We thank the reviewers for their time and insightful feedback. We now address the concerns. **C** - Concern, **R** - Response

(*) To all reviewers. C: **Dataset choices and the scale of the experiments.** R: While our proposed method is more generally applicable, we showcased it for experience replay-based continual learning and streaming, where the idea of coresets is largely under-explored. In these scenarios, the *standard datasets* used for evaluation are *SplitMNIST and PermMNIST* and their variations, see [44, 16, 2, 9], and the standard architectures are fully-connected nets or simple CNNs. Commonly, the experience replay size is also restricted to a few samples to emulate a memory-constrained environment. Thus, we decided to use the *same setup* also in our paper to *conform to the practices* in the field and to be easily comparable to competing methods, e.g., VCL. For the final version of paper we will add more summarization and continual learning experiments of CIFAR. **C: Proxy kernels and ResNets. R:** With recent libraries (e.g., [45] used in our work), the CNTK can be *calculated efficiently* even for ResNets. To prove this point, we *repeated our imbalanced streaming* experiment with the CNTK of *ResNet-18* and report the results in Table 1. This kernel provides an accuracy improvement 1.7% over the kernel used in the submission at the expense of moderately increased runtime, as shown in Table 2. However, CNTK is only one particular proxy choice. For the revised paper, we will expand our discussion and results on the proxy choices. **C: Theoretical guarantees. R:** We proved in the paper that with L2 loss and infinitely wide neural networks our coreset construction provides convergence of order 1/T as per Theorem 1. However, with other losses and finite-width neural networks both the bilevel coreset and neural network optimization problems become NP-hard in general, thus convergence guarantees are not easily obtainable. We will clarify this in the paper.

Table 1: New imbalanced CIFAR-10 streaming results.

	SplitCIFAR-10
Coreset + SimpleCNTK (original)	32.30 ± 0.84
Coreset + ResNet-18CNTK	33.98 ± 1.44
OCL [A]	32.25 ± 1.69

Table 2: Runtimes for generating a coreset of size 100 out of 1000 points with CNTK.

	SimpleCNTK	**ResNet-18CNTK**
Coreset gen	57.3 s	63.1 s
Kernel calc	6.3 s	56.2 s

Reviewer #1. C: **Uniformly sampled, weight-optimized CNTK summary performs better than RBF coreset.** R: This is indeed the case, since the CNTK is better suited for images than the RBF kernel, as it takes into account the inductive biases of locality and translation invariance. Thus, the performance gap is due to the kernel choice rather than the bilevel optimization scheme. For choosing the right kernel, we opt in the paper for choosing the CNTK equivalent to a neural network that works well for the specific problem.

Reviewer #2. C: Dataset choices (mostly MNIST) and further theoretical guarantees. R: Please see **(*)**. **C: Evaluation with ResNets and CIFAR-10. R:** Please see **(*)**. You correctly observe that we used "proxy of a proxy" in the imbalanced streaming experiment. We used it for keeping the coreset generation time the lowest possible, while still achieving good results. In our new CIFAR-10 experiments shown in Table 1, the CNTK corresponding to ResNet-18 provides an additional 1.7% compared to the SimpleCNTK proxy reported in the paper, but is 50 s slower in generating a coreset of size 100 (Table 2). We have also performed the continual learning experiments on SplitCIFAR with ResNet-18 and ResNet-18CNTK as proxy, and observe a 1.9% improvement with coresets over uniform sampling; we will report the rest of the results and provide a more detailed discussion on the proxy choices in the final version of our paper.

Reviewer #4. C: Coreset generation time burden. R: Please see **(*)**. We restricted the coreset sizes to the order of hundreds as this is the standard evaluation practice in the continual learning literature. In this setting, our coreset construction time is lower than the training time, thus our method is at most twice as slow as the fastest competing method. We will provide wall-clock time measurements for summary generation sizes of the different methods in the final version of the paper. **C: Imbalanced Streaming comparison to [2]. R:** Similarly as reported in the streaming setting (line 302), we did perform such an experiment, but we were not able to tune the method of [2] to outperform reservoir sampling - this is in line with the observation of the authors in [2] that their method requires a summary of size at least 1k to work on CIFAR-10. We will include this observation together with the time comparison.

Reviewer #5. C: Hardness of the baselines, CNTK choice, ResNets and further theoretical guarantees. R: Please see **(*)** and the new ResNet results in Table 1. **C: More challenging imbalanced streaming baselines. R:**. You suggest to improve reservoir sampling by a method "that samples from every class not uniformly but propositional to class frequency" - we note that this is *exactly* what reservoir sampling is doing on an imbalanced stream, i.e., keeps more samples from more representative classes, thus under-representing minority classes. We believe you intended to propose a class-balancing version of reservoir sampling similar to the Algorithm 1 of the concurrent work of [A]. We implemented this strategy designed for imbalanced streaming and report the results in Table 1, where we find it to match the performance of our method with the SimpleCNTK kernel, and slightly underperforming compared to the ResNet-18CNTK. We note that our method is general, and was not purposefully designed for imbalanced streams.

[A] Chrysakis et al. Online Continual Learning from Imbalanced Data, ICML 2020