Based on previous work it seems probable that in order to achieve the overall system design I need CL scaling laws that should be understood for different models.

7-3-24 LR Studies/Predictors

What is known:
1. Rate of loss decay per step training is directly proportional to model collapsing metrics and changes to the model structure. Due to the LR settings. But if a model learns nothing from a training step then it collapses still.
2. Training a model faster does not mean it will be damaged less.

What to explore here:
1. What is the rate of loss decay per step correlation to model size?
2. When mixing original training data (the pile) what ratio of it per batch mixed at a given loss decay rate prevents model collapse?
3. When mixing data and continual training what is the effect on model size

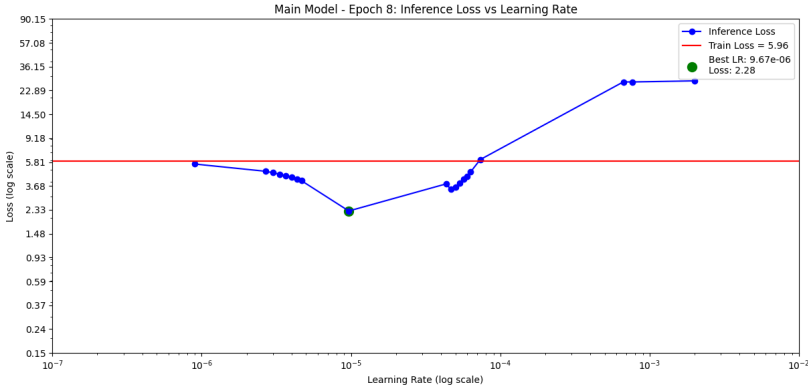
Initial Experiment Concept:

Mapping the LR loss landscape, choose specific loss decrease values that are a percentage less than the original loss.

Using those values train data on random pile data in certain ratios for each loss decrease value.

Run this over several different model sizes so scaling laws can be understood.

For example, given this LR loss plot of a given epoch:
Find the learning rate value equivalent to 1%,5%,10%, and a best loss



With those value train the model on several different ratios of data from the pile. Ensure you are looking at the loss of the data I want to learn not the pile data.
Stop training either of these:

- Cannot find a loss decrease value that is at the % decrease needed
- Correct count > 80%

Unknown:
data density and amount of training the model has gotten is probably a factor that is proportional to size. Current idea is that “density” of a model is proportional to size which is proportional to these metrics.

However scaling laws of fixed models are well known. https://arxiv.org/abs/2205.10487

Is choosing a fixed learning rate at the low end the same thing as following a specific loss decrease value?

- This may result in increased loss at time but it will probably still train. Albeit slower.

How this connects to the LR loss landscape stepping is not known which is the goal of the following work.

What I want to explore in the future:
- What is happening when choosing loss values that are greater than the initial training loss?
- Can I choose sets of reasonable learning rates and never learn?
- Why does choosing a single learning rate result in increased loss at times?
This is probably a boring result so may do it in the future.

See previous document for graphs that show:
- model collapse at a fixed rate is less than choosing the lowest loss value LR
- Model collapse at lower loss value decent is much lower than

I would also like to make a LR value that is a slopping value over the amount of weights, each layer getting different values. This is because when looking at the layer differences from training the layers change more than the earlier layers. I would simply like to balance this effect.

- This test will have to be done in the future as I don’t want to pollute my results with another unvalidated method. Especially since I am trying to show learning rate importance.
- Vanishing gradients may imply that adding higher learning rates in the beginning of the model would be changing the model more based on error that has backpropagation. This could result in more damage to the model.
- This feels like it could be an entire research project and should be done later.

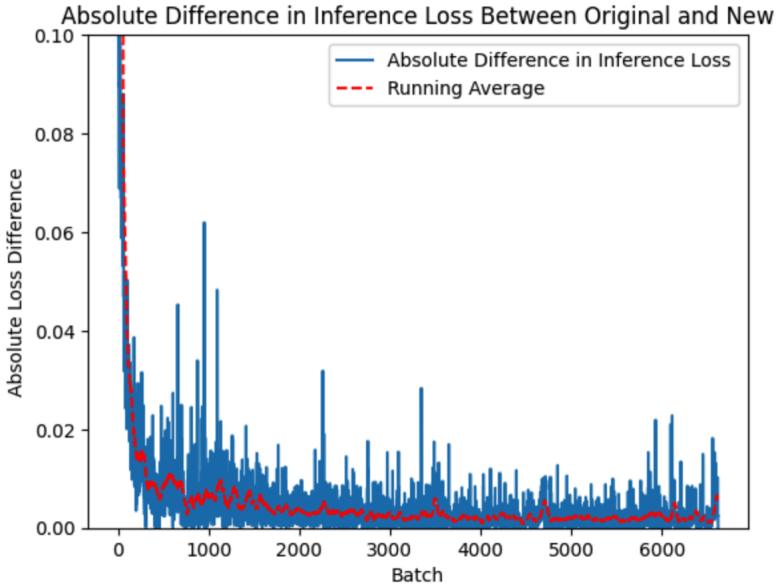
7-30-24

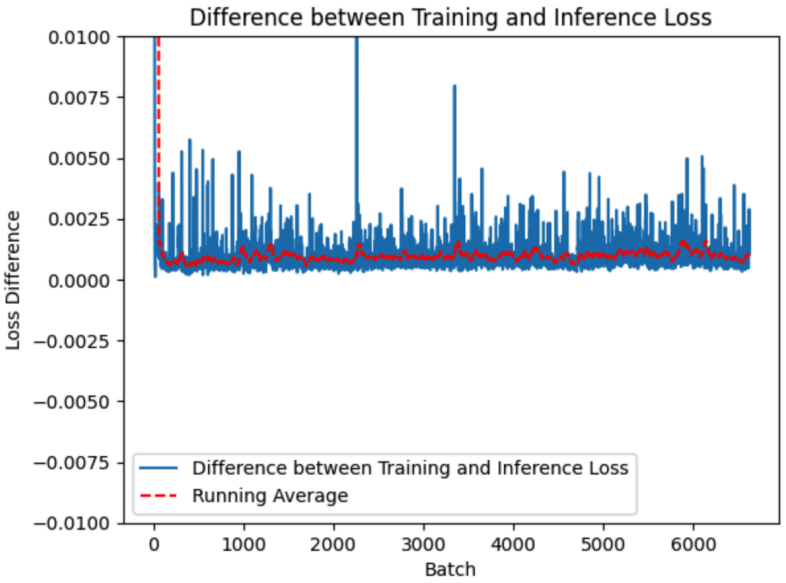
From previous studies it's seen that collapse can be mitigated by following the layer mad collapse plateau when training. Starting lower then slowly increasing the learning rate.

Steps for overall study:

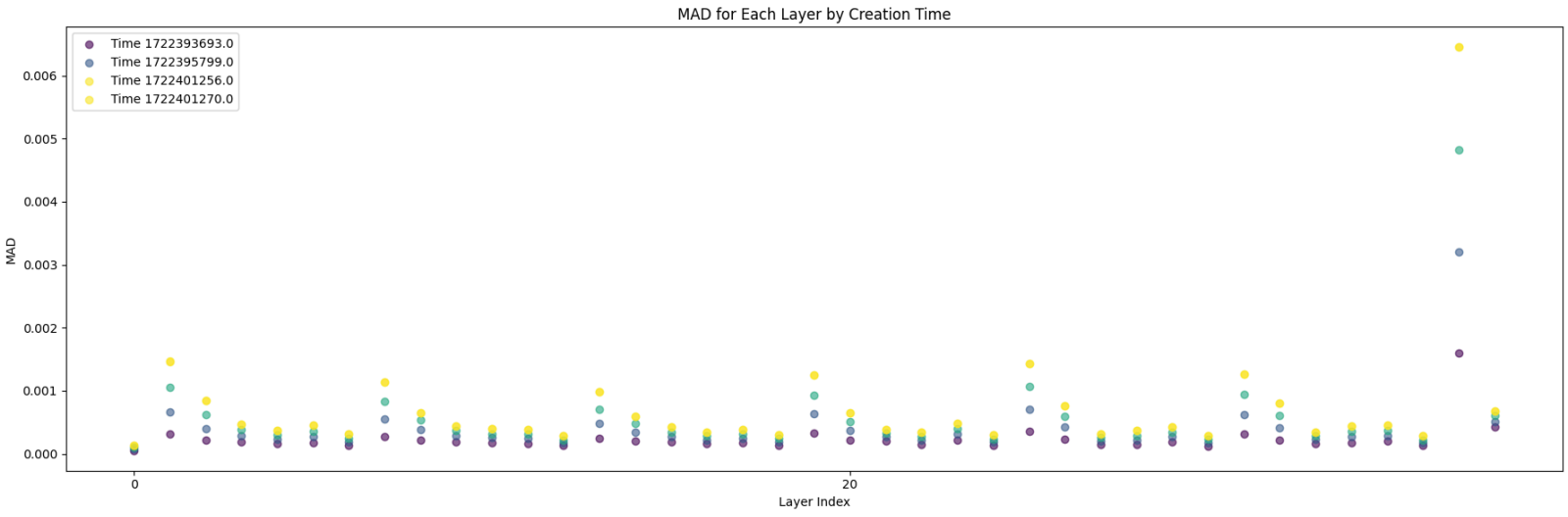
1. Show CL prevents general eval metric collapse via pile data mixing.
    First show this is done with the 10 questions from before
    Second show this is done with the DnD dataset
2. With these models trained start to look at if the knowledge trained on the DnD task is time dependently remembered by the LLM

7-31-24: memory in DnD data
- clear forgetting of data but it is very small. False this is actually a MEMORY signal
    - I want to increase this signal by increasing the amount of training step per batch

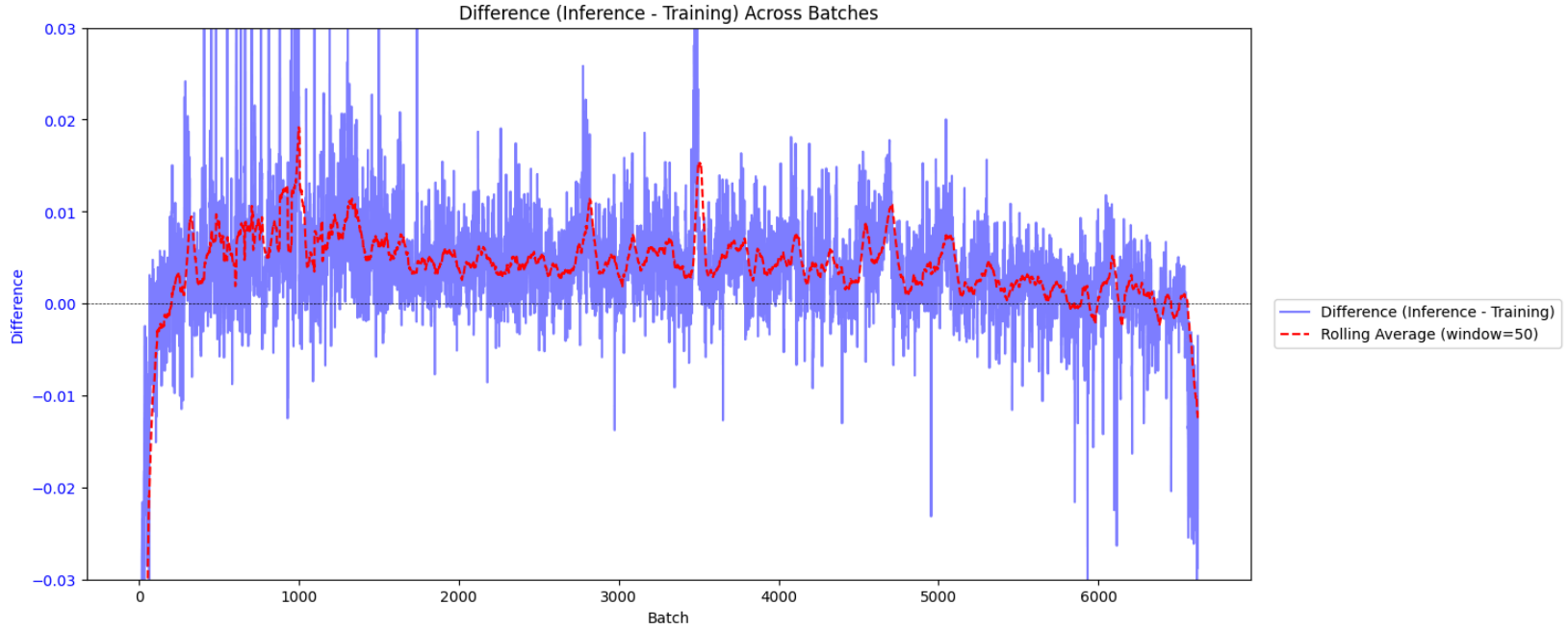




Collapse metrics are pretty crap but that's expected. Collapse to around 1-2e-3 so exponentially worse than previous work.

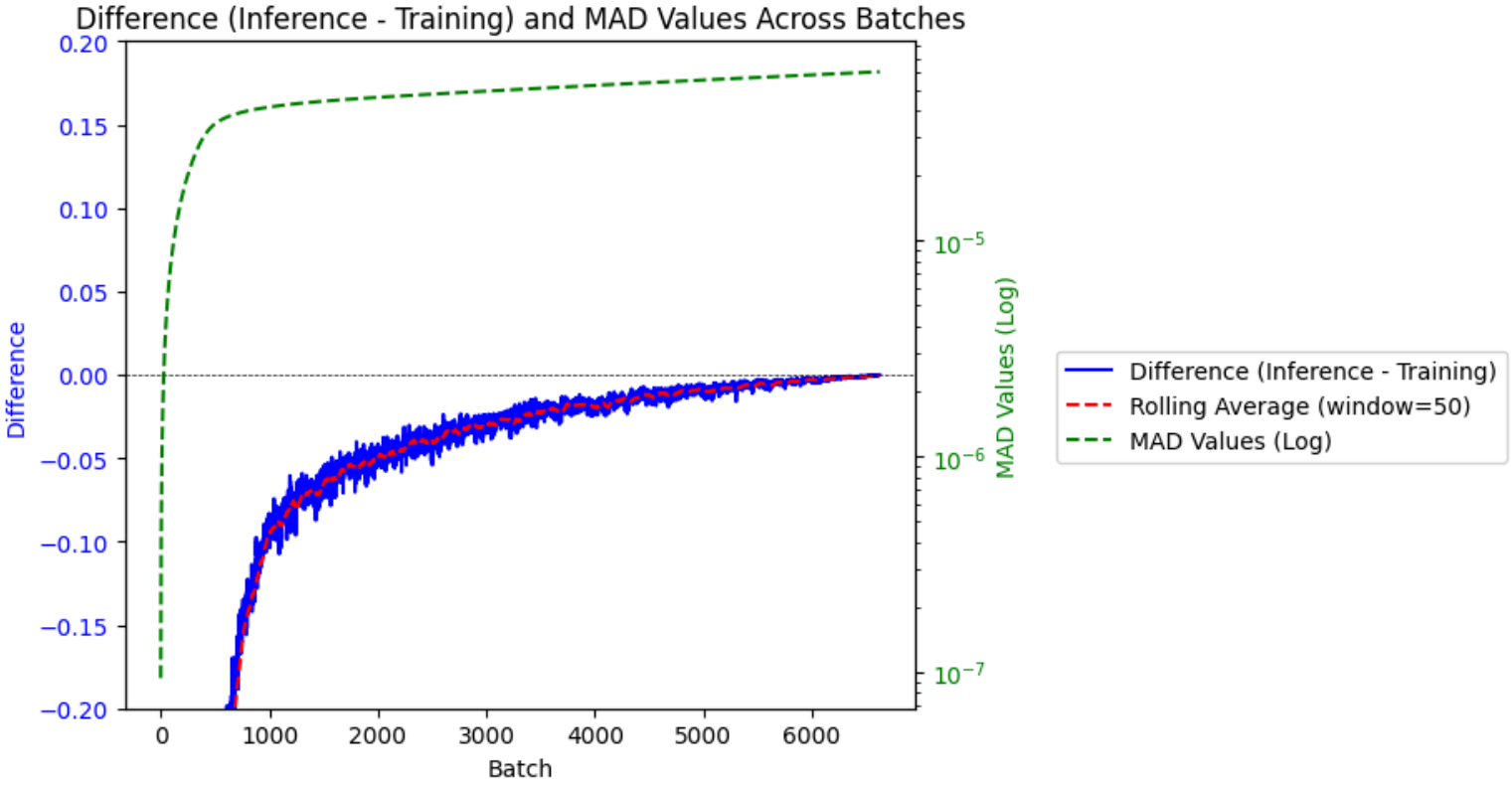


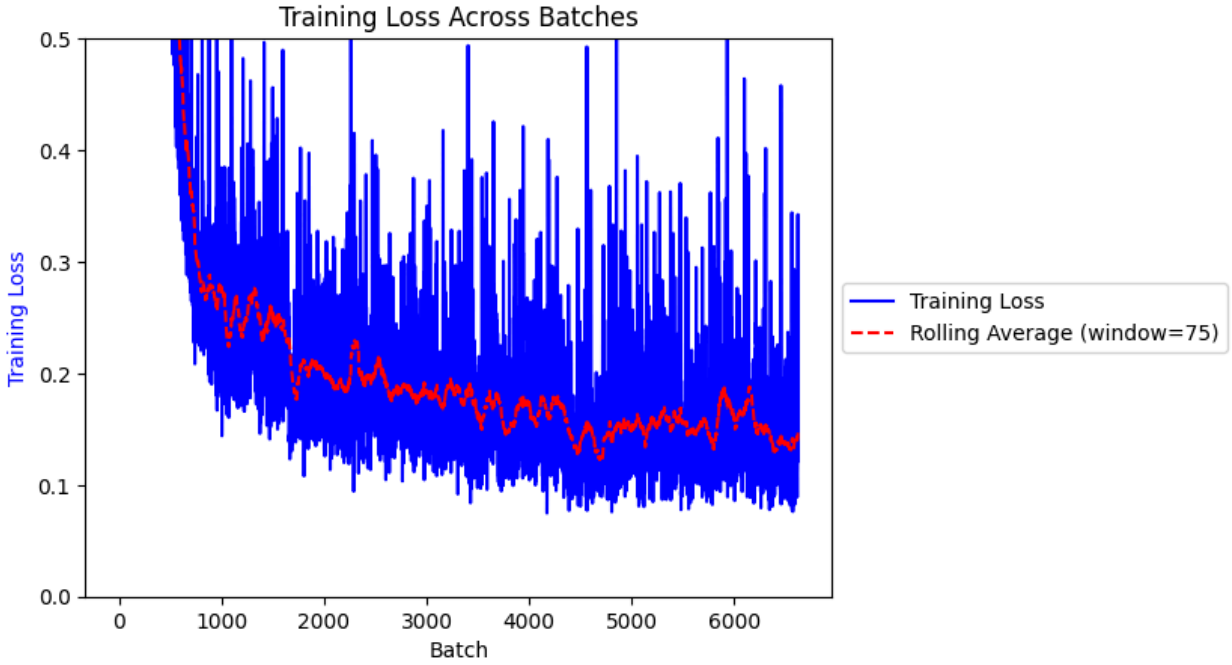
Training 4 time per batch:
Even smaller of a “forget” signal. It’s probably the opposite of what I’m thinking and this is heavily overtrained. The data is too similar. I would like a pareto front analysis.



Ok so I was wrong before in how I looked at it. This is the only example of the model actually forgetting the data. You can see that at the beginning is higher than the end and the very end the model clearly remembers it.

Learning rate 1e-7, all of these plots show how the model is better after training than before. This is opposite of what I want to show. I want to show that after training the loss on earlier is higher than when it started. If negative it is better after training, so the model forgot nothing.



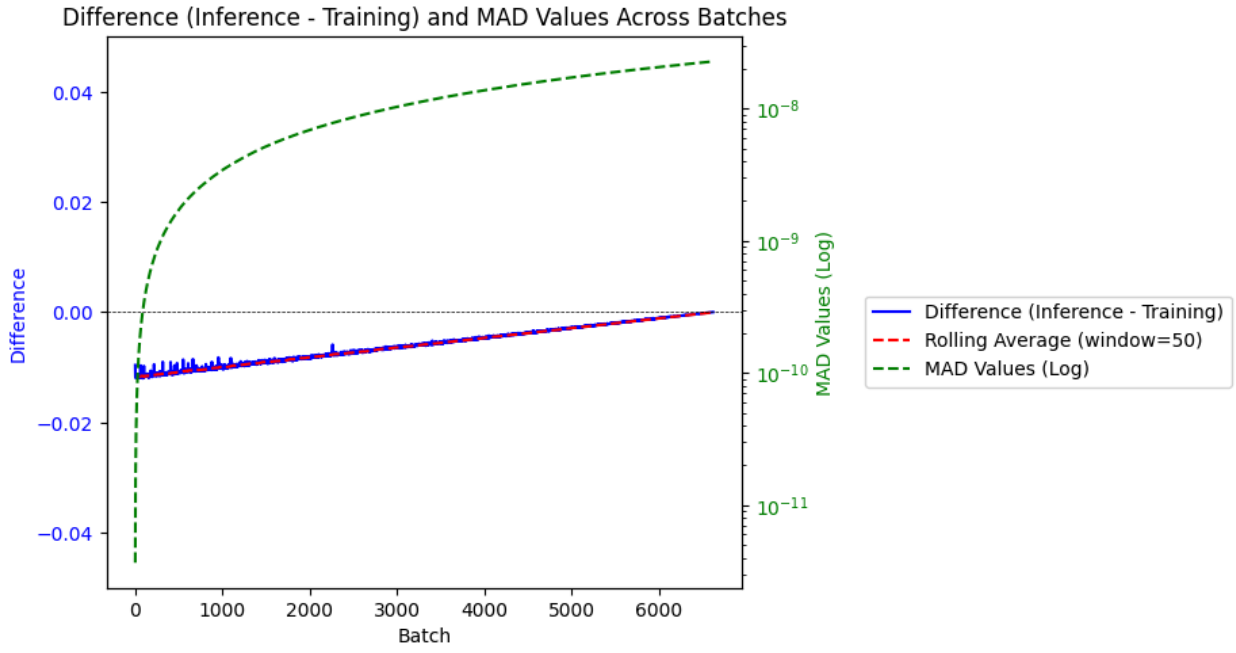


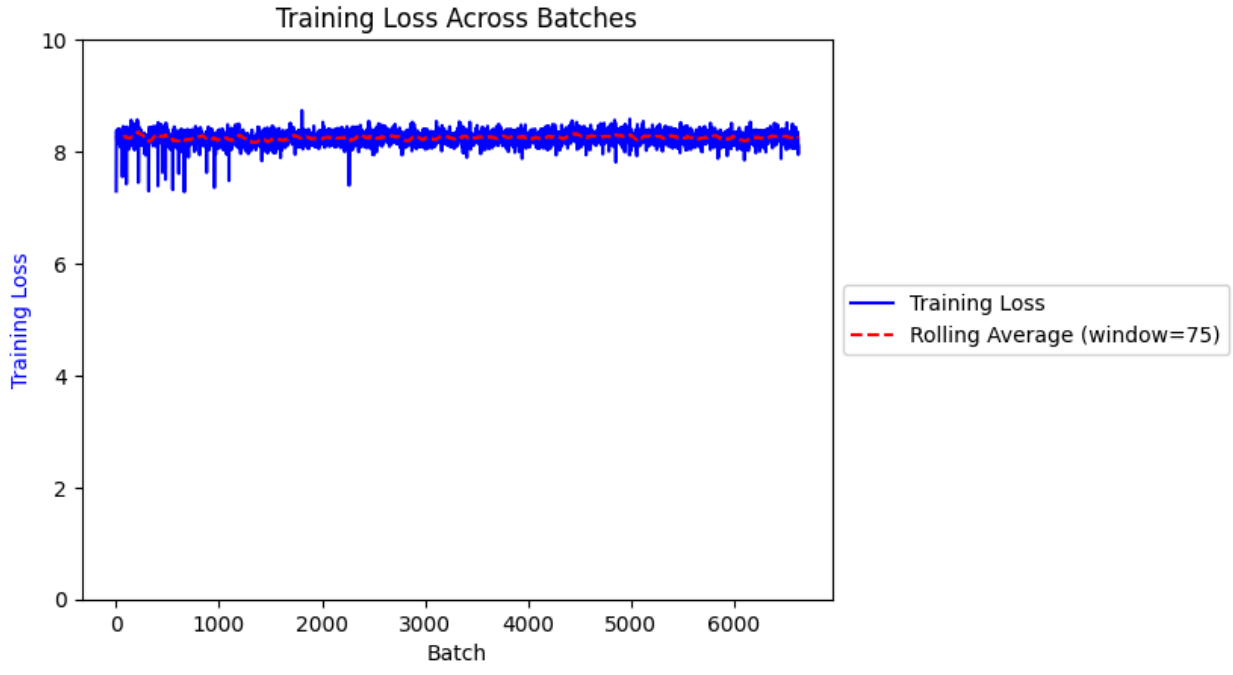
Try again with an extremely low learning rate. I also need to plot negative and positive not absolute so if it is negative that means it's better after training and if it is positive that means it forgot it. This data is extremely similar so this may be hard to find the signal.

Cycling between high and low LRs to facilitate forward momentum forgetting?

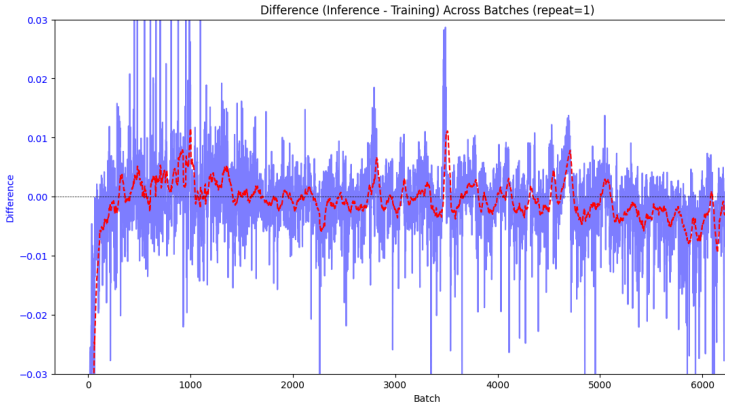
Maybe I'll only see this when I start mixing in pile data to prevent collapse?

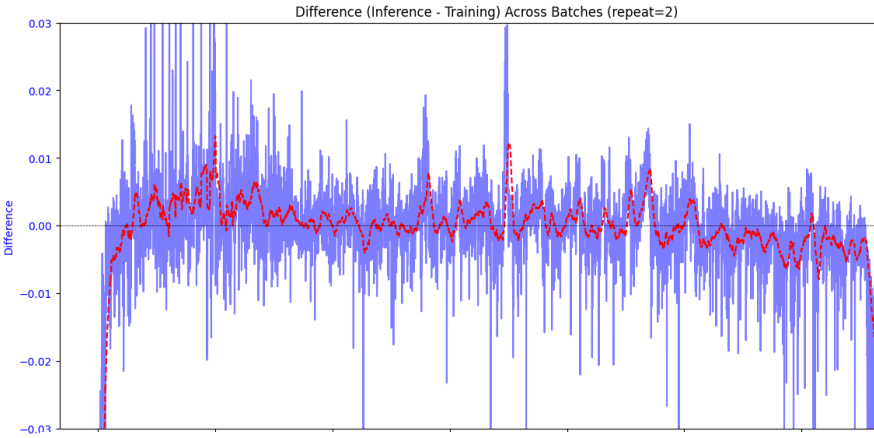
Lr 1e-10
If it doesn't work on this then low learning rates simply wont do it.

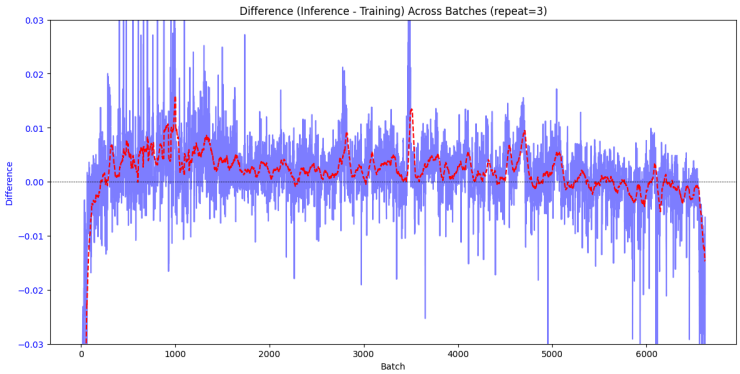


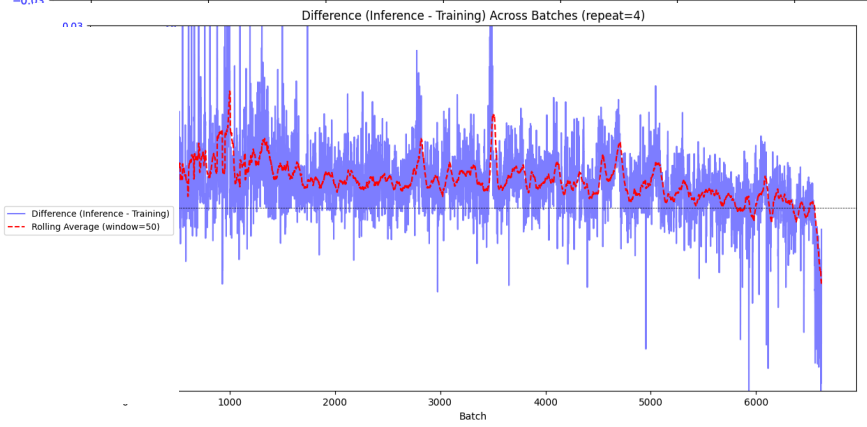


Overtrained Repeat Batches is what shows forgetting. Observe these plots. They are the repeat batch training by which repeat in increasing order. So the first one here is the first training of that batch and the last on is the 4 times trained repeatedly on that same batch (overfitting) on that one batch example. It is clearly seen that the early batches are closer to being remembered and like the others and the repeats are forgotten later in the model. YAY!

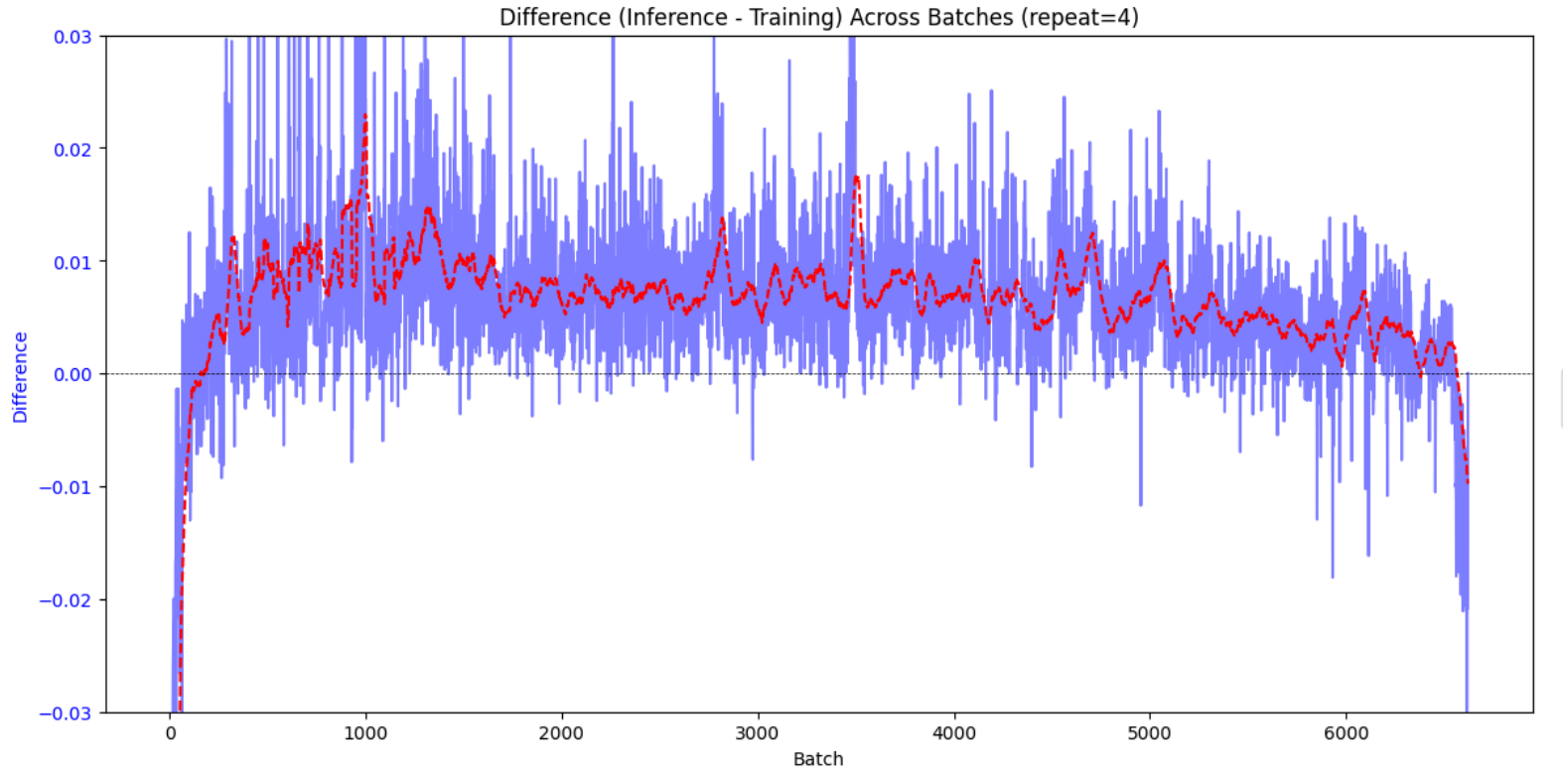




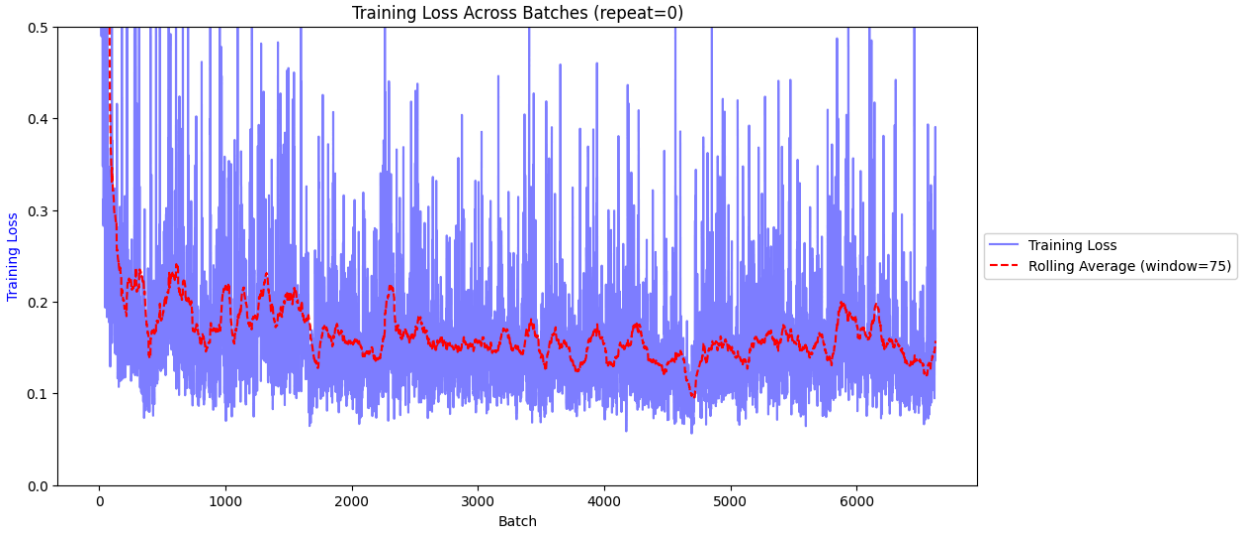


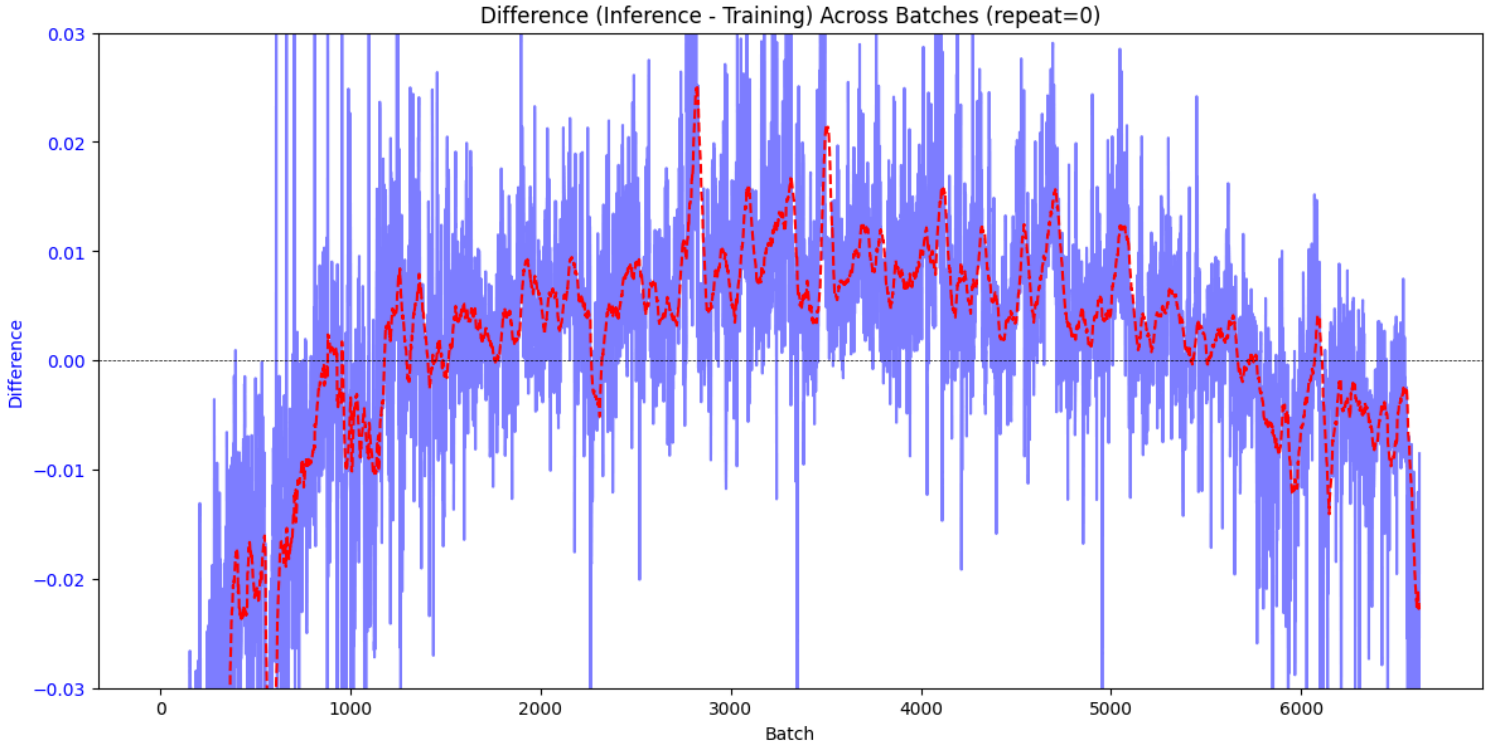


This last one is the inference loss on repeat 4 shows even stronger forgetting:



1e-3 training, very strong forgetting signal. Why is it in all these plots it remembered earliest and latest the best??? But forgets the middle the most?





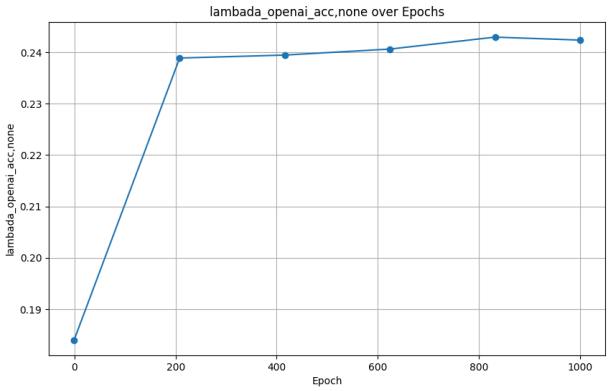
8-2-24: DnD pareto study + pile training

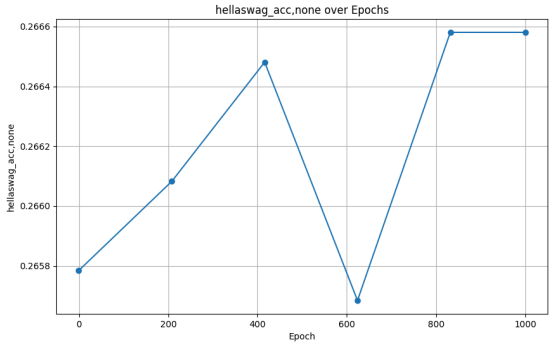
1 epoch, 10k examples of pile training:

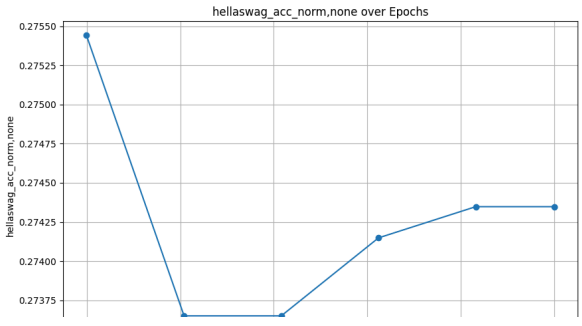
Results are confusing, seems wrong?

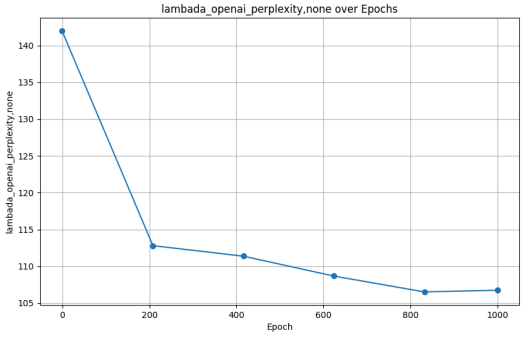
But it also shows it improved?

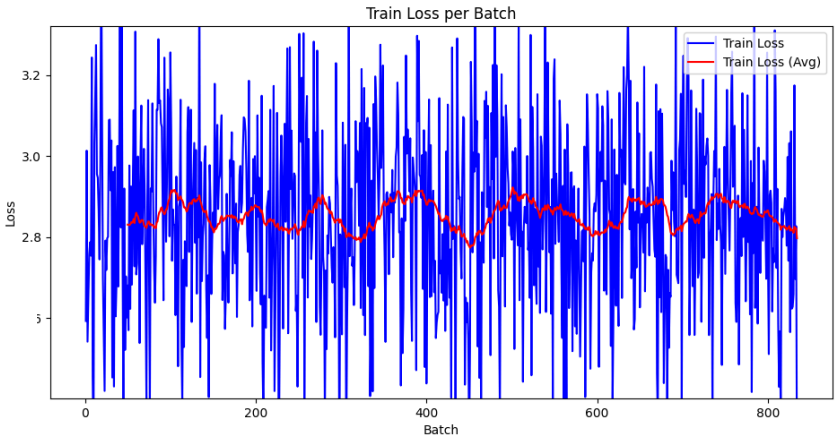
I want to repeat this with the 410m model

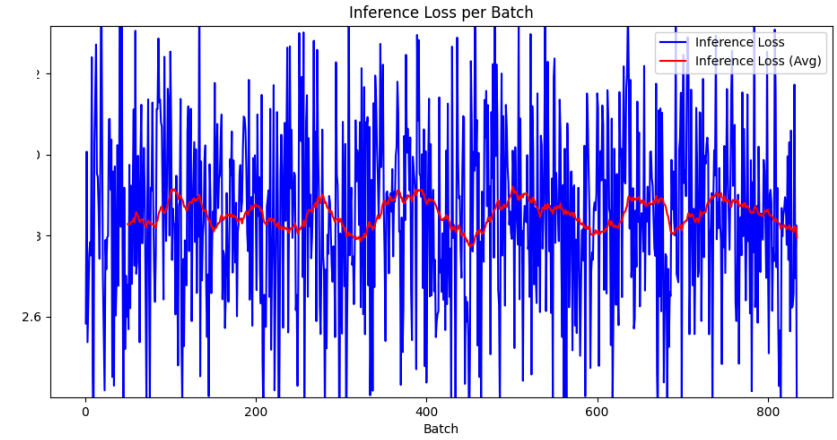


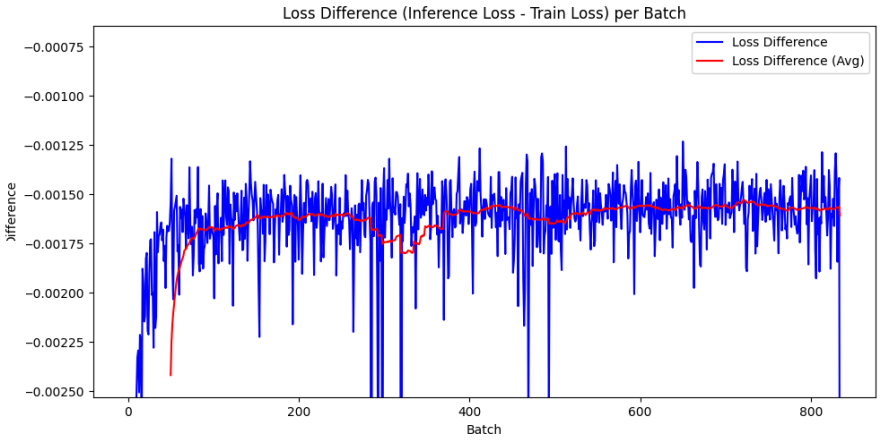


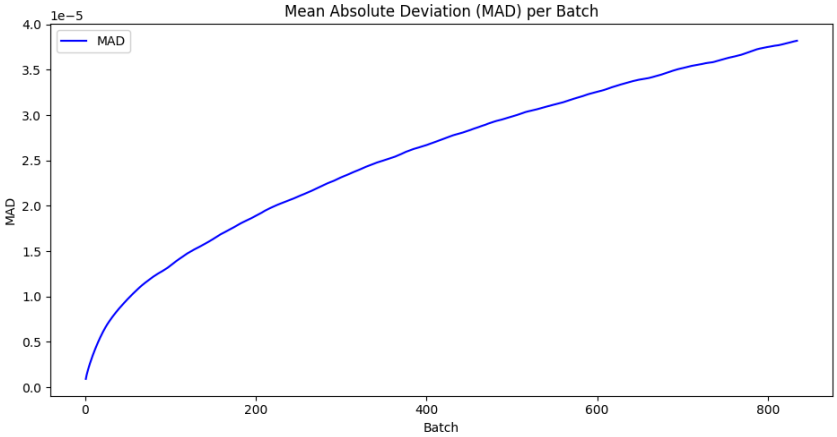








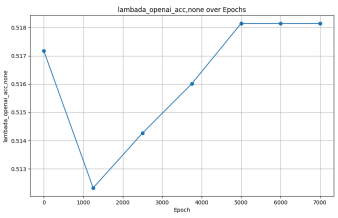


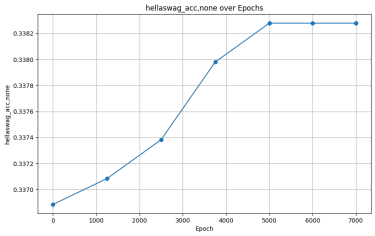


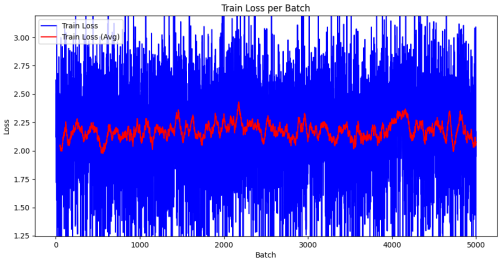
Repeat the above with the 410m model. I had to lower the batch size by a lot so I lowered the learning rate to 3e-7 a BS of 12 -> 2

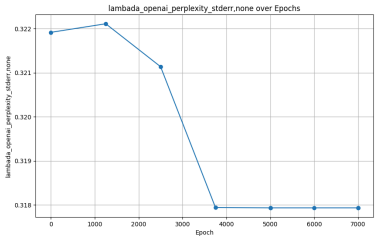
Another note: the mad value did not plateau like it did when repeatedly training on same data? -> 99% sure it's because only a single epoch and no repeat data

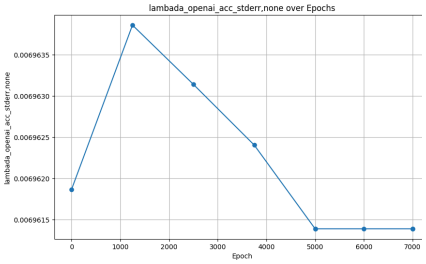
All metrics show improvement. Still has an initial drop on the first model? Clearly the initial adaption is the main driver of collapse

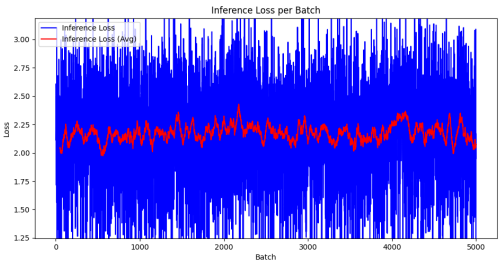


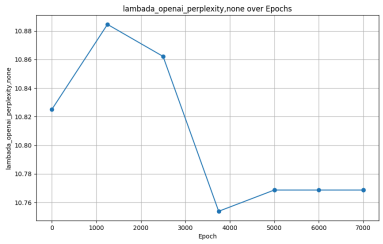


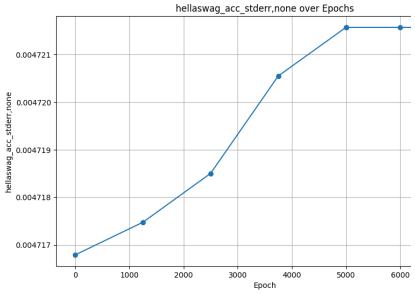


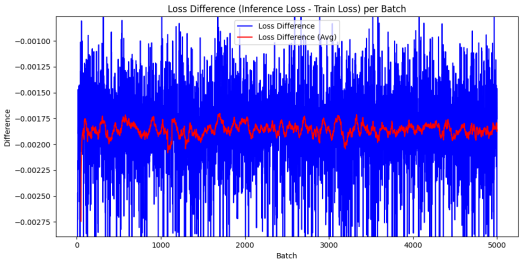


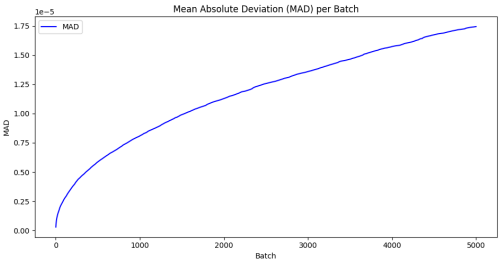


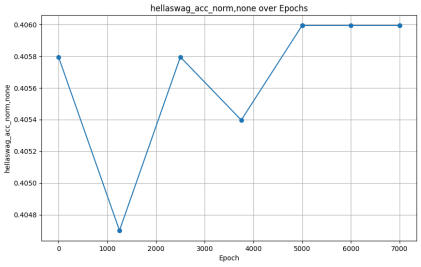


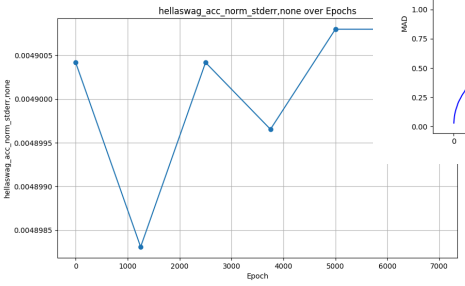




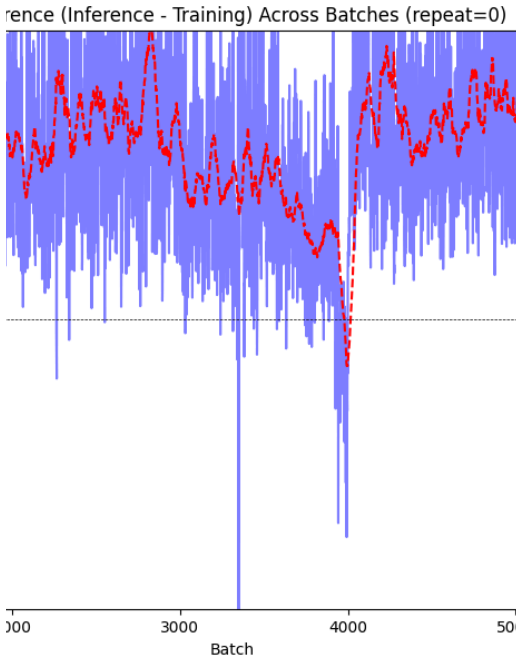


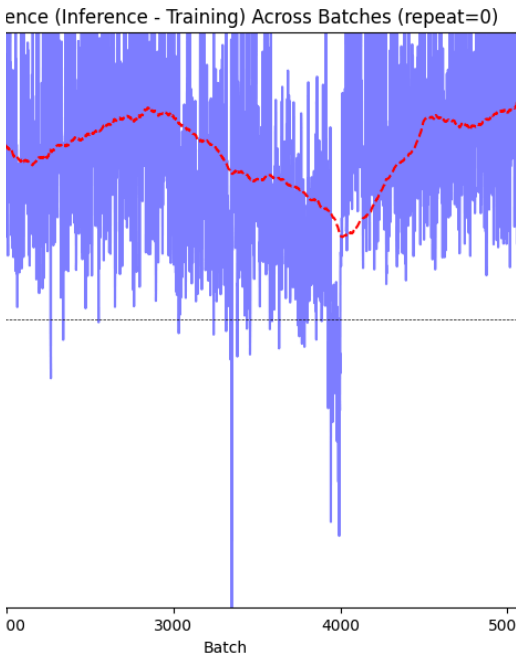


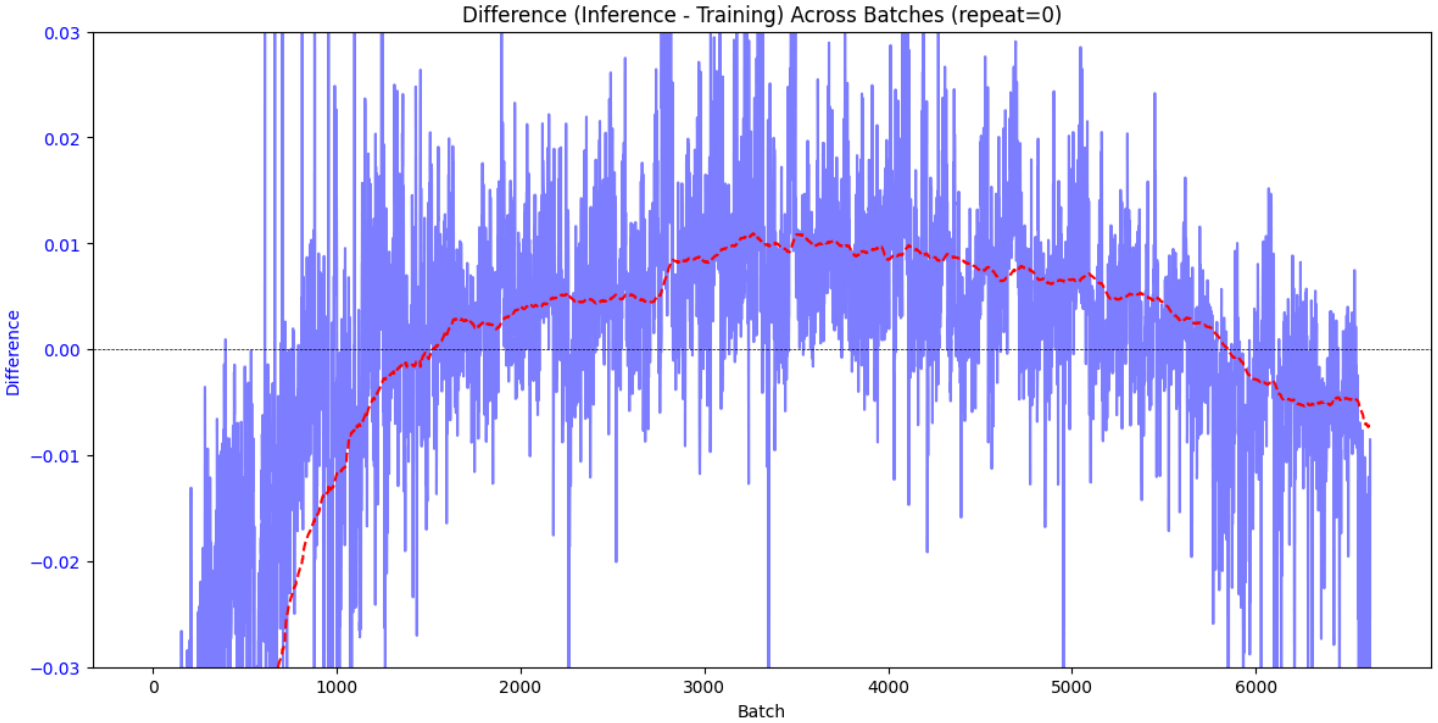




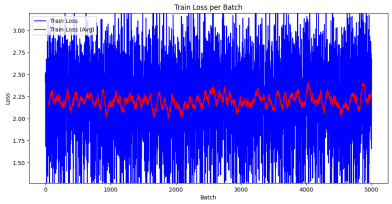
Overall the loss increased and it forgot everything more. Did not remember data in sequential order. Rolling average lies on right plot. Sure it remembers what it just trained on but nothing before. I should retry with the rolling data idea to see if this changes based if I connect it better. If this works it should look like the graph on the right except without the huge rolling average.

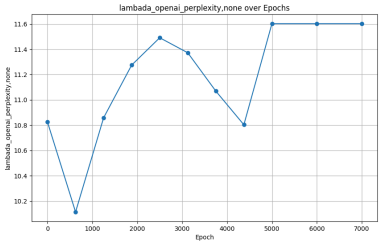


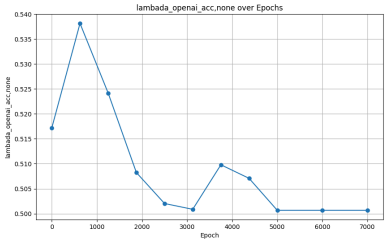


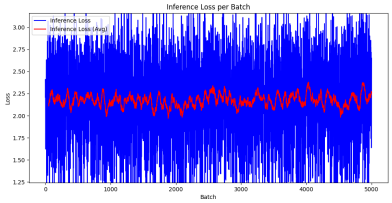


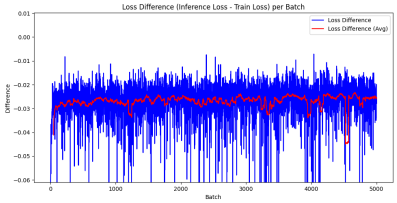
Repeated pile train with higher learning rate at 5e-6
Pretty similar metrics. Higher loss values. Training loss
Is about double the size of inference. Mad values
Are an order of magnitude higher. LLM eval is replicated almost.
Pretty low damage to the model overall. Small change but negligible since
This is a much higher learning rate than the previous. Good to use!

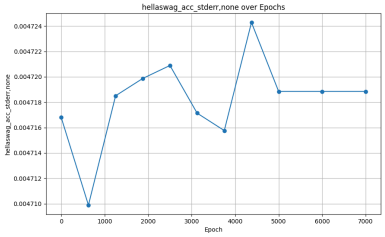


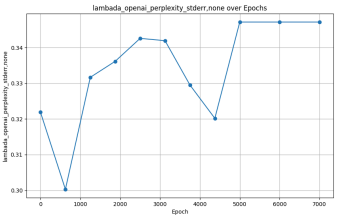


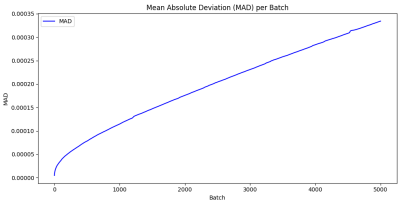


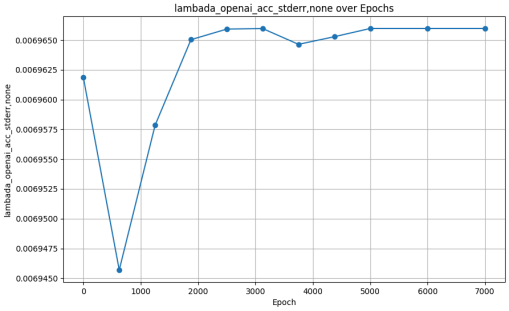


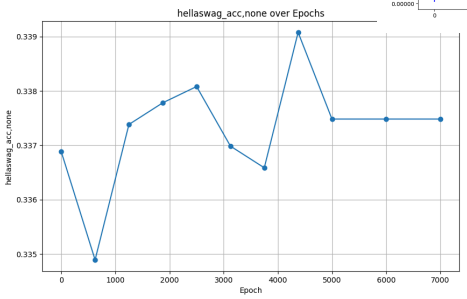


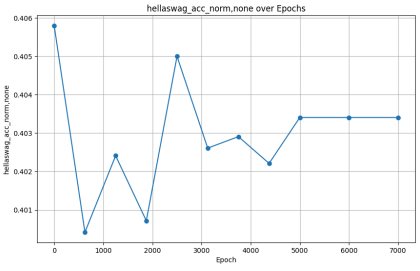


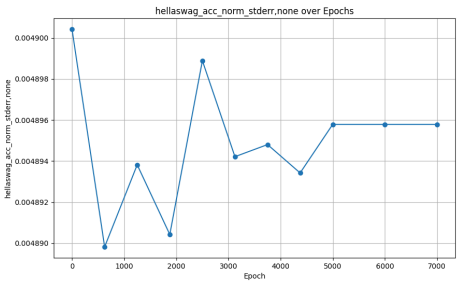












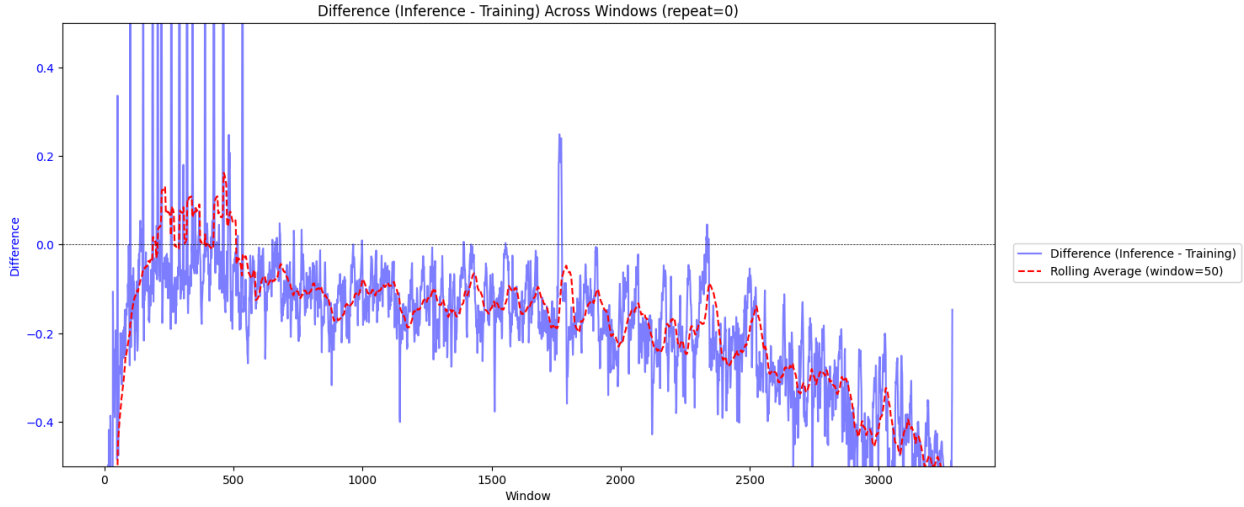
These results show the desired effect. MAD value increase but collapse not happening. Unlike before using the pile data's mad value are no longer correlated to collapse metrics.

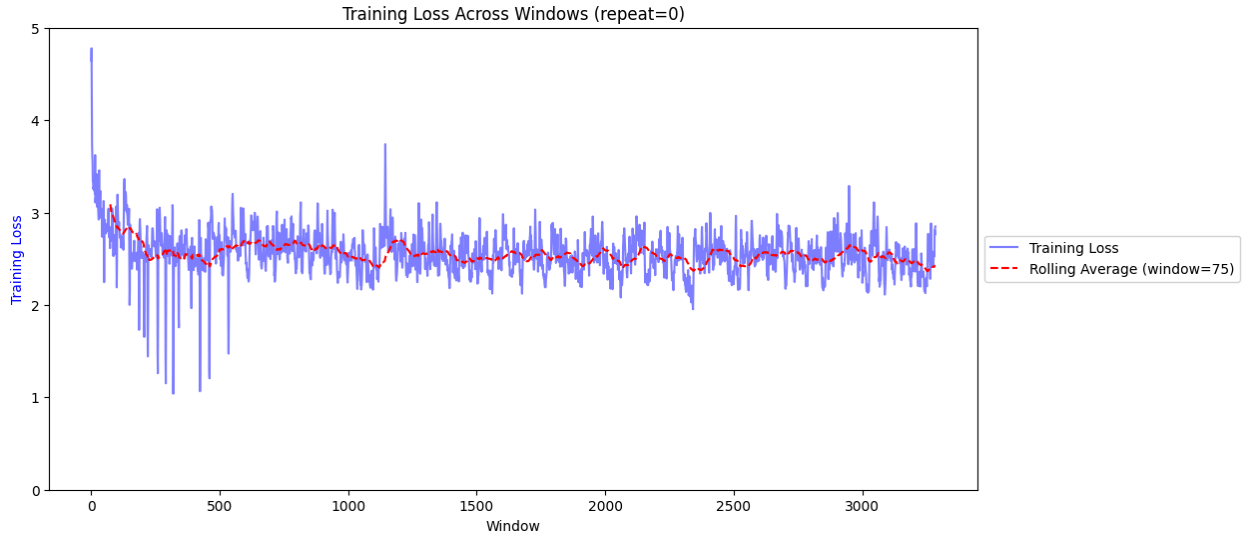
Rolling window training.

Batch size = 1
But inside a single one is a rolling window of multiple utterances. The goal of this is to try and get a time dependent memory.

I will need to have it forget the data and then training unlocks lower values on future untrained ones. This would imply it remembered where in the training sequence it was.

Rolling window 2: step_size = 1500, window size = 4096, lr 5e-6

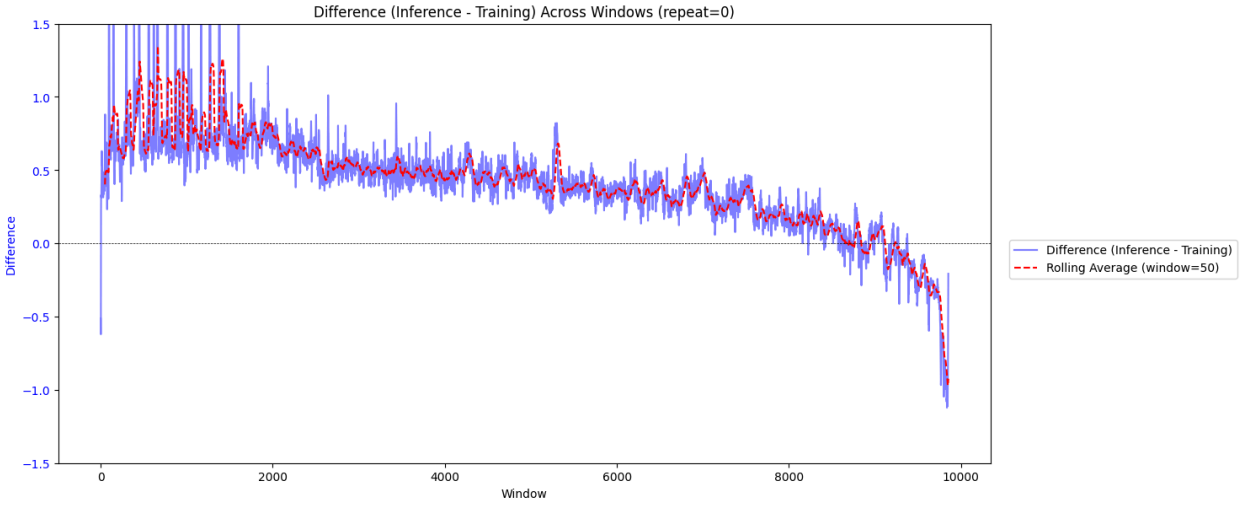


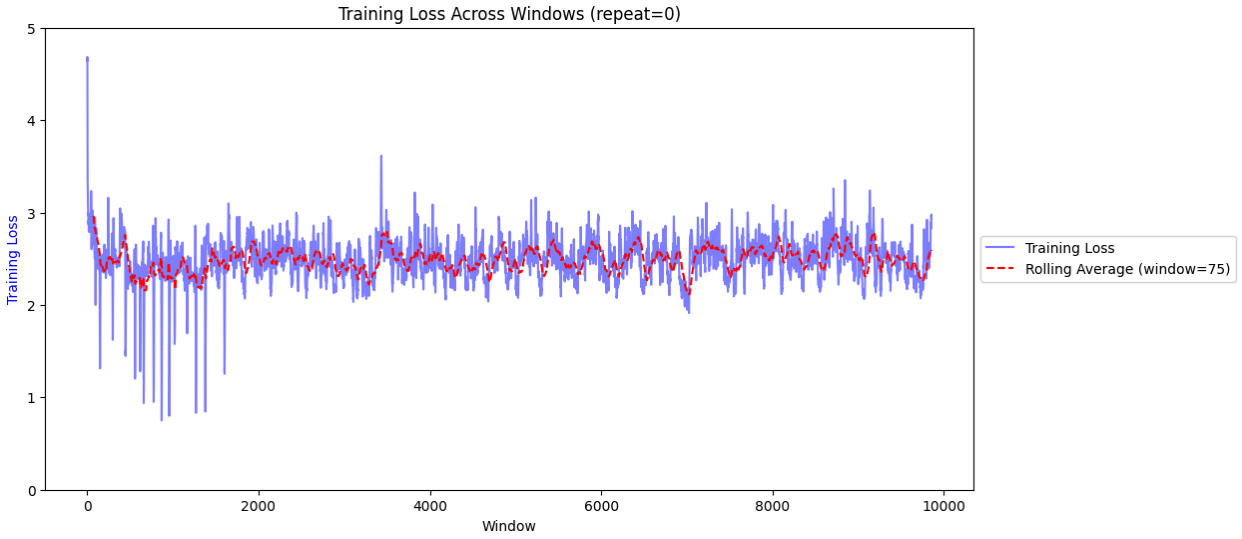


It's pretty weird, different than the other results? Behaving opposite? Perhaps issue in inference? Training loss stayed steady but somehow inference loss got a lot better??
Also it forgets the beginning but the rest it remembers? The end it remembers even better?
Not sure how to interpret this. Now I wish I had the inference loss per training point. The training loss is also very high. Similar to the pile training in loss value.

Rolling_window_1 LR 5e-5, step size 500. Is this what I wanted to see originally? Remembers earlier events and forgets older ones in a decreasing fashion. It sure seems like it.

Great Success!? Now I want to see what happens when in retrain in the middle



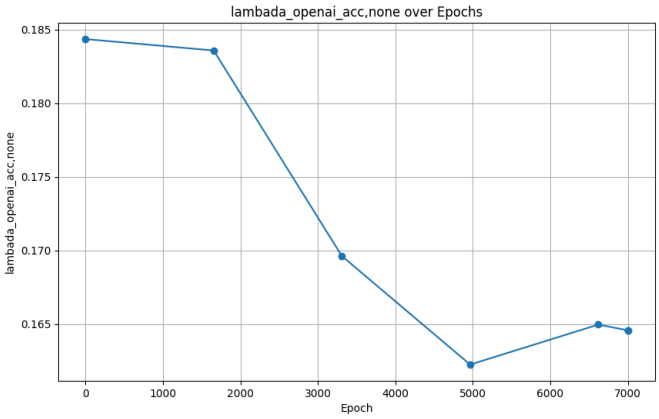


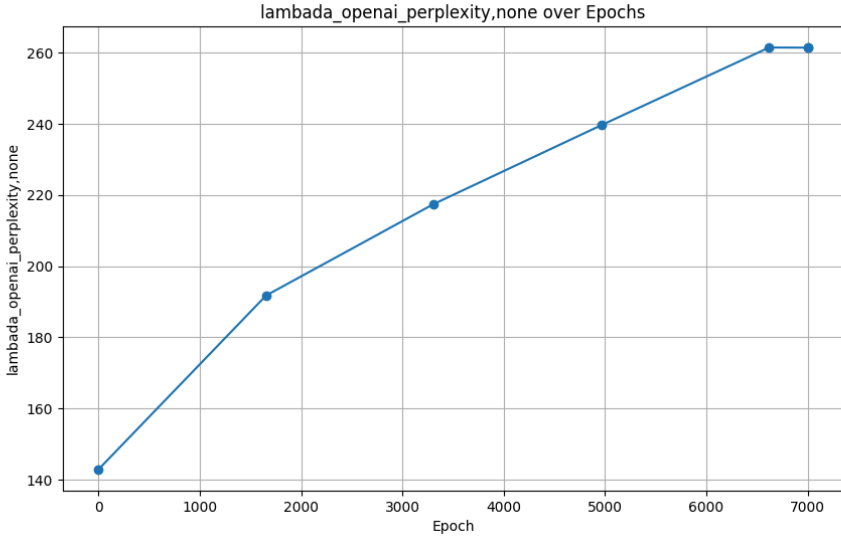
8-5-24:

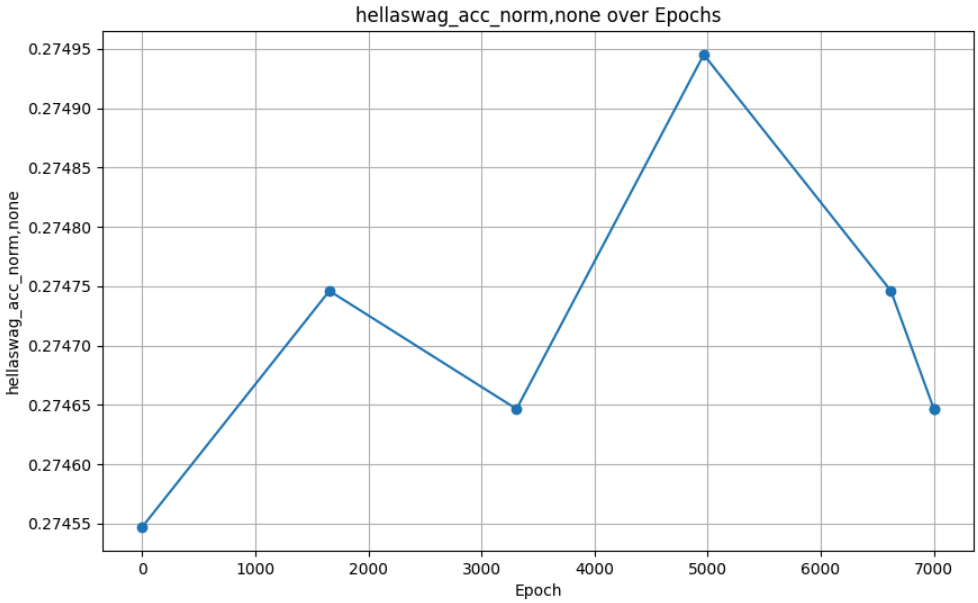
Model Collapse on DND training

Looked at llm eval on the 1e3 and 1e7 lr of standard dnd training (not rolling window)

1e7 clear collapse signal. Wish I did this with the 410m model as 70m has weird results but it works. **Original 70m model has a 0.185 lambada acc score.** Not including all graphs, hellaswag is happy as usual though.

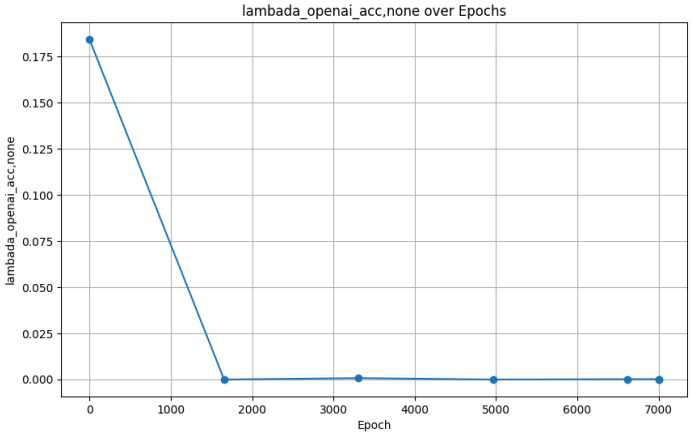


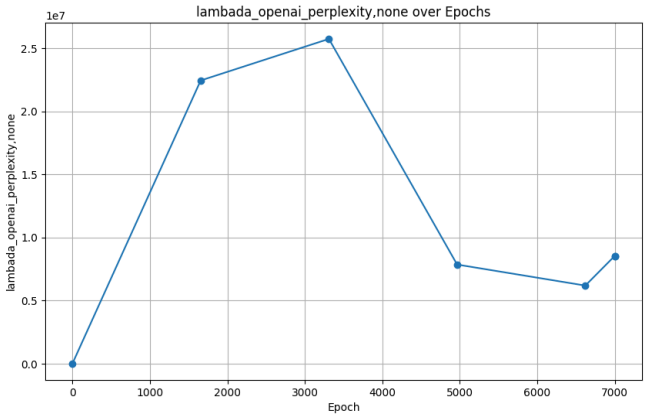


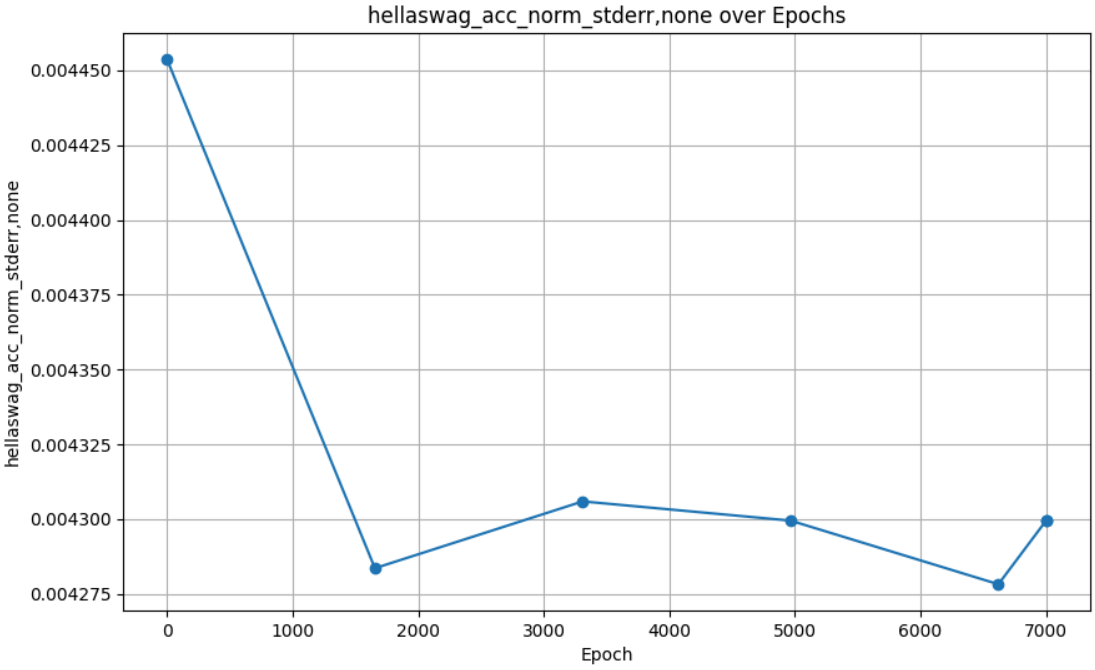


1e-3: Hellaswag and lambada collapse massive collapse metrics

Basically a complete failure of the eval metric.

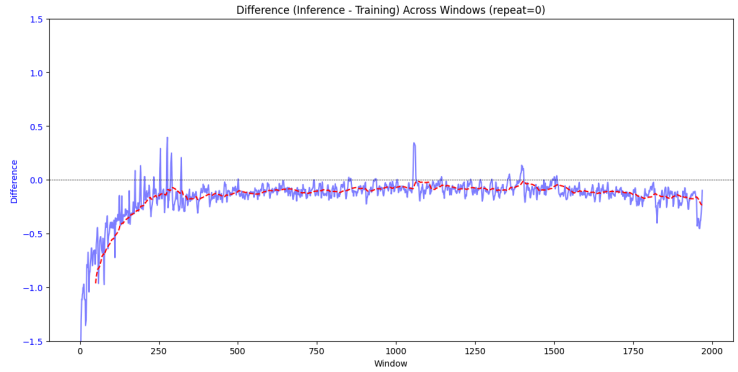


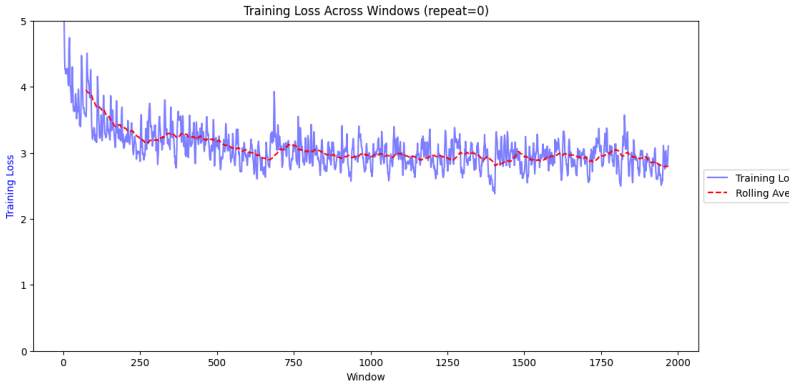




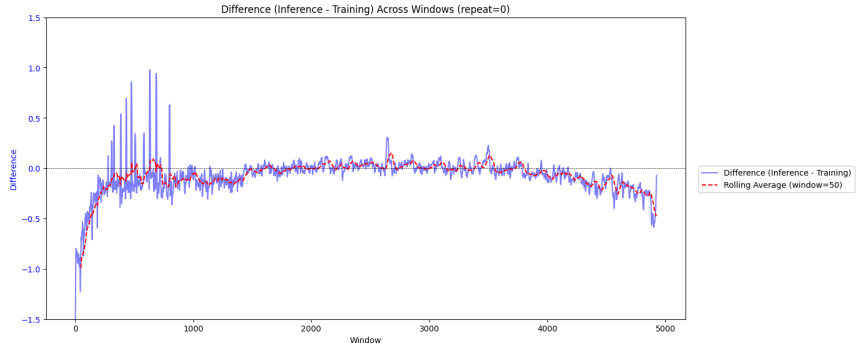
Using pythia 70m for speed increase reran this training system.

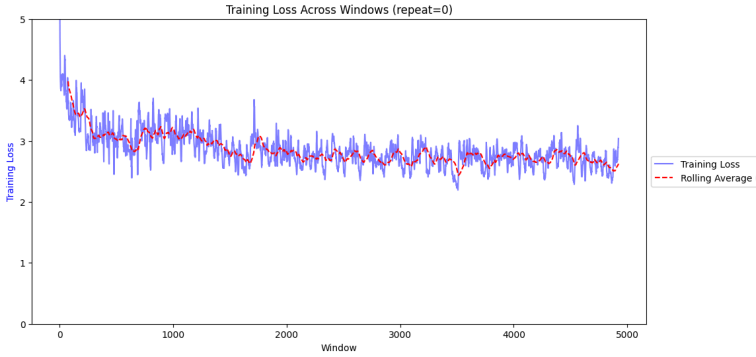
"experiment_name": "rw_6",
"fixed_learning_rate": 5e-05,
"window_size": 8192,
"step_size": 2500,



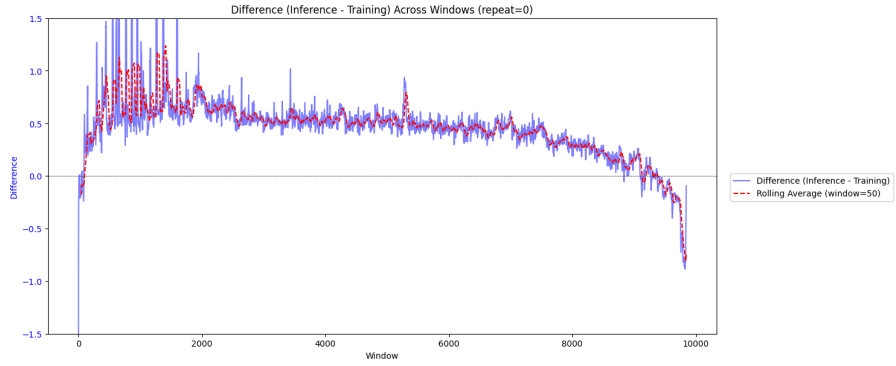


"experiment_name": "rw_5",
"fixed_learning_rate": 5e-05,
"window_size": 8192,
"step_size": 1000,





"experiment_name": "rw_4",
"fixed_learning_rate": 5e-05,
"window_size": 8192,
"step_size": 500,



8-8-24
The only way forward may be to implement a CL with the pile. Will preventing collapse help in this sequential memory process? Is it possible that the model is collapsing so hard it forgets everything from before? Keeping a structure in place via the pile CL may allow sequential memory to be placed without destruction. I was thinking this could be the case but I really wished I would see at least a tiny bit of memory signal.

Fighting pile data errors

9-9-24: training working finally
Results still show collapse but it is clearly mitigated by the 50% pile data.
Original eval results:

Tasks	Version	Filter	n-shot	Metric		Value		Stderr
hellaswag	1	none	0	acc	↑	0.3574	±	0.0068
		none	0	acc_norm	↑	0.4152	±	0.0070
lambada_openai	1	none	0	acc	↑	0.5130	±	0.0071
		none	0	perplexity	↓	10.8462	±	0.3278

Combined_training_1, Window size, 2049, step size 512
CL 50% pile + 50% RW on 410m model. Warmup LR to 1e-6


```
  "hellaswag": {
    "acc,none": 0.30830511850229037,
    "acc_stderr,none": 0.004608495469860368,
    "acc_norm,none": 0.34993029277036447,
    "acc_norm_stderr,none": 0.004759729267943182
  "lambada_openai":
    "perplexity,none": 38.193907232689114,
    "perplexity_stderr,none": 1.3246606666931486,
    "acc,none": 0.29225693770619054,
    "acc_stderr,none": 0.006336249908388209
```


Combined_training_2
50/50, 410m, warmup LR to 1e-5, Window size, 2049, step size 512


```
  "hellaswag": {
    "acc,none": 0.31607249551882094,
    "acc_stderr,none": 0.004639913709615945,
    "acc_norm,none": 0.35839474208325034,
    "acc_norm_stderr,none": 0.004785488626807579
  "lambada_openai":
    "perplexity,none": 37.7598530011878,
    "perplexity_stderr,none": 1.3707541608385199,
    "acc,none": 0.3011837764409082,
    "acc_stderr,none": 0.006391596488933436
```

These are the rolling window studies just done on the retraining. Clear large collapse of all.

RW_1:
"fixed_learning_rate": 5e-05, "window_size": 4096, "step_size": 500, 410m
"hellaswag":
"acc,none": 0.2619000199163513,
"acc_stderr,none": 0.00438769952585491,
"acc_norm,none": 0.25980880302728543,
"acc_norm_stderr,none": 0.00437633345190981
"lambada_openai":
"perplexity,none": 403136.6908342035,
"perplexity_stderr,none": 33708.847190732,
"acc,none": 0.012225887832330681,
"acc_stderr,none": 0.001531020818762761

RW_2:
"fixed_learning_rate": 5e-05,"window_size": 4096,"step_size": 1500, 410m
"hellaswag":
"acc,none": 0.2748456482772356,
"acc_stderr,none": 0.004455240755811604,
"acc_norm,none": 0.29506074487153955,
"acc_norm_stderr,none": 0.004551379838156111
"lambada_openai":
"perplexity,none": 1613.707473138719,
"perplexity_stderr,none": 99.54206330441666,
"acc,none": 0.10110615175625849,
"acc_stderr,none": 0.004200055868668937

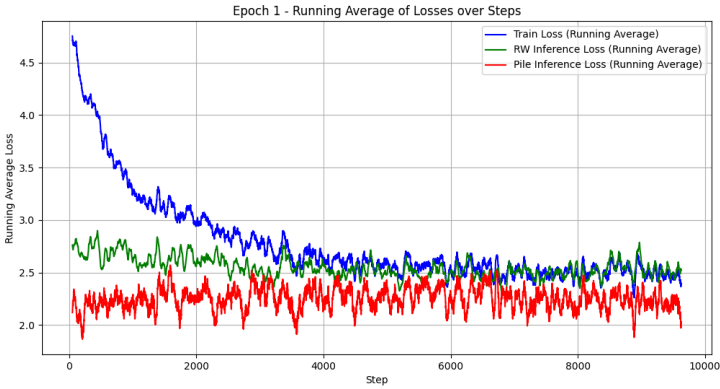
RW_3:
"fixed_learning_rate": 5e-05,"window_size": 4096,"step_size": 1000, 410m
"hellaswag":
"acc,none": 0.2698665604461263,
"acc_stderr,none": 0.004429831152914696,
"acc_norm,none": 0.2813184624576778,
"acc_norm_stderr,none": 0.004487235657955701
"lambada_openai":
"perplexity,none": 5104.269939816932,
"perplexity_stderr,none": 341.0635109999814,
"acc,none": 0.07723656122647002,
"acc_stderr,none": 0.003719364375534321

Loss values at the end of training are around the same. ~ 2-2.5 for rolling window inference. Pile massively helped prevent collapse of this. Next test is to do the same but on different ratios of pile and window data. I need a pareto front when it doesn't collapse.

8-12-24: Analysis of pile training and new adaptive training ratios.

Combined_training_1


```
  "hellaswag":
    "acc,none": 0.30830511850229037,
    "acc_stderr,none": 0.004608495469860368,
    "acc_norm,none": 0.34993029277036447,
    "acc_norm_stderr,none": 0.004759729267943182
  "lambada_openai":
    "perplexity,none": 38.193907232689114,
    "perplexity_stderr,none": 1.3246606666931486,
    "acc,none": 0.29225693770619054,
    "acc_stderr,none": 0.006336249908388209
```

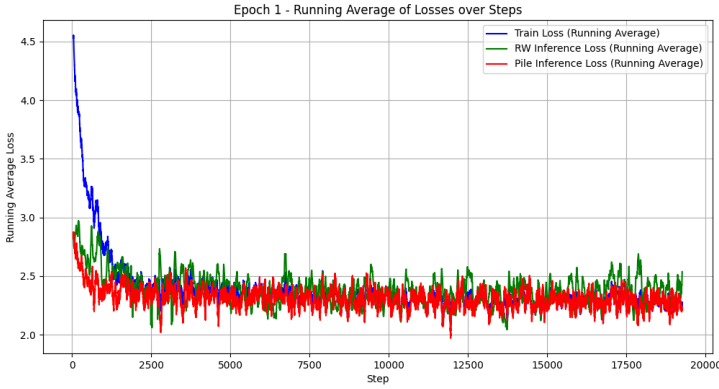
Combined_training_2


```
  "hellaswag":
    "acc,none": 0.31607249551882094,
    "acc_stderr,none": 0.004639913709615945,
    "acc_norm,none": 0.35839474208325034,
    "acc_norm_stderr,none": 0.004785488626807579
  "lambada_openai": {
    "perplexity,none": 37.7598530011878,
    "perplexity_stderr,none": 1.3707541608385199,
    "acc,none": 0.3011837764409082,
    "acc_stderr,none": 0.006391596488933436
```


Find it interesting the loss only overlaps in the higher learning rate setup (2). Perhaps the other one would have done that regardless over time. However it's also interesting that everything seems better in the 1e-5 regime. This was also somehow true in the lm-eval for some reason.

Combined_training_4
"step_size_percentage": 25, "percentage_window": 25, "max_tokens": 4098,

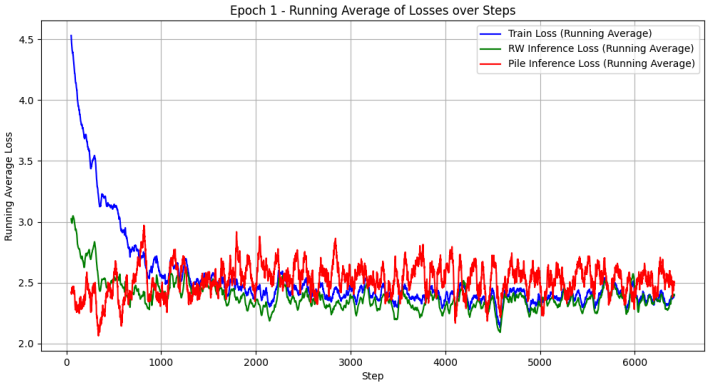
"hellaswag":
"acc,none": 0.3174666401115316,
"acc_stderr,none": 0.0046453934776806765,
"acc_norm,none": 0.36347341167098185,
"acc_norm_stderr,none": 0.0048001644342332474
"lambada_openai":
"perplexity,none": 35.451164158174635,
"perplexity_stderr,none": 1.277142369377081,
"acc,none": 0.3137977876964875,
"acc_stderr,none": 0.00646491871092863



So even with 75% pile data collapse still happens just less. May want to train just a pure pile for comparison as the 70m was confusing results.

Combined_training_5
"step_size_percentage": 25,"percentage_window": 75,"max_tokens": 4098,

"hellaswag":
"acc,none": 0.3148775144393547,
"acc_stderr,none": 0.004635178371110076,
"acc_norm,none": 0.3536148177653854,
"acc_norm_stderr,none": 0.004771143074426129
"lambada_openai":
"perplexity,none": 53.919043108367006,
"perplexity_stderr,none": 2.1197750359977516,
"acc,none": 0.26062487871143025,
"acc_stderr,none": 0.006115788029333527



Collapse seems to be another long slope. To get the exact collapse it seems you have to train on diminishing returns of pile vs rw data. Not good!

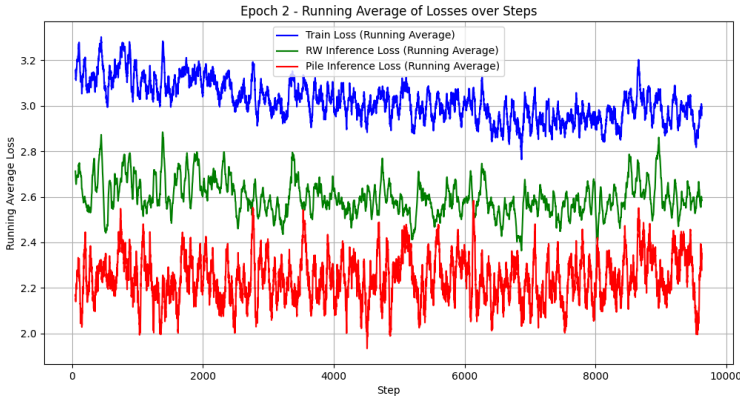
Combined Train 3 (2 epochs)
"starting_learning_rate": 1e-07, "step_size_percentage": 25, "percentage_window": 50,

Epoch 1:

"hellaswag":
"acc,none": 0.3229436367257518,
"acc_stderr,none": 0.004666457279979418,
"acc_norm,none": 0.37870942043417644,
"acc_norm_stderr,none": 0.0048407422067180
"lambada_openai":
"perplexity,none": 15.790000263555177,
"perplexity_stderr,none": 0.4965757444498425
"acc,none": 0.444401319619639,
"acc_stderr,none": 0.00692277705453026

Epoch 2:

"hellaswag":
"acc,none": 0.31686914957179846,
"acc_stderr,none": 0.004643050902503901,
"acc_norm,none": 0.37054371639115713,
"acc_norm_stderr,none": 0.004819633668832542
"lambada_openai":
"perplexity,none": 21.508709305012985,
"perplexity_stderr,none": 0.6897308148112494,
"acc,none": 0.38695905297884725,
"acc_stderr,none": 0.006785616642963463



So very slow training prevents collapse as expected. Makes me wonder if I continued to train the previous tests would the eval metrics go back up over time ever? Perhaps after it learned the new data it would relearn the old?

Maybe 10 epochs on the best method of 1 epoch training above.

9-16-24: 10 epochs

Interesting results. Seems that the rw data is easier to learn so it optimizes for it, reducing loss but at the same time damaging the pile performance and eval metrics. I imagine this is exactly what the model is doing on the pile data itself as well. Over Optimizing on the easy results ignoring the harder ones.

So could I massively improve the pile model by first inference and sorting the data by difficulty?

9-26-24: Thoughts:
So what if CL is already solved by methods such as EWC the core issue is that we need a dream state to reset it?

As in EWC has the problem of eventually not learning anything knew because everything is frozen. What if that actually is the long term solution to CL.

That would mean the only thing we need in the next iteration is something like dreaming and sleep where the brain prunes and “resets” the brain for the next day.

It’s unknown how this pruning step would work but if EWC works well or its related work then perhaps all that’s needed is a consolidation mechanism at the end when it’s “full”.