Comparison of uncertainty sampling and sensitivity analysis in active learning for feed forward neural networks

Sebastien Meniere

Stellenbosch University
Stellenbosch, South Africa
26484765@sun.ac.za

***Abstract*—This report compares active learning with uncertainty sampling (ALUS) and sensitivity analysis for selective learning (SASLA) against a passive fixed-set learning for shallow feed-forward neural networks trained with stochastic gradient descent (SGD). A compute focused view is adopted with the primary budget being the number of backpropagated instances. Across six datasets (Table I) hyperparameters are tuned by grid search (Table II) and mean $\pm$ SD over five seeds are reported. Results (Table III, Figures 1–6) show no uniform winner. ALUS excels on a 1-D sinusoid where predictive variance is highly diagnostic, but underperforms on Friedman1 and is roughly on par with passive on MNIST and Energy. SASLA matches or slightly improves over passive on Banknote and Iris, but trails on Energy. These outcomes are explained by the trade-off between (i) spending compute on finding informative points and (ii) spending compute on optimizing model parameters. The evidence supports a "no free lunch" conclusion: benefits depend on data, model bias, and how selection overhead is budgeted.**

I. Introduction

Active learning aims to reduce training effort and generalization by prioritizing informative examples instead of treating all data equally. In the context of shallow feed-forward neural networks trained with SGD, this prioritization competes directly with optimization compute. Where time spent identifying informative samples is not spent updating weights. Therefore pool-based uncertainty sampling (ALUS) and sensitivity-based selective learning (SASLA) is studied under a shared compute budget measured in backpropagated instances.

ALUS iteratively trains a model, scores the unlabeled pool with an uncertainty utility, and acquires the most uncertain items; for regression an ensemble is used with a predictive variance utility. SASLA starts from the fully labeled training set and periodically reconstructs a training subset using a gradient-based sensitivity score. These strategies differ in starting condition and in how they construct or acquire informative instances, leading to different compute profiles and generalization.

The experimental design spans three classification and three regression problems (Table I) using shallow architectures tailored per dataset. Hyperparameters for the optimizer and for each strategy are chosen by grid search under the same budget (Table II). Performance is evaluated with macro-F1 (classification) and R^2 (regression), averaged over five seeds. Learning curves as a function of budget complement final summaries (Table III, Figures 1–6).

Shallow neural networks & stochastic gradient descent: A *shallow* neural network (single hidden layer) maps an input $\underline{x} \in \mathbb{R}^d$ to an output by

$$\hat{\underline{y}} \;=\; g(W_2\,\sigma(W_1\underline{x} + \underline{b}_1) + \underline{b}_2)\,,$$

where W_1, W_2 and $\underline{b}_1, \underline{b}_2$ are trainable weights, $\sigma(\cdot)$ is a hidden layer activation, and $g(\cdot)$ is the output activation. Parameters are learned by SGD with backpropagation. Gradients of the loss with respect to all weights are computed and used to update the parameters. SGD adjusts the weights in the direction that reduces the loss in a small batch. The main hyperparameters are the learning rate (LR), momentum, and regularization parmeter weight decay (WD). Throughout this work, computational cost is approximated by the number of backpropagated instances.

Passive versus active learning: All models used disjoint train/validation/test splits. Passive fixed-size learning (FSL) trains on a fixed labeled set and updates weights with every available training example. In contrast, active learning aim to select training points that are deemed most informative for generalization. (i) **ALUS:** the model is trained iteratively, all candidates in $D_{\text{unlabeled}}$ are scored by an uncertainty utility, and the most uncertain points(s) are added to D_{labeled}. For classification, utilities are least-confidence, margin, or entropy; for regression/function approximation, uncertainty is computed from predictive variation. (ii) **SASLA:** starting from the full training set, instances are periodically added or removed from the active training set using a sensitivity score derived from output gradients. In both cases the calculation of the utility or sensitivity trades part of the compute budget for more informative updates.

Uncertainty-based active learning: This work uses (ALUS) as the active learning strategy. The training set D_{train} is partitioned into a labeled pool D_{labeled} and an unlabeled pool $D_{\text{unlabeled}}$. At each iteration the learner computes a utility score for every $\underline{x}_i \in D_{\text{unlabeled}}$ selecting the most informative

instance(s), adding them to the D_{labeled} and then retraining on the expanded set. The retrain cycle repeats until the query budget B is exhausted. Algorithm 1 summarizes the generic procedure. When $\underline{\theta}_{\text{models}}$ is a single classifier $\theta_{\text{classifier}}$, the Algorithm 1 recovers ALUS for classification; when $\underline{\theta}_{\text{models}}$ is an ensemble, it produces the regression variant described below.

Algorithm 1 Pool-based uncertainty sampling (general form)

1: **Input:** $D_{\text{unlabeled}}$, D_{labeled}, $\underline{\theta}_{\text{models}}$, B
2: **repeat**
3: **for all** $(\underline{x}_i, ?) \in D_{\text{unlabeled}}$ **do**
4: compute utility$(\underline{x}_i; \underline{\theta}_{\text{models}})$
5: **end for**
6: choose $\underline{x}^* \leftarrow \arg\max_{\underline{x}_i \in D_{\text{unlabeled}}} \text{utility}(\underline{x}_i; \underline{\theta}_{\text{models}})$ and query y^*
7: $D_{\text{labeled}} \leftarrow D_{\text{labeled}} \cup \{(\underline{x}^*, y^*)\}$
8: $D_{\text{unlabeled}} \leftarrow D_{\text{unlabeled}} \setminus \{(\underline{x}^*, ?)\}$
9: train $\underline{\theta}_{\text{models}}$ on D_{labeled} (from scratch or by warm-start)
10: $B \leftarrow B - 1$
11: **until** $B = 0$

a) Classification: For classification the utility function is a measure of predictive uncertainty. Standard choices include entropy, least confidence (one minus the maximum class probability), and minimum margin. The minimum-margin criterion is used as the standard utility in this report. Minimum-margin selects the instance whose top two class probabilities are closest:

$$\underline{x}^* = \arg \min_{\underline{x}_i \in D_{\text{unlabeled}}} \Big(P_\theta(y_m \mid \underline{x}_i) - P_\theta(y_n \mid \underline{x}_i) \Big), \quad (1)$$

where y_m and y_n are, the most probable and second most probable class labels under the current model $\theta_{\text{classifier}}$. Intuitively, for K classes, $\underline{x}_i$ is maximally uncertain when $P_\theta(y_k \mid \underline{x}_i) \approx 1/K$ for all $k \in \{1, \ldots, K\}$. Substituting (**??**) for `utility` in Algorithm 1 and letting $\theta_{\text{classifier}}$ be a shallow neural network with K output units yields the ALUS procedure used in our experiments.

Several practical knobs are implicit in Algorithm 1: (i) the initial size of D_{labeled}, (ii) the number of points acquired per iteration, and (iii) the training stop rule for $\theta_{\text{classifier}}$ after each acquisition. Each trades compute for improved new instance quality. Note that line 9 does not need to retrain from scratch instead warm starting from the previous weights can be used to avoid wasting early epochs just to 'catch up' with the previous loss.

b) Regression: For regression algorithm 1 is changed by letting $\underline{\theta}_{\text{models}}$ be an ensemble of M shallow neural networks. Each model produces a prediction $\hat{y}_m(\underline{x}_i)$ for every candidate $\underline{x}_i \in D_{\text{unlabeled}}$. The utility of $\underline{x}_i$ is the dispersion of ensemble predictions. Here, the sample standard deviation:

$$\text{utility}(\underline{x}_i; \underline{\theta}_{\text{models}}) \;=\; \text{SD}\left(\{\hat{y}_m(\underline{x}_i)\}_{m=1}^{M}\right).$$

The algorithm adds the instance(s) with the largest predictive standard deviation to each model's labeled set before retraining the ensemble. Two practical aspects follow from this design. First, as iterations proceed the labeled sets of each of the ensembles become increasingly similar. Ensemble predictions tend to agree resulting variance signal becoming stale and weakening selection quality. Diversity can be sustained by injecting randomness so that member prediction remain varied. Second, training M networks multiplies the backpropagation cost per iteration. Thus, US for regression is expected to have the following trade off: too few models decrease the value of the uncertainty signal; too many will inflate compute with diminishing returns.

Sensitivity analysis for selective learning: SASLA reconstructs an active training set from a fixed fully labeled training set. A basic sketch of the algorithm is given in 2. The aim is to concentrate updates on informative patterns. Following **?**, informativeness for a pattern p is defined via the sensitivity of the NN's output to small input perturbations. Let

$$S_{oz}^{(p)} \;=\; \frac{\partial \mathbf{o}}{\partial \mathbf{z}}\Big|_{\mathbf{z}=\underline{x}_p}$$

be the output to input Jacobian at $\underline{x}_p$ (inputs $\mathbf{z}$, outputs $\mathbf{o}$). SASLA ranks patterns by a norm of this Jacobian. For classification, the paper proposes

$$\phi(p) \;=\; \big\|\tilde{S}_o^{(p)}\big\| \;\text{ with }\; \big\|\tilde{S}_o^{(p)}\big\| \;=\; \max_k \big\|S_{oz;k}^{(p)}\big\|_q, \quad q \in \{1, 2\},$$

which is to take an ℓ_q-norm across input dimensions per output unit k, then the maximum over outputs.

SASLA operates in selection intervals, which is set to one in the **?** work. Let D_C denote the full candidate set, D_T the current training subset, $F(\cdot; W)$ the NN mapping from input to output. At a selection interval, compute $\phi(p)$ for all $p \in D_C$, let $\bar{\phi}$ be the mean over D_C, and keep the patterns whose informativeness exceeds a data-dependent threshold:

$$A(D_C, F(\cdot; W)) \;=\; \big\{\, p \in D_C \;:\; \phi(p) > \tau \,\big\}$$

Here $\tau \;=\; (1 - \beta)\,\bar{\phi}$ and $\beta \in [0, 1]$ controls the threshold. Larger β admits more patterns; $\beta = 1$ degenerates to fixed-set learning (FSL), so selection should be bypassed in implementation.

Algorithm 2 SASLA (resampling over a fixed labeled pool)

1: **Input:** full pool D_C, initial subset $D_T \leftarrow D_{subset_0}$, network $F(\cdot; W)$, threshold control β, selection interval M
2: **repeat**
3: Train $F(\cdot; W)$ on D_T for M epochs (or until a local termination on D_T)
4: For all $p \in D_C$: compute $\phi(p)$; compute $\bar{\phi} = \frac{1}{|D_C|} \sum_{p \in D_C} \phi(p)$
5: Set $\tau = (1 - \beta)\,\bar{\phi}$; define $D_{subset} = \{p \in D_C : \phi(p) > \tau\}$
6: Update $D_T \leftarrow D_{subset}$
7: **until** a global stopping criterion is met

Algorithmic sketch:

Classification: The classification domain uses the Jacobian-based informativeness above. Intuitively, patterns near decision boundaries exhibit larger $\phi(p)$ because small input perturbations produce large output changes, so emphasizing such patterns focuses updates on boundary refinement.

Regression (adaptation used in this report): The original SASLA definition targets classification via output-sensitivity. For regression, SASLA is modified to rank patterns by a scalar sensitivity score derived from the backpropagated gradients

$$\phi_{\text{reg}}(p) \;\propto\; \left\|\nabla_{\mathbf{z}} L\big(F(\underline{x}_p; W),\, y_p\big)\right\|_q$$

with L a regression loss like mean squared error (MSE).

Practical notes: There is a three-way trade-off among the selection interval M, the threshold control β, and compute. Shorter intervals and smaller β increase selection overhead but shrink the training subset and allow faster training steps. Longer intervals or $\beta \to 1$ reduce selection overhead but act more like FSL. In implementation, sensitivities should be computed over all candidates D_C at each interval, and when $\beta = 1$ the selection step should be skipped to truly recover FSL (no extra selection cost).

IMPLEMENTATION

The study was implemented in Python using standard scientific libraries. Shallow feed-forward neural networks with a single hidden layer were built in PyTorch; preprocessing and metrics relied on sklearn for scaling, splitting, and evaluation; analysis and plotting were done in a notebook environment.

Model and optimizer: All models were shallow MLPs with one hidden layer and sigmoid hidden activations. Task specific output activations are a single logit with sigmoid for binary classification. Softmax over the K logits for multiclass. Linear output(s) for regression, matching the output dimension. Parameters were learned with SGD using learning rate, momentum, and weight decay as the only optimizer hyperparameters. Training proceeded in mini-batches with backpropagation with task specific loss functions: binary cross-entropy with logits for binary classification. Cross-entropy for multiclass classification. Mean-squared error for regression.

Compute accounting: A compute tracker enforced a shared budget across regimes. Counting the number examples used in backpropagation. All training, evaluation, and selection steps queried the tracker before work and recorded usage afterward. Unless stated otherwise, the backprop budget counted per model: a minibatch update across an M-member ensemble consumed M times the examples of a single update.

Data and preprocessing: Six datasets were supported: Iris, Banknote Authentication, MNIST, and Sine-1D, Friedman1, Energy Efficiency summaries shown in table I. Data were split into disjoint train/test partitions, with stratification for classification. Features were standardized by statistics computed on the training split and applied to test. MNIST pixels were scaled to $[0, 1]$ prior to standardization. Regression targets were scaled between $[0, 1]$ to accommodate the sigmoid activation in the hidden layer.

Problem	**Task**	**Attr. types**	**Missing**	**Class / Range**	**NN arch.**	**Source**
Iris	class.	cts.	none	setosa – 33.3% versicolor – 33.3% virginica – 33.3%	4-16-3	**?**
Banknote	class.	cts.	none	authentic – 44.6% fake – 55.4%	4-16-2	**?**
MNIST	class.	cts.	none	0 – 9.8% 1 – 11.2% 2 – 9.9% 3 – 10.2% 4 – 9.8% 5 – 9.0% 6 – 9.8% 7 – 10.4% 8 – 9.7% 9 – 10.2%	784-32-10	**?**
Energy	reg	cts.	none	[6.0, 48.0]	8-16-2	**?**
Sine wave	reg	cts.	none	$[-1, 1]$	1-16-1	synthetic
Friedman1	reg	cts.	none	[0, 10]	10-16-1	**?**

TABLE I
CHOSEN DATASETS. FOR CLASSIFICATION PROBLEMS, CLASS DISTRIBUTIONS ARE SHOWN; FOR REGRESSION, THE RANGE OF TARGETS.

Learning strategies: Three regimes were implemented under the same budget policy. *Passive learning* trained on the full labeled training. *ALUS* for classification initialized from a small, stratified labeled seed, trained to an inner stop, scored the unlabeled pool by probability margin, acquired the lowest-margin items, and repeated until the query or compute budget was exhausted. For regression ALUS used an ensemble of shallow networks. Pool utility was the sample standard deviation across the members. Acquisitions were shared to all members. Training used warm-start, initializing the weights of the new model to the weigths found in before acquisition of the new instances. *SASLA* maintained a dynamic training subset drawn from a full training set. Per-sample sensitivities were computed by backpropagated gradients. A threshold based on the mean sensitivity and a β controlled subset construction. The model was retrained/continued to train on the selected subset at a fixed selection interval.

METHODOLOGY

Training protocol and evaluation: Training loops zeroed gradients, executed forward passes, computed losses, performed backward passes, and stepped the optimizer while recording costs. Evaluation ran without gradient tracking and produced losses and task metrics on train/test splits at regular checkpoints. Primary metrics were macro-F1 for classification and R^2 for regression. Learning curves plotted performance versus backpropagated instances with shaded one-standard-deviation bands over seeds.

Hyperparameter search and multi-seed runs: Grid searches tuned optimizer and strategy-specific controls under the same budgets across candidates. Passive learning searched learning rate, momentum, and weight decay. Optimization parameters found in the passive learning search were then used in both active learning strategies. ALUS searched initial labeled set size, items per round, and training epochs per

round. The number of members in the ALUS ensemble for regression was not included in the grid search, simply to reduce search size. The ensemble amount was set to three for all ALUS regression experiments. SASLA searched the sensitivity threshold control β and selection interval. The best settings by validation performance were carried forward to final evaluations. Each configuration was executed over five seeds to summarize mean $\pm$ standard deviation and to generate aggregated learning curves.

Logging and reproducibility: Each run recorded configuration, budgets consumed, losses, metrics, and (when applicable) labeled/subset sizes over time. Random seeds were set at process start to support reproducibility of data splits, weight initialization, and acquisition order.

EMPIRICAL PROCESS

In order to empirically compare the learning strategies, the ideal procedure would have involved conducting many runs with different random seeds for each dataset and applying formal significance testing. However, due to computational constraints, only five runs (seeds 0–4) were performed for each strategy–dataset pair. With such a small sample, classical statistical tests such as the paired $t-test$ were underpowered and potentially misleading. So, a descriptive comparison approach was adopted. For each run, the primary metrics recorded for classification tasks was $macro - F1$. For regression tasks reported the coefficient of determination, R^2. The $macro-F1$ score was defined as the harmonic mean of precision and recall, averaged across classes:

$$F1_{\text{macro}} = \frac{1}{C}\sum_{k=1}^{C}\frac{2\,\text{Prec}_k\,\text{Rec}_k}{\text{Prec}_k + \text{Rec}_k},$$

where C denoted the number of classes. Precision and recall were computed per class using one-vs-rest mappings, and then averaged. For regression tasks, the coefficient of determination was defined as

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

where y_i were the true targets, $\hat{y}_i$ were the predictions, and $\bar{y}$ was the sample mean of the targets.

For each strategy–dataset pair, the results were aggregated by reporting the mean and standard deviation of each metric across the five seeds. These values were presented in tables (mean $\pm$ standard deviation) and in learning-curve plots that displayed performance as a function of the compute budget, with error bars indicating one standard deviation. Comparisons were then made qualitatively: if the error bars overlapped substantially, this was interpreted as no clear advantage; if differences appeared consistent across budgets and datasets, they were cautiously noted as indicative of superior efficiency. Given the small number of runs, no strict claims of statistical significance were made. Instead, the results were framed as indicative trends.

RESEARCH RESULTS

The results from the hyperparameter tuning are shown in II. Overall, the search repeatedly settled on learning rate $LR = 0.1$ with momentum $M = 0.9$ for all datasets. With standardized inputs and a shallow NN, this pair gives fast, well-damped progress. Step sizes are large enough to move through flat regions, while momentum smooths noisy mini-batch gradients and helps traverse narrow valleys. The weight decay (WD) is small which is consistent with low-capacity models. Enough $L2$ to curb weight growth and help generalization, not so much that it underfits. The selection interval ξ and the acquisition batch size Top-K reflect the compute trade-off: for classification (Iris, Banknote, MNIST) $\xi = 1$ and $k \in \{5, 10\}$ encourage frequent and small updates where selection is cheap. For regression (Energy, Sine, Friedman1) $\xi = 10$ and $k = 1$ reduce scoring overhead and avoid needless batch picks. Finally, the US epoch (USE) is largest where lots of updates are needed to produce models capable of producing informative uncertainty scores (Sine). Smallest where little training is needed to produce quality uncertainty scores (Energy).

Problem	**LR**	**M**	**WD**	β	ξ	**Top K**	**USE**	**Budget**
Iris	0.1	0.9	0.0001	0.5	1	5	50	5,000
Banknote	0.1	0.9	0.001	0.7	1	10	50	15,000
MNIST	0.1	0.9	0.001	0.7	1	5	50	30,000
Energy	0.1	0.9	0.0001	0.7	10	1	50	30,000
Sine wave	0.1	0.9	0.001	0.9	10	1	100	500,000
Friedman1	0.1	0.9	0.001	0.7	10	1	50	200,000

TABLE II
TUNED HYPERPARAMETERS

For Iris, both active strategies produced higher final F1-macro than passive. In Figure 1 ALUS showed a consistent early advantage, while SASLA mean scores trail behind before exceeding passive. That trend is was consistent with the mechanics of ALUS and SASLA. ALUS quickly identified boundary-defining observations and ignored redundant instances. SASLA starts with the complete training data and $\xi > 1$, therefore initially follows exactly with passive. Then the cost of computing sensitivities dragged down the learning curve before the benefits of the newly constructed dataset kicked in. All strategies converge to a similar $F1$-macro with overlapping standard deviations as shown in table III.

Dataset	Passive	SASLA	ALUS	Metric
Iris	0.947 ± 0.0506	0.966 ± 0.0337	0.966 ± 0.0337	$F1$
Banknote	0.987 ± 0.0080	0.991 ± 0.0056	0.988 ± 0.0109	$F1$
MNIST	0.872 ± 0.0108	0.867 ± 0.0080	0.871 ± 0.0128	$F1$
Energy	0.875 ± 0.0194	0.849 ± 0.0462	0.871 ± 0.0159	R^2
Sine wave	0.795 ± 0.3898	0.792 ± 0.3916	0.961 ± 0.0445	R^2
Friedman1	0.899 ± 0.0387	0.900 ± 0.0547	0.777 ± 0.0350	R^2

TABLE III
SUMMARY PERFORMANCE METRICS

For the Banknote dataset, the learning curves rose rapidly and converged near the ceiling for all regimes, with narrow error bars throughout Figure 2. The final means were tightly clustered, with SASLA slightly higher on average than the

Fig. 1.

Fig. 2.

passive baseline and US. This behaviour was expected on an easily separable, low-dimensional problem where the passive learner encountered sufficiently informative patterns early under the available budget, limiting the room for acquisition to improve coverage. The parity also reflected that the acquisition steps consumed compute that would otherwise have been spent on additional weight updates, while offering little extra information once the boundary had been pinned.

For MNIST, the passive learner slightly outperformed US and essentially matched SASLA at the final budget in Figure 3, with all gaps small relative to the standard deviations. This outcome was plausible for a shallow, fully connected model. Repeated acquisitions of uninformative digits by US reduced broad coverage that the chosen architecture needed for good generalization. While SASLA, with a relatively gentle selection configuration, behaved much like passive fixed-set learning. The shared-budget protocol also implied that acquisition overhead displaced some backprop updates, further reducing potential gains.

For Energy, the passive learner achieved the highest final R^2, US trailed slightly, and SASLA lagged more clearly as seen in 4. Two factors explain this ordering. First, sensitivity-thresholding periodically reduced the effective training subset; when configured too aggressively, that reduction emphasised steep or locally high-curvature regions at the expense of the broad coverage needed in this domain. This increased variance

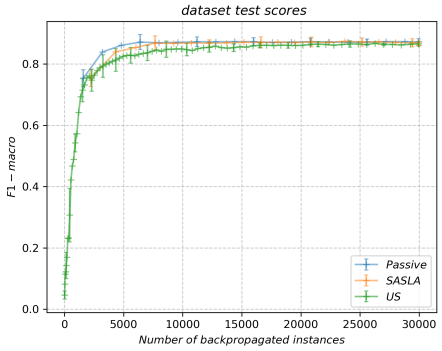

Fig. 3.

and slowed convergence. Second, the US ensemble's selections tended to stale as members increasingly agreed, lowering acquisition quality while still incurring the extra compute per round. Both mechanisms were consistent with the documented compute trade-offs and ensemble dynamics.

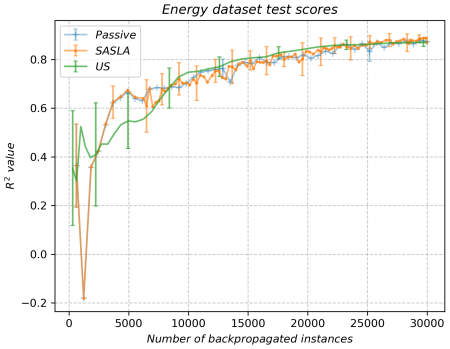

Fig. 4.

For Friedman1, US underperformed clearly, with early instability and a lower final plateau in Figure 5. SASLA matched the passive baseline at convergence. The synthetic function contains strong interaction effects; as the US ensemble converged, predictive variance diminished and the acquisition signal degraded. All while still multiplying the per-update cost by the ensemble size. SASLA's gradient-based filter retained emphasis on higher-impact regions without starving general coverage under the shared budget, which explained the parity with passive.

For Sine-1D, US delivered a decisive advantage, attaining high R^2 very early and maintaining it across the full budget in Figure 6, with small error bars. The ensemble variance used by US aligned strongly with the true information structure of the function. Regions around peaks and troughs were identified as most uncertain and were labeled promptly. Passive and SASLA showed substantially larger run-to-run variability and lower means because early batches did not reliably expose those critical regions. Here, the additional compute for the ensemble paid for itself through a large label-efficiency gain.

Fig. 5.

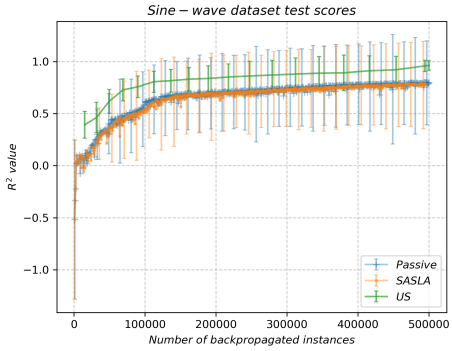

Fig. 6.

Across datasets, no single regime dominated. The observed ordering depended on the interaction between acquisition logic and data, as well as on the inductive bias of the model. US excelled on Sine-1D, matched or trailed on MNIST and Energy, and suffered on Friedman1; SASLA matched passive on MNIST and Friedman1, led slightly on Banknote, and trailed on Energy. These patterns were consistent with a no-free-lunch view and with the shared-budget framing, where active learning traded a portion of the allowed backpropagations for the selection steps needed to identify informative points. Consequently, outcomes varied with the usefulness of the acquisition signal relative to the task, the selection frequency, and the ensemble or threshold controls.

Variance in the curves added context to the means. Wide error bars for passive and SASLA on Sine-1D and transient instability for US on Friedman1 indicated sensitivity to seeds and acquisition hyperparameters. Means and standard deviations over five seeds therefore served as descriptive evidence, and the learning-curve shapes in Figures 3–8 carried much of the interpretive weight. This approach aligned with the empirical process adopted for the study.

The compute trade-offs were central to the interpretation. The ensemble size in US directly scaled backprop cost, while also risking a stale variance signal as members converged; SASLA's selection interval and β parameter controlled a three-way trade-off among selection overhead, subset size, and convergence speed. Different choices for these controls would lead to different results under the same total budget. The tuned values used to produce Figures 3–8 and Table III are specified in Table II. Cross-reference guide for the reader. Final per-dataset means and standard deviations appear in Table III. Learning-curve behaviour and error bars appear in Figures 3–8. This report section referred to those materials in the text, as required for figures and tables.

II. Conclusion

Under a shared compute budget, neither uncertainty sampling nor sensitivity-based selection dominates a passive baseline across tasks. ALUS delivers large, stable gains on Sine-1D where uncertainty pinpoints high-information regions, but is fragile on Friedman1 and offers little advantage on MNIST or Energy. SASLA matches or slightly improves on passive when decision-boundary refinement is the main bottleneck (Iris, Banknote), yet it can underperform when broad coverage matters (Energy). These results are consistent with the central trade-off observed throughout: compute spent on *acquiring* informative points (pool scoring, ensembles, resampling) reduces compute available for *optimizing* the network.

Practically, the choice of strategy should be guided by problem geometry and model bias. Boundary-dense, low-dimensional or highly structured tasks tend to reward targeted acquisition; heterogeneous tabular regression often prefers broad, passive coverage unless selection signals are exceptionally well aligned with error. With only five seeds and shallow models, our findings are descriptive rather than inferential; nevertheless, the patterns across Table III and Figures 1–6 support a clear message: there is no free lunch in active learning for neural networks. Future work could integrate selection cost into the objective explicitly, explore stronger inductive biases (e.g., CNNs for images), and test adaptive schedules for ensemble size and sensitivity thresholds to balance search versus optimization more effectively.

References

All code can be found at on my public git repository at