**Continuing CL on edge using TensorFlow-Lite**

1. **Offline Experiments:**

We want to achieve greater accuracy than Latent Replay ([[1912.01100] Latent Replay for Real-Time Continual Learning (arxiv.org)](https://arxiv.org/abs/1912.01100)) before going into application:

The changes tried below are proposed in this paper: [[2010.05595] Rethinking Experience Replay: a Bag of Tricks for Continual Learning (arxiv.org)](https://arxiv.org/abs/2010.05595)

1. **Done -> Try different optimizer like Adam** to see if we get better accuracy and maybe some different learning rates experiment. Best is SGD so far.
2. **In Progress ->** **Independent Buffer Augmentation (IBA).**

See **Data Augmentation** to help in class balancing too (e.g., rotate image, etc. There are libraries in TensorFlow probably, check those)

* 1. We don’t save the augmented data in the replay buffer, we just generate them for that training batch and it’s over.
  2. Could check if CORe50 had something with data augmentation (Another paper maybe)
  3. The data augmentation will mean that we have to store the raw images in the replay buffer and not only the features since we want to augment on the training step and not save any augmented images (you can’t augment from image features, you need raw images). It is not a big issue but it makes the model occupy more space.

1. **In Progress ->** See how I can **achieve class balancing and change of the heuristic with the replay buffer replacement algorithm**

**Trying BRS, LARS**

* 1. Think how it would affect if we had mini-batches or a lot of training batches and had a class introduced way later.
  2. BRS and LARS need to be tested on a new NIC CORe50 benchmark since the current random sample buffer replacement algorithm that is shown in the latent replay paper as well, seems to be overfitted to the NICv2 benchmark.

**Done -> BRS** is fully implemented and can be found in the code repository.

**In Progress ->** **LARS** has not been fully implemented since it is tricky to find a way to be able to obtain individual loses for each training sample in a training batch that the algorithm requires.

1. **Done -> Exponential LR Decay (ELRD):**

Implemented as seen on the CL bag of tricks paper. Didn’t produce better results but some more experimentation with different parameters on the decay might have better results.

1. **NOT NOW** but Few-Shot Generative Replay is a research area that we can see instead of data augmentation. We have like a generative model that will produce data for class balancing.
2. **Application Related:**

Doesn’t concern us right now since the main goal is to increase the model’s accuracy.

1. Once we get better accuracy, we can transfer the new model to the application.
2. We also want to increase the available classes to be more than 4. (Could also have dynamic classes instead of static)
3. Can also see about evaluating the performance of the model within the application (android studio benchmarks to evaluate a variety of metrics and not just accuracy).
4. **Offline Experiments – A new benchmark:**
5. Find a way to create a custom NIC scenario of CORe50.
6. Check Continuum and avalanche.
7. Something similar was done here: [HamedHemati/CIR: Class-Incremental Learning with Repetition (github.com)](https://github.com/HamedHemati/CIR)