A HYPERPARAMETERS

Parameter	Value
Learning rate	2e-5
Adam warmup steps	150
Batch size	16
Adam ϵ	1e-6
Adam β_1	0.9
Adam β_2	0.98
Weight decay	0.01

Table 9: Transformer hyperparameters

Transformer models were fine-tuned for a fixed 1,000 steps, evaluating on the dev set every 100 steps and using the checkpoint with best dev accuracy to obtain the final test set accuracy. We found from preliminary experiments that the hyperparameters in Table 9 generally worked well on all models and datasets. The learning rate followed a sloped triangular schedule, warming up over the first several steps and then linearly decreasing until the end of training. The maximum sequence length for models was set per-dataset based on the length of input texts. In general it was set so that at least 99% of examples fit completely within the sequence length, and those that did not fit were pruned.

For BALD, the number of Monte Carlo dropout samples was set to 5, and we found in preliminary experiments on CSQA and MRP that increasing the number of samples to 20 did not substantially improve results. We performed the Monte Carlo dropout using the normal dropout probabilities in the dropout layers of the models (generally 0.1); increasing the dropout probabilities to 0.5 (just for the BALD step, not model training) added noise due to the large number of dropout layers and actually caused performance to slightly decrease in our preliminary tests.

The experiments were run on servers with Nvidia RTX 2080 Ti and GTX 1080 gpus. We estimate that all reported experiments together represent about 3,000 gpu-hours for an RTX 2080 Ti.

Our experiments are implemented using Pytorch (Paszke et al., 2019) and code is available in the supplementary material. We used version 2.4.1 of the HuggingFace Transformers library (Wolf et al., 2020), which has a small bug in the RoBERTa implementation. Testing two of our datasets with a newer version suggests that this doesn't change our conclusions.

B DATASET SIZES

See Table 10.

C PRUNING LEVEL

For our main experiments we pruned the 50% of examples most likely to be collective outliers, as that threshold is what appeared to work best in the work of Karamcheti et al. (2021). However, as we did not find pruning to be as effective as in that work, we also tried a 25% threshold with just one AL method (BatchBALD) to see if the pruning threshold made a difference. The results are displayed in Table 11. We observe that both the relative gain from AL and the absolute performance fall between the corresponding numbers for 0% and 50% on average, and therefore conclude that while the pruning threshold could be adjusted to trade relative and absolute gains, it would not produce a boost in both at once across our datasets.

D LABEL BUDGET

We primarily focus on a low-budget setting in this work, but it is interesting to consider how performance might improve with more labels. In Table 12 we show the results of experiments with

Dataset	Train	Dev	Test
AGN-SB-c	2000	2500	5000
DBpedia-SB-c	2000	2000	2000
CODAH	2376	100	200
CODAH-c	4752	200	400
CSQA	9141	300	1221
MRP-c	8614	512	1024
MRS-c	9000	500	500
PIQA	14113	1000	1838
PIQA-c	28226	2000	3676
HellaSWAG	39905	1000	8042
HellaSWAG-c	65198	1988	15508
SWAG	65354	4096	20006
AGN-c	100000	5000	10000
SWAG-c	130708	8192	40012
aNLI	150000	1036	1532
aNLI-c	300000	2072	3064

Table 10: Dataset sizes. “Train” is the set we used as the unlabeled pool for active learning. Note also that the numbers add up to slightly less than the official sizes of these datasets, as we held out some additional data that ultimately went unused in this work.

	0% pruned (from Table 1)		25% pruned		50% pruned (from Table 2)	
Dataset \ Method	BatchBALD-MC	Random	BatchBALD-MC	Random	BatchBALD-MC	Random
AGN-c	**87.5 (0.2)**	86.6 (0.2)	**87.0 (0.2)**	86.6 (0.2)	**85.4 (0.3)**	84.6 (0.3)
AGN-SB-c	**97.4 (0.0)**	96.7 (0.1)	**97.4 (0.1)**	96.8 (0.1)	**97.5 (0.1)**	96.9 (0.1)
aNLI	**58.9 (0.2)**	58.8 (0.2)	58.1 (0.1)	58.4 (0.1)	57.4 (0.1)	57.5 (0.2)
CODAH	58.5 (0.3)	59.0 (0.6)	58.9 (0.5)	60.2 (0.5)	**59.2 (0.4)**	58.4 (0.6)
CSQA	**43.9 (0.4)**	43.8 (0.2)	**45.8 (0.3)**	45.2 (0.3)	46.0 (0.2)	46.1 (0.2)
DBPedia-c	**98.9 (0.0)**	98.8 (0.1)	**99.1 (0.0)**	99.0 (0.1)	**99.1 (0.0)**	99.0 (0.0)
HellaSWAG	38.5 (0.2)	38.7 (0.2)	**37.4 (0.4)**	37.2 (0.4)	35.7 (0.5)	35.7 (0.3)
MRP-c	**83.3 (0.2)**	83.0 (0.3)	**82.6 (0.2)**	81.8 (0.2)	**80.2 (0.3)**	79.0 (0.6)
MRS-c	**95.3 (0.1)**	94.0 (0.2)	**95.0 (0.1)**	94.0 (0.1)	**94.9 (0.1)**	94.1 (0.2)
PIQA	55.9 (0.2)	56.8 (0.3)	55.8 (0.3)	56.2 (0.2)	55.7 (0.3)	56.1 (0.3)
SWAG	62.7 (0.2)	63.5 (0.2)	**61.1 (0.2)**	60.7 (0.3)	59.4 (0.5)	59.8 (0.3)
Average	**70.96 (0.07)**	70.88 (0.08)	**70.75 (0.08)**	70.54 (0.08)	**70.06 (0.09)**	69.76 (0.10)

Table 11: Roberta-base with different levels of pruning.

	Q=1000, $\lvert\Delta L\rvert$=50		Q=500, $\lvert\Delta L\rvert$=25 (from Table 1)	
Dataset $\setminus$ Method	BatchBALD-MC	Random	BatchBALD-MC	Random
AGN-c	**88.6 (0.1)**	88.2 (0.1)	**87.5 (0.2)**	86.6 (0.2)
AGN-SB-c	**97.7 (0.1)**	97.3 (0.1)	**97.4 (0.0)**	96.7 (0.1)
aNLI	60.4 (0.2)	60.5 (0.2)	**58.9 (0.2)**	58.8 (0.2)
CODAH	**62.6 (0.4)**	62.1 (0.4)	58.5 (0.3)	59.0 (0.6)
CSQA	**47.8 (0.4)**	47.7 (0.2)	**43.9 (0.4)**	43.8 (0.2)
DBpedia-SB-c	**99.1 (0.0)**	99.1 (0.0)	**98.9 (0.0)**	98.8 (0.1)
HellaSWAG	41.2 (0.1)	41.2 (0.1)	38.5 (0.2)	38.7 (0.2)
MRP-c	**85.6 (0.1)**	84.5 (0.3)	**83.3 (0.2)**	83.0 (0.3)
MRS-c	**95.8 (0.1)**	95.2 (0.0)	**95.3 (0.1)**	94.0 (0.2)
PIQA	**57.7 (0.1)**	57.4 (0.3)	55.9 (0.2)	56.8 (0.3)
SWAG	65.7 (0.3)	65.8 (0.2)	62.7 (0.2)	63.5 (0.2)
Average	**72.92 (0.06)**	72.63 (0.07)	**70.96 (0.07)**	70.88 (0.08)

Table 12: RoBERTa-base AUC results with $Q = 1000$ and $|\Delta L| = 50$. Original results from Table 1 are reprinted for comparison. AL has higher relative performance on the results with greater label budget and batch size.

budget Q=1000 and batch size[4] $|\Delta L|$=50. The results show that a higher budget generally results in better relative AL performance. This result is encouraging, as it suggests the instability of AL-selected datasets is mitigated simply by collecting more labels. Of course, it does not help in cases where the labels are costly and the budget cannot easily be increased, so an interesting question for future work is how to determine the minimum viable label budget needed for AL to become effective.

---

[4] Unfortunately we cannot avoid changing multiple variables here, as either the batch size or the number of batches must increase when we use a larger budget. Due to our earlier batch size ablation we consider it unlikely that the batch size plays a large role, and generally attribute the results here to the budget increase.