch/https-medium-com-robert-munro-active-learning-with-pytorch-2f3ee8ebec
- https://devblog.pytorchlightning.ai/active-learning-made-simple-using-flash-and-baal-2216df6f872c
- https://docs.wandb.ai/
- https://arxiv.org/abs/1708.00489