Sample Efficiency in Neural Networks: An Empirical Evaluation of Sensitivity Analysis and Conflicting-Evidence Query Strategies

Theunis Frans Kruger
Department of Economic and Management Sciences
Stellenbosch University
Stellenbosch, South Africa
25846205@sun.ac.za

***Abstract*—The practical application of supervised machine learning, particularly with data-intensive models such as neural networks, is often constrained by the high cost associated with data labelling. Active learning presents an idea to mitigate this issue by allowing a model to intelligently query for the labels of the most informative unlabelled data points. This report presents an empirical comparison of a standard passive learning approach against two distinct active learning strategies for training a feedforward neural network. The strategies evaluated are the informativeness-based Sensitivity Analysis Selective Learning Algorithm (SASLA) and the uncertainty-based Conflicting-Evidence Uncertainty Sampling (ALUS). The evaluation is conducted across six benchmark datasets, comprising three classification and three function approximation problems. A single-hidden-layer neural network is trained iteratively within a pool-based active learning simulation. The results indicate that the effectiveness of active learning is highly context-dependent. No single active learning strategy proved universally superior across all tasks, and in some cases, the simple passive learning baseline remained competitive, particularly on simpler datasets. The implementation of this project is available online[1].**

***Index Terms*—Active Learning, Neural Networks, Supervised Learning, Sample Efficiency, Uncertainty Sampling, Sensitivity Analysis.**

I. Introduction

The rise of deep learning and artificial neural networks has led to advances across various domains. A common characteristic of these powerful models is their reliance on large quantities of labelled data to achieve high performance. In many real-world applications, obtaining high-quality labelled datasets is however a major bottleneck. The process of data labelling can be disruptively expensive, requiring substantial investment in terms of time, financial resources, and the consultation of domain experts. This practical constraint motivates the investigation of data-efficient learning methods that can maximise model performance while minimising the need for labelled examples.

Active learning is a machine learning concept designed specifically to address this challenge. In contrast to conventional passive learning, where a model is trained on a fixed, randomly sampled set of labelled data, an active learner has the ability to interact with an oracle, such as a human labeller. The learner can selectively query the oracle for the labels of unlabelled instances that it considers to be the most valuable for its training process.The main aim is that by intelligently choosing which data to learn from, an active learner can achieve a desired level of performance with significantly fewer labelled instances than a passive learner [1], [2], [5].

The objective of this report is to empirically investigate the performance and sample efficiency of a simple neural network trained under three distinct learning scenarios. The scenarios are:

1) A passive learning baseline, which represents the standard supervised learning approach.
2) An informativeness-based active learning strategy, the Sensitivity Analysis Selective Learning Algorithm (SASLA).
3) An uncertainty-based active learning strategy, the Conflicting-Evidence Uncertainty Sampling (ALUS).

The investigation is to determine not only whether active learning provides a clear benefit over passive learning, but also to understand the conditions under which different types of query strategies perform better. The analysis is based on an experimental framework that utilises a single-hidden-layer neural network as the learner, evaluated on a three classification and three function approximation (regression) benchmark datasets.

The findings of this study reveal inconsistent performance patterns. The results suggest that the choice of a learning strategy is not straightforward and that the benefits of active learning are highly dependent on the nature of the task and the data. The SASLA strategy, which selects data based on model sensitivity, showed particular promise for regression problems. In contrast, the ALUS strategy, which resolves a specific form of model uncertainty, was effective for some of the considered classification tasks. Active learning did not universally outperform the passive baseline, highlighting that the additional complexity of an active learning strategy must be justified by proven gains in sample efficiency.

This report is structured as follows. Section II provides the theoretical background on artificial neural networks and the

[1] https://github.com/FrancoKrugerUni/ML_Assign3

concept of active learning. Section III describes the specific network architecture and the three learning strategies. The practical implementation is described in Section IV, while the empirical setup and experimental setups are outlined in Section V. Section VI presents and discusses the results of the comparative experiments. Finally, Section VII concludes the report with a summary of the findings and potential future work.

II. Background

This section provides the foundational concepts required to understand the components of this study. The discussion begins with an overview of artificial neural networks, followed by a description of active learning and the theoretical principles behind common query strategies.

A. Artificial Neural Networks

An artificial neural network (ANN) is a computational model inspired by the structure and function of biological neural systems. The specific architecture employed in this study is the Multi-Layer Perceptron (MLP), a form of feedforward ANN. An MLP consists of an input layer, one or more hidden layers, and an output layer, organised in a directed graph where information flows from input to output.

Each layer is composed of basic processing units called neurons. A neuron in a given layer receives weighted inputs from all neurons in the preceding layer. The neuron computes a weighted sum of these inputs, adds a bias term, and then passes the result through a non-linear activation function. This non-linearity is crucial, as it allows the network to learn complex, non-linear relationships between inputs and outputs. A common choice for the activation function in hidden layers is the Rectified Linear Unit (ReLU), defined as $f(z) = \max(0, z)$.

The process of training an MLP involves adjusting the network's weights and biases to minimise a loss function, which quantifies the difference between the network's predictions and the true target values. This optimisation is normally achieved using the backpropagation algorithm together with a gradient-based optimisation method, such as stochastic gradient descent (SGD). Backpropagation efficiently computes the gradient of the loss function with respect to each weight in the network, and the optimiser uses this gradient to update the weights in the direction that reduces the loss.

B. Active Learning

Active learning is a subfield of machine learning where a learning algorithm can interactively query a source of information (an oracle) to obtain labels for new data points. This study focuses on the pool-based active learning scenario, which is the most common setting.

The core components of a pool-based active learning system are:

- A small, initial set of labelled training data, denoted as L.
- A large pool of unlabelled data, denoted as U.
- A learner model, which is the underlying machine learning model to be trained (in this case, an MLP).
- A query strategy, which is an algorithm used to select the most valuable instances from U.

The active learning process proceeds in an iterative cycle:

1) The learner model is trained on the current labelled set L.
2) The query strategy is applied to the unlabelled pool U to select one or more instances.
3) An oracle provides the true labels for the selected instances.
4) The newly labelled instances are moved from the unlabelled pool U to the labelled set L.
5) The process repeats from step 1 until a predefined stopping criterion, such as a labelling budget, is satisfied.

The goal of this process is to achieve a higher level of model performance for a given number of labelled instances compared to a passive learning approach, thus improving the overall sample efficiency of the learning process.

C. Query Strategy Frameworks

The effectiveness of an active learning system is primarily determined by its query strategy. These strategies can be grouped into several families based on the heuristic used to measure the value of an unlabelled instance.

1) Uncertainty Sampling: Uncertainty sampling is the simplest and most common query strategy framework. The core principle is to query the instances for which the model is least certain about its prediction. The intuition is that these instances lie close to the decision boundary or in areas of the input space that the model has not learned well yet. Labelling these points provides the most direct information for refining the model's understanding of these uncertain regions. Several metrics can be used to quantify uncertainty, including least confident sampling, margin sampling, and entropy-based approaches.

2) Informativeness-Based Sampling: This framework aims to select instances that are expected to impart the most information to the model. The value of an instance is measured by the potential impact that its label would have on the model's state. One way to approximate this impact is to measure the expected change in the model's parameters, often quantified by the gradient of the loss function. Another approach, relevant to this study, is to measure the sensitivity of the model's output to changes in the input. Instances that cause high output sensitivity are considered informative because they fall in regions where the model's function is changing rapidly.

D. Evaluated Query Strategies

This study evaluates three specific query strategies, each representing a different approach to sample selection. The high-level logic for each is presented below.

1) Passive Learning (Baseline): The passive learning strategy serves as the baseline for comparison, representing a standard supervised learning scenario where data labels are acquired without guidance from the model. To ensure an interpretable comparison with the iterative active learning strategies, the passive learning process is also performed iteratively.

After training on the small initial labelled set, the performance of the model is evaluated. Then, in each subsequent "query" step, a single instance is chosen uniformly at random from the unlabelled pool and added to the training set. The model is then retrained on this expanded set, and its performance is evaluated again. This process of random selection, retraining, and evaluation is repeated until the labelling budget is exhausted. The operation of this method is formalised in Algorithm 1.

Algorithm 1 Passive Learning Selection

```
procedure PASSIVE-SELECT($U$)
    $i \leftarrow$ randomly select one index from the pool $U$
    return $\{i\}$
end procedure
```

2) Sensitivity Analysis Selective Learning (SASLA): This method implements the informativeness-based approach, assuming that the most informative samples for a classifier are those near the decision boundaries. It quantifies this "closeness" by measuring the sensitivity of the output of the model to small input changes using the Jacobian matrix. Samples with sensitivity scores above a dynamic threshold are selected for training.

Algorithm 2 SASLA Selection

```
procedure SASLA-SELECT($M, U, \beta$)
    $\Phi \leftarrow$ empty list for informativeness scores
    for all samples $x \in$ U do
        $\vec{S_o} \leftarrow$ Calculate output sensitivity to $x$    ▷ With Jacobian $\nabla_x M(x)$
        $\phi_x \leftarrow ||\vec{S_o}||_\infty$    ▷ Max-norm of sensitivity vector
        Append $\phi_x$ to $\Phi$
    end for
    $\bar{\Phi} \leftarrow$ Average($\Phi$)
    $\tau \leftarrow (1 - \beta) \times \bar{\Phi}$    ▷ Calculate selection threshold
    $S \leftarrow$ indices of all samples $x$ where $\phi_x > \tau$
    if $S$ is empty then
        $S \leftarrow$ index of sample with max($\Phi$)
    end if
    return $S$
end procedure
```

3) Conflicting-Evidence Uncertainty Sampling (ALUS): This approach modifies the uncertainty sampling framework by analysing the reason for the uncertainty of the model. It first identifies a pool of the most uncertain candidates using an entropy-based metric. Then, within this pool, it selects the single instance with the most conflicting evidence, where different features provide strong but opposing signals for different classes.

Algorithm 3 ALUS Conflicting-Evidence Selection

```
procedure ALUS-SELECT($M, U, t$)
    $P \leftarrow$ Calculate probabilities of getting $U$ with $M$
    $H \leftarrow$ Calculate entropy of $P$    ▷ Uncertainty scores
    $U_{uncertain} \leftarrow$ indices of top $t$ samples from $H$
    $max\_score \leftarrow -1$; $best\_index \leftarrow$ null
    for all index i $\in U_{uncertain}$ do
        $\{y_m, y_n\} \leftarrow$ top two predicted classes for $P_i$
        $E_m \leftarrow$ Calculate evidence for $y_m$
        $E_m \leftarrow$ Calculate evidence for $y_m$
        $score \leftarrow E_m \times E_n$    ▷ Conflicting evidence score
        if $score > max\_score$ then
            $max\_score \leftarrow score$; $best\_index \leftarrow i$
        end if
    end for
    return $\{best\_index\}$
end procedure
```

III. METHODOLOGY

This section provides a description of the neural network model and the three learning strategies implemented for this comparative study. The design choices are outlined to ensure the work is reproducible.

A. Network Architecture and Training

The learner model used in all experiments is a fully-connected feedforward neural network. The architecture consists of an input layer, a single hidden layer, and an output layer. The number of neurons in the input layer is determined by the feature dimensionality of the specific dataset. The number of neurons in the hidden layer, N_h, is a hyperparameter tuned for each dataset. The output layer contains a number of neurons equal to the number of classes for classification tasks, or a single neuron for regression tasks. The rectified linear unit (ReLU) activation function is applied to the outputs of the hidden layer neurons. The ReLU activation function is selected mainly due to its computational efficiency and potential of overcoming the vanishing gradient problem. For classification tasks, a Softmax function is applied to the output layer to produce a probability distribution over the classes. For regression tasks, no activation function is used on the output neuron. The network weights are optimised using the stochastic gradient descent (SGD) algorithm. To mitigate overfitting, L2 regularisation is incorporated directly into the optimiser through its `weight_decay` hyperparameter. The weight decay hyperparameter is tuned individually for each dataset. The choice of loss function is dependent on the task. For multi-class classification, the Cross-Entropy Loss is used, and for regression the Mean Squared Error (MSE) loss.

B. Passive Learning Strategy

Starting with a small initial set, random instances are added to the training set at each stage. No guidance or information is considered regarding which instance might be valuable. This iterative random sampling approach produces a learning curve that directly illustrates the performance gains relative to the number of labels, providing a baseline against which the intelligence of the active learning strategies can be measured. This aligns with the process formalised in Algorithm 1. The passive learning strategy represents a baseline for the active learning strategies to improve upon.

C. Sensitivity Analysis Selective Learning Algorithm (SASLA)

The SASLA strategy is an informativeness-based approach that aims to select a batch of patterns that are most sensitive to the model's current state [2]. The selection process for each query iteration involves three steps. The informativeness $\Phi(x_i)$ is first computed for each instance x_i in the unlabelled pool U. This is defined as the sensitivity of the output of the network to small changes in the input vector [2]. In this implementation, this begins by computing the Jacobian matrix, J_i, of the network's output vector with respect to the input vector x_i. The output sensitivity vector, $\mathbf{s}_o$, is then calculated as the L_1 norm of the rows of the Jacobian matrix [2]. The final informativeness score, $\Phi(x_i)$, is the max-norm of this sensitivity vector, $\mathbf{s}_o$ [2]. A selection threshold is then determined. The average informativeness $\bar{\Phi}$ is calculated across all instances currently in the pool U. A dynamic threshold T is then defined as $T = (1.0 - \beta) \times \bar{\Phi}$, where β is a hyperparameter that controls how selective the strategy is [2]. To finish, the query set is formed. All instances x_i from the pool U for which the informativeness $\Phi(x_i)$ is greater than the threshold T are selected for labelling. As a fallback mechanism, if this process yields an empty set, the single instance with the highest informativeness score is selected [2]. This method is formalised as Algorithm 2.

D. Conflicting-Evidence Uncertainty Sampling (ALUS)

The ALUS strategy is an uncertainty-based approach designed for classification tasks, following the framework proposed by Sharma and Bilgic [1]. With ALUS, a single instance per query iteration is selected by identifying a point of high uncertainty that is also supported by strong but conflicting internal evidence within the network. The process involves three main steps. An initial candidate set is formed by identifying the top t most uncertain instances from the unlabelled pool U. The hyperparameter t defines the size of this candidate set. Uncertainty is quantified using Shannon entropy, calculated from the probability distribution produced by the Softmax function at the network's output layer [1]. Then a "conflicting evidence" score is computed for each instance in the candidate set. This score allows the network's decision process to be investigated. For a given instance, the two most likely output classes, denoted y_m and y_n, are identified. The "evidence" for each class is then calculated based on the contributions from the hidden layer. Following the formulation for models similar to logistic regression, the evidence for class y_m, denoted $E_{+1}(x)$, is the sum of positive weighted activations contributing to its output logit [1]. The evidence for class y_n is calculated similarly. The final score for the instance is the product of these two evidence values, $E_{+1}(x) \times E_{-1}(x)$ [1]. This product is large when the network has strong internal support for two competing options. Lastly, the instance from the candidate set with the highest conflicting evidence score is selected for labelling [1]. This approach is formalised in Algorithm 3.

IV. Implementation

This section focuses on the specific implementation of the discusses methods. Flow diagrams are provided to showcase the logical steps as followed by each query strategy.

A. Passive Learning

Passive learning serves as the baseline strategy, in which no selection mechanism is applied. Instead, the model is trained on all available data points without assessing their informativeness or uncertainty. In the implementation, this corresponds to simply returning the indices of all samples in the pool. Although computationally inexpensive, passive learning does not attempt to optimise the labelling process, making it less efficient when labelled data is costly to obtain. The step is visualised in Figure 1.

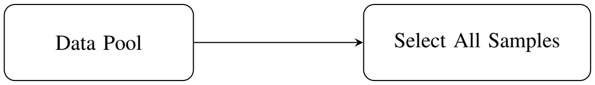

Figure 1. Passive learning process.

B. SASLA

The SASLA method selects samples based on their informativeness, which is derived from the sensitivity of the model outputs with respect to its inputs. For each candidate sample, the Jacobian of the model outputs with respect to the input features is computed. The sensitivity for a given output class is defined as the L1 norm of the Jacobian row, capturing how strongly the output responds to changes in the input. The informativeness score of a sample is then the maximum of these sensitivities across output classes.

To avoid selecting samples with very low informativeness, a threshold is introduced. This threshold is proportional to the average informativeness across the pool, scaled by a factor of $(1 - \beta)$, where $\beta \in [0, 1]$ controls the level. All samples with informativeness above the threshold are selected, with a fallback mechanism that ensures at least the most informative sample is chosen. This approach, illustrated in Figure 2, guides the selection towards data points expected to have the biggest impact on improving model performance.

C. ALUS

The third strategy combines standard uncertainty sampling with an additional conflicting evidence criterion, visualised in Figure 3. The model first predicts class probabilities for all samples, and the uncertainty of their prediction is measured using entropy. The top t most uncertain samples are identified as candidates. From this subset, the algorithm then measures how much conflicting evidence exists in the network's representations for the two most likely classes. Specifically, the

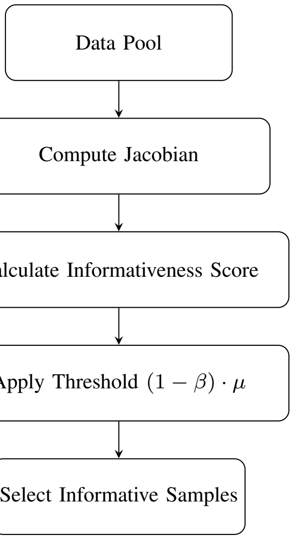
Data Pool
Compute Jacobian
Calculate Informativeness Score
Apply Threshold $(1-\beta)\cdot\mu$
Select Informative Samples

Figure 2. SASLA selection process.

method computes the positive contributions of hidden layer activations to the logits of the two most likely classes. The final evidence score is computed as the product of these contributions, showing the extent of conflict in the model's internal computations. The sample with the highest evidence score is selected. If all evidence scores are zero, the algorithm defaults to selecting the most uncertain sample.

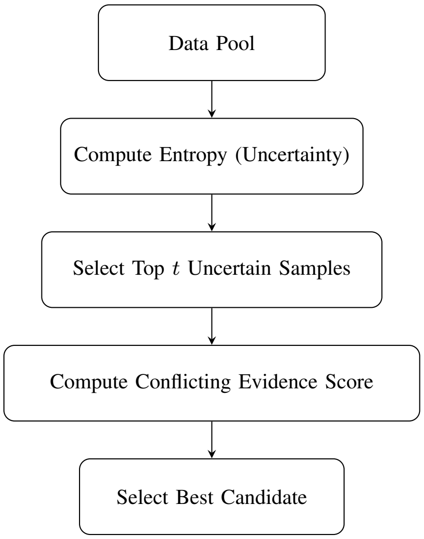
Data Pool
Compute Entropy (Uncertainty)
Select Top t Uncertain Samples
Compute Conflicting Evidence Score
Select Best Candidate

Figure 3. ALUS process.

Together, these strategies allow for a comparative study between baseline training (passive), gradient-based sensitivity analysis (SASLA), and a hybrid approach that combines uncertainty with hidden-layer evidence (ALUS).

D. Programming

All processes are implemented in Python 3. The `scikit-learn` library is used for general machine learning purposes including scaling, metrics, and loading and splitting of datasets. Visualisations are made using `matplotlib`. The six datasets are imported from `scikit-learn` using the following six packages: `load_iris`, `load_wine`, `load_breast_cancer`, `load_diabetes`, `make_friedman1`, and `make_regression`. The `PyTorch` library is used for construction of the neural network architecture. Specifically, a `Sequential` implementation is built for interpretability. All models are trained using standard backpropagation and loss functions appropriate for the specific task, namely cross-entropy for classification and MSE for regression. A statistical test is performed to evaluate the strength of the final results. These tests are implemented with the `stats` module of the `scipy` library. The implementation is made and run on a Dell XPS 13 laptop with an Intel i5 core and 8 gigabytes of RAM. Computing power and time are limited, and implementation decisions reflect this. Although resource constraints limit the extent of certain experiments, such as the hyperparameter search for the neural network and the number of active learning iterations, all experiments are executed in a manner designed to maximise the statistical power and validity of the final comparisons within the available computational budget.

V. Empirical Process

This section outlines the experimental procedure followed to evaluate and compare the learning strategies. The details provided cover the benchmark datasets, data preprocessing steps, performance metrics, and the overall experimental setup.

A. Benchmark Datasets

The study utilises six publicly available benchmark datasets to provide a range of learning challenges. These are divided into three for classification and three for function approximation (regression), as described in Table I.

The classification datasets are Iris[2], Wine[3], and Breast cancer[4], while the regression datasets are Diabetes[5], Synthetic regression[6], and Friedman1[7].

Table I
Benchmark Dataset Characteristics

Dataset	Task	Instances	Features	Classes
Iris	Classification	150	4	3
Wine	Classification	178	13	3
Breast cancer	Classification	569	30	2
Diabetes	Regression	442	10	1
Synthetic regression	Regression	500	15	1
Friedman1	Regression	500	10	1

A brief description of each dataset is provided for context:

[2] https://archive.ics.uci.edu/dataset/53/iris

[3] https://archive.ics.uci.edu/dataset/109/wine

[4] https://archive.ics.uci.edu/dataset/14/breast+cancer

[5] https://archive.ics.uci.edu/dataset/34/diabetes

[6] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_regression.html

[7] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_friedman1.html

- Iris: A classic, relatively simple dataset for multi-class classification. It involves predicting one of three species of iris flower based on four botanical measurements. Its well-defined class boundaries make it a good baseline for measuring fundamental algorithm performance.
- Wine: A slightly more complex multi-class problem than Iris, involving classifying three types of Italian wines based on 13 chemical attributes. The higher feature dimensionality and less distinct class separation offer a bigger challenge.
- Breast Cancer: A binary classification task where the goal is to diagnose tumours as positive or negative based on 30 real-valued features computed from a digitised image.
- Diabetes: A standard regression task where the objective is to predict the progression of diabetes one year after baseline, based on ten baseline variables.
- Synthetic Regression: A synthetic dataset generated with 15 features, 10 of which are informative. This controlled environment is useful for testing an algorithm's ability to model a linear relationship while ignoring non-informative features.
- Friedman1: A synthetic regression problem with built-in non-linear relationships. Successfully modelling this function requires capturing complex interactions between features.

B. Data Preprocessing and Partitioning

For each dataset, the available data is partitioned into a training set, containing 80% of the instances, and a testing set, containing the remaining 20%. To maintain the proportional representation of classes in both partitions for the classification datasets, stratified sampling is performed during the split. A check confirmed no missing values in any dataset. Since feature scaling is a necessary preprocessing step for neural networks, a `StandardScaler` is used to transform the features to have zero mean and unit variance. To prevent data leakage from the test set into the training process, the scaler is fitted only on the training data. The mean and standard deviation learned from the training data are then used to transform both the training and the testing sets.

C. Performance Metrics

The performance of the trained models is evaluated using standard metrics appropriate for each task type. For classification tasks, performance is measured using accuracy, defined as the proportion of correctly classified instances in the held-out test set. For regression tasks, performance is measured using mean squared error (MSE), which represents the average of the squared differences between the predicted and actual target values on the test set. For regression, a lower MSE value indicates better performance.

D. Hyperparameter Tuning

To ensure a fair and robust comparison, hyperparameters for the neural network and the active learning strategies are tuned independently for each dataset. The tuning is performed using a systematic one-at-a-time approach. The process for each dataset is as follows:

1) Define Parameter Space: A predefined search space is established for key hyperparameters:
   - Learning rate: [0.01, 0.005, 0.001]
   - L2 weight decay strength: [1e-3, 1e-4, 1e-5]
   - Number of hidden neurons: [30, 50, 70]
   - SASLA β: [0.7, 0.8, 0.9]
   - ALUS pool size t: [15, 20, 25]
2) Iterative Search: The tuning proceeded in a greedy, iterative manner. For each parameter, each value in its search space is evaluated while keeping other parameters fixed at their current best known values.
3) Benchmark-Based Evaluation: The evaluation for each parameter value is performed by running a lightweight active learning simulation. For general parameters such as learning rate and hidden neurons, the dataset's primary active learning strategy (ALUS for classification, SASLA for regression) is used as the benchmark. For strategy-specific parameters (β and t), the relevant strategy is used. Each simulation is run for 3 independent trials with a small budget of 30 queries to quickly estimate the effectiveness of the parameter.
4) Selection Criterion: The parameter value that results in the best average final performance (accuracy for classification, MSE for regression) across the trials is selected as the optimal value for that hyperparameter. This new value is then used in the subsequent tuning steps for the remaining parameters.

This approach ensures that the selected hyperparameters are customised for each dataset, minimising the risk that performance differences are due to poor configuration rather than the learning strategy itself. The detailed results of this tuning process are presented in Tables II–VII. The parameter configurations eventually used for each dataset are summarised in Table VIII.

Table II
HYPERPARAMETER TUNING RESULTS FOR THE BREAST CANCER DATASET

Param.	Value	Benchmark	Mean Acc. ± Std. Dev.
Alus T	15.0	ALUS	0.9620 ± 0.0083
	20.0	ALUS	0.9649 ± 0.0124
	25.0	ALUS	0.9591 ± 0.0041
Hidden Neurons	30.0	ALUS	0.9532 ± 0.0041
	50.0	ALUS	0.9620 ± 0.0230
	70.0	ALUS	0.9649 ± 0.0124
Learning Rate	0.01	ALUS	0.9444 ± 0.0180
	0.005	ALUS	0.9240 ± 0.0230
	0.001	ALUS	0.6608 ± 0.0735
Sasla Beta	0.7	SASLA	0.9123 ± 0.0143
	0.8	SASLA	0.9152 ± 0.0083
	0.9	SASLA	0.9298 ± 0.0072
Weight Decay	0.001	ALUS	0.9503 ± 0.0109
	0.0001	ALUS	0.9532 ± 0.0109
	1e-05	ALUS	0.9561 ± 0.0248

Table III
HYPERPARAMETER TUNING RESULTS FOR THE DIABETES DATASET

Param.	Value	Benchmark	Mean MSE ± Std. Dev.
Hidden Neurons	30.0	SASLA	3705.3353 ± 312.9905
	50.0	SASLA	3513.7454 ± 88.1993
	70.0	SASLA	3500.4720 ± 50.2960
Learning Rate	0.01	SASLA	3604.4284 ± 238.0436
	0.005	SASLA	3419.6340 ± 217.5989
	0.001	SASLA	4478.0088 ± 555.2061
Sasla Beta	0.7	SASLA	3539.1935 ± 185.1776
	0.8	SASLA	3547.2313 ± 193.2966
	0.9	SASLA	3588.4338 ± 195.3918
Weight Decay	0.001	SASLA	3628.4963 ± 56.2789
	0.0001	SASLA	3215.0059 ± 117.0991
	1e-05	SASLA	3589.1738 ± 220.7312

Table IV
HYPERPARAMETER TUNING RESULTS FOR THE FRIEDMAN1 DATASET

Param.	Value	Benchmark	Mean MSE ± Std. Dev.
Hidden Neurons	30.0	SASLA	11.8179 ± 0.4457
	50.0	SASLA	9.3502 ± 0.3402
	70.0	SASLA	8.9964 ± 0.6089
Learning Rate	0.01	SASLA	8.5516 ± 0.8284
	0.005	SASLA	12.3754 ± 2.0247
	0.001	SASLA	21.5073 ± 1.4854
Sasla Beta	0.7	SASLA	8.4326 ± 0.5793
	0.8	SASLA	8.2571 ± 0.7149
	0.9	SASLA	7.8798 ± 0.6096
Weight Decay	0.001	SASLA	8.3597 ± 0.8430
	0.0001	SASLA	9.9626 ± 0.7645
	1e-05	SASLA	7.6479 ± 0.4520

Table V
HYPERPARAMETER TUNING RESULTS FOR THE IRIS DATASET

Param.	Value	Benchmark	Mean Acc. ± Std. Dev.
Alus T	15.0	ALUS	0.7778 ± 0.0567
	20.0	ALUS	0.8556 ± 0.0157
	25.0	ALUS	0.8778 ± 0.0157
Hidden Neurons	30.0	ALUS	0.8000 ± 0.0272
	50.0	ALUS	0.8333 ± 0.0272
	70.0	ALUS	0.9000 ± 0.0272
Learning Rate	0.01	ALUS	0.8556 ± 0.0157
	0.005	ALUS	0.7889 ± 0.0314
	0.001	ALUS	0.6222 ± 0.1133
Sasla Beta	0.7	SASLA	0.7889 ± 0.0157
	0.8	SASLA	0.8222 ± 0.0314
	0.9	SASLA	0.7889 ± 0.0416
Weight Decay	0.001	ALUS	0.8111 ± 0.0786
	0.0001	ALUS	0.8444 ± 0.0416
	1e-05	ALUS	0.8556 ± 0.0157

Table VI
HYPERPARAMETER TUNING RESULTS FOR THE SYNTHETIC REGRESSION DATASET

Param.	Value	Benchmark	Mean MSE ± Std. Dev.
Hidden Neurons	30.0	SASLA	10301.8219 ± 515.0920
	50.0	SASLA	8267.5342 ± 1181.0339
	70.0	SASLA	7162.7847 ± 1032.7444
Learning Rate	0.01	SASLA	7094.0584 ± 1379.1993
	0.005	SASLA	13727.4404 ± 809.3949
	0.001	SASLA	22238.4128 ± 2702.4239
Sasla Beta	0.7	SASLA	8305.3249 ± 1201.0869
	0.8	SASLA	7720.3312 ± 2415.5924
	0.9	SASLA	8886.4704 ± 816.0330
Weight Decay	0.001	SASLA	7511.2686 ± 2342.7943
	0.0001	SASLA	7200.9577 ± 1248.2303
	1e-05	SASLA	8477.6759 ± 1260.8202

Table VII
HYPERPARAMETER TUNING RESULTS FOR THE WINE DATASET

Param.	Value	Benchmark	Mean Acc. ± Std. Dev.
Alus T	15.0	ALUS	0.9630 ± 0.0131
	20.0	ALUS	0.9444 ± 0.0600
	25.0	ALUS	0.9722 ± 0.0227
Hidden Neurons	30.0	ALUS	0.9167 ± 0.0454
	50.0	ALUS	0.9074 ± 0.0729
	70.0	ALUS	0.9352 ± 0.0346
Learning Rate	0.01	ALUS	0.9167 ± 0.0393
	0.005	ALUS	0.8611 ± 0.0227
	0.001	ALUS	0.5926 ± 0.0131
Sasla Beta	0.7	SASLA	0.9537 ± 0.0131
	0.8	SASLA	0.9444 ± 0.0227
	0.9	SASLA	0.9352 ± 0.0262
Weight Decay	0.001	ALUS	0.9630 ± 0.0262
	0.0001	ALUS	0.9444 ± 0.0000
	1e-05	ALUS	0.9907 ± 0.0131

E. Experimental Setup

All experiments are conducted within the described active learning framework. To ensure a statistically sound comparison, each configuration is repeated for ten independent trials using different random seeds for initialisation. The mean and standard deviation across trials are reported, and the Wilcoxon signed-rank test is applied to paired results to assess the significance of differences between strategies. Each trial starts with a small, randomly selected initial labelled set. The model is then trained on this set, and subsequently used to query new instances from the unlabelled pool according to the specified strategy. This iterative process of training, querying, and labelling continues until a predefined labelling budget is exhausted.

VI. RESULTS AND DISCUSSION

This section presents and discusses the empirical results obtained from the comparison of the passive, SASLA, and ALUS learning strategies across the six benchmark datasets. The analysis is organised by problem type, first discussing the performance on the three classification problems, followed by the three function approximation problems. The findings are supported by the illustrated learning curves and the statistical significance tests presented in Table XI. The final hyperparameters found by the search are presented in Table VIII. The configurations in this table are used to evaluate performance per dataset in this section.

Table VIII
HYPERPARAMETER CONFIGURATIONS

Dataset	LR	Decay	Neurons	β	t	Budget	Epochs
Iris	0.01	1e-05	70	0.8	25	40	100
Wine	0.01	1e-05	70	0.7	25	50	100
Cancer	0.01	1e-05	70	0.9	15	100	100
Diabetes	0.005	0.0001	70	0.7	-	100	100
Friedman	0.01	1e-05	70	0.9	-	100	100
Synthetic	0.01	0.0001	70	0.8	-	100	100

On nearly all datasets, optimal performance was achieved with a high (0.01) learning rate. The only exception is the Diabetes regression dataset. Performance also improved for all datasets when more hidden layer neurons were used.

A. Performance on Classification Tasks

The classification experiments revealed task-dependent advantages for the active learning strategies compared to the passive learning baseline. The learning curves demonstrate that both SASLA and ALUS frequently achieved higher accuracy with fewer labelled instances, though the magnitude of improvement varied considerably across datasets.

1) Breast Cancer Dataset: The breast cancer dataset provided a clear demonstration of active learning effectiveness. The ALUS strategy showed superior performance, achieving a final mean accuracy of 0.953 ± 0.017 compared to 0.935 ± 0.013 for passive learning and 0.942 ± 0.017 for SASLA. The learning curve in Figure 4 shows that ALUS reached high performance levels with fewer labelled samples than the passive baseline. Statistical testing confirmed the significance of the improvement (p = 0.0039), supporting the superiority of ALUS over passive learning. Furthermore, the better performance of ALUS compared to SASLA is also confirmed to be statistically significant (p = 0.0195).

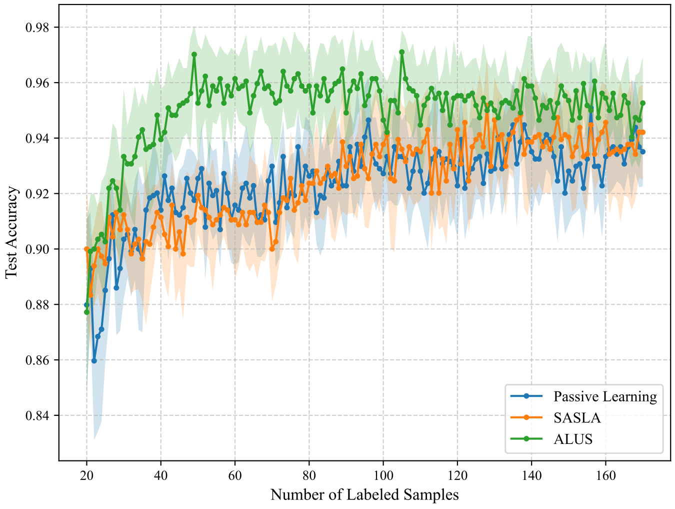

Figure 4. Learning curves for breast cancer dataset.

The effectiveness of ALUS on the binary classification task aligns with the theoretical foundation of the strategy. The conflicting-evidence mechanism excels at identifying instances where the model exhibits strong internal support for competing class predictions. In the context of cancer diagnosis, the instances represent the most informative samples for refining the decision boundary between positive and negative classifications. The ability of the strategy to resolve model uncertainty through targeted querying proved more efficient than the general informativeness heuristic of SASLA for this specific problem. SASLA did also outperform passive learning at the start and end of the experiment, but was inconsistent in middle stages.

2) Wine Dataset: The wine classification task demonstrated nearly equivalent performance between the two active learning strategies. Both SASLA and ALUS achieved final mean accuracies of 0.950 ± 0.037 and 0.947 ± 0.046 respectively, slightly below the passive baseline accuracy of 0.953 ± 0.045. The learning curves in Figure 5 illustrate nearly identical performance progress for the active strategies, rising more steeply and consistently than the passive learning curve. Statistical testing confirmed no significant differences between the performance of the strategies.

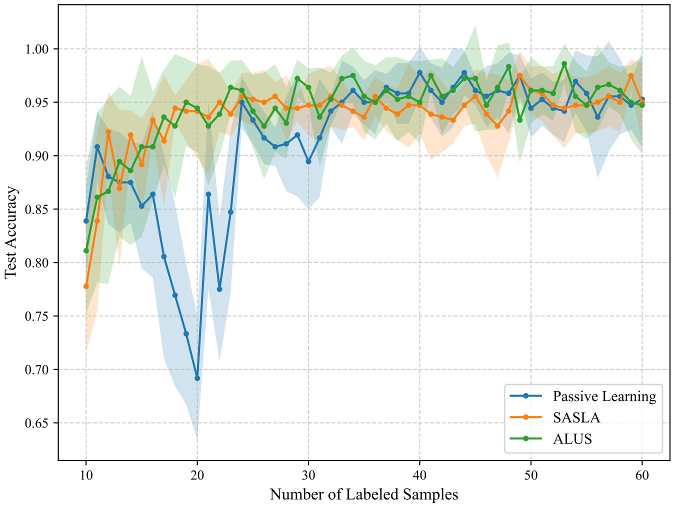

Figure 5. Learning curves for wine dataset.

The convergence in performance suggests that for multi-class problems with moderate complexity, both informativeness-based and uncertainty-based heuristics effectively identify valuable training instances. The three-class structure of the wine dataset provided sufficient complexity to benefit from intelligent sample selection while remaining manageable enough for both strategies to succeed. Passive learning performed especially poorly on this dataset, at least initially. While some lucky initial guesses are made, performance then drastically decreases before again increases. This behaviour is inferior to the consistent gradual increase in performance observed for both active learning strategies.

3) Iris Dataset: The iris classification experiment showed some clear performance differences among the three learning strategies. All methods achieved final accuracies exceeding

0.7, with the passive baseline reaching the lowest at 0.707 ± 0.053, SASLA achieving 0.747 ± 0.045, and ALUS attaining 0.813 ± 0.050. Statistical testing indicated significance for ALUS versus passive learning ($p = 0.0039$) as well as SASLA versus passive ($p = 0.0156$). The performance of passive learning gradually decreases after some good initial lucky guesses, and is consistently outperformed by the active learning strategies after 15 iterations. This effect can be seen in the learning curves in Figure 6.

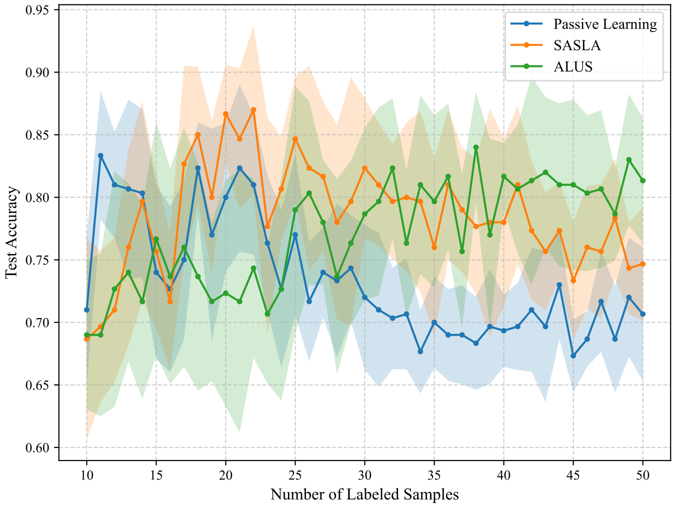

Figure 6. Learning curves for iris dataset.

The iris data is well-known and the classes are easily separable. For this reason, passive learning is initially able to make lucky guesses and achieve comparable performance to active learning strategies without any dramatic rises or falls in performance. However, this effect does balance out, and both active learning strategies eventually outperform passive learning on a statistically significant level.

B. Performance on Regression Tasks

The regression experiments provided a direct comparison between the SASLA strategy and passive learning, as ALUS is specifically designed for classification tasks. The results demonstrated some advantages for the informativeness-based approach across all three regression problems, with the magnitude of improvement varying.

1) Friedman1 Dataset: The Friedman1 synthetic regression problem provided evidence for the effectiveness of SASLA. The strategy achieved a final mean squared error of 7.71 ± 0.08 compared to 8.08 ± 0.62 for passive learning, representing a roughly 5% improvement in predictive accuracy. The smaller variation (std. dev. of 0.08) achieved by SASLA is particularly useful for ensuring good generalisation ability. The learning curves in Figure 7 demonstrate that SASLA reached performance levels with fewer labelled samples than the passive baseline. The performance differences are not extremely pronounced, especially initially. SASLA more consistently outperforms passive learning when 80 or more samples are trained on.

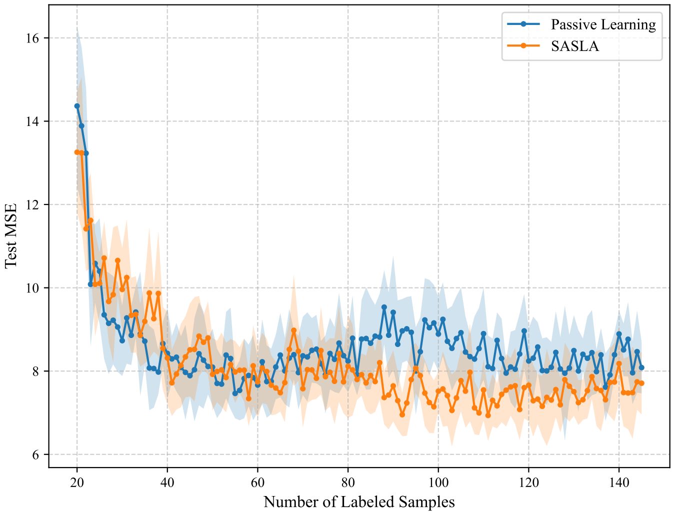

Figure 7. Learning curves for Friedman1 dataset.

The improvement on the dataset can likely be attributed to the core mechanism of SASLA and the inherent structure of the Friedman1 function. The synthetic problem contains complex non-linear relationships and interaction terms that create regions of rapid change in the function. The sensitivity analysis of SASLA effectively identifies the high-gradient regions, guiding the labelling step towards samples that provide the most information about the curvature of the function. The passive strategy, by contrast, allocates samples uniformly across the input space, wasting labels on flat regions where the function varies slowly.

2) Synthetic Regression Dataset: Similar advantages were observed on the synthetic regression dataset, where SASLA achieved a final MSE of 4 066 ± 768 compared to 5 003 ± 1 122 for passive learning. This represents a 19% increase in performance. The learning curves in Figure 8 illustrate inconsistent superiority throughout the training process, indicating varying performance across different stages of model development.

Although SASLA consistently outperformed passive learning in the middle stages (between 90 and 140 samples) this should be interpreted with care. The results on this dataset are particularly volatile, and statistical tests also confirm no clearly superior candidate.

3) Diabetes Dataset: The diabetes regression task revealed more consistent improvements for SASLA. The strategy achieved a final MSE of 3 144 ± 125 compared to 3 214 ± 91 for passive learning, representing a marginal advantage. The learning curves show similar trajectories with SASLA maintaining a slight edge throughout the training process.

The most visible gains by SASLA were made in the early stages, with between 25 and 75 samples. In this interval, the active learning strategy achieves a clearly lower MSE as seen in Figure 9. SASLA is again the consistent winner at

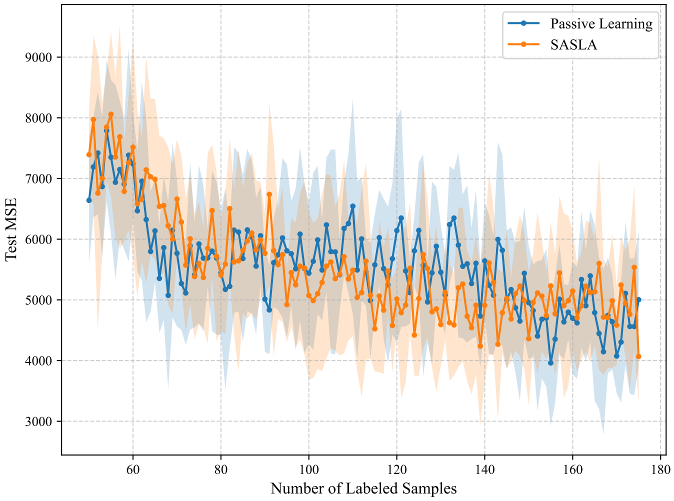

Figure 8. Learning curves for synthetic regression dataset.

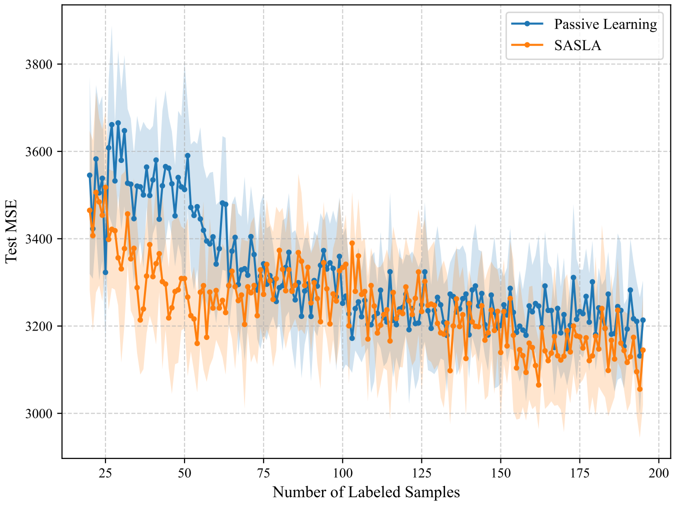

Figure 9. Learning curves for diabetes dataset.

later stages, between 150 and 190 samples. Statistical testing confirmed the significance of the result (p = 0.0488). However, the middle period shows inconsistent results both methods performing similarly. Real-world medical data often contains measurement errors, confounding variables, and individual variation that limit the effectiveness of any sample selection strategy. While SASLA demonstrated superior performance, the practical significance remained limited due to the inherent difficulty of the prediction task.

C. Query Time Analysis

In addition to predictive performance, the computational efficiency of the query selection mechanisms is evaluated. For each dataset and strategy, ten query selections are executed and averaged over five repetitions. The resulting mean query times are reported in Table IX, while the average query time across all datasets for each strategy is summarised in Table X.

Runtime is an important performance metric in active learning. While accuracy or loss values indicate the effectiveness of a strategy, runtime quantifies the practical feasibility of deploying that strategy in real-world systems. High computational overhead during query selection can limit the usefulness of all methods, even if they are accurate.

Table IX
QUERY SELECTION TIME ACROSS DATASETS AND STRATEGIES.

Dataset	Strategy	Mean Query Time (s)	Std. Dev. (s)
Synthetic	Passive Learning	0.00034	0.00077
Synthetic	SASLA	0.18755	0.04541
Diabetes	Passive Learning	0.00000	0.00000
Diabetes	SASLA	0.13914	0.00925
Friedman	Passive Learning	0.00000	0.00000
Friedman	SASLA	0.16046	0.00679
Iris	Passive Learning	0.00000	0.00000
Iris	SASLA	0.08783	0.00351
Iris	ALUS	0.00616	0.00223
Wine	Passive Learning	0.00000	0.00000
Wine	SASLA	0.11631	0.01496
Wine	ALUS	0.00514	0.00064
Cancer	Passive Learning	0.00000	0.00000
Cancer	SASLA	0.28574	0.00969
Cancer	ALUS	0.00419	0.00049

Table X
AVERAGE QUERY SELECTION TIME PER LEARNING STRATEGY.

Strategy	Mean Query Time (s)	Std. Dev. (s)
ALUS	0.00516	0.00112
Passive Learning	0.00006	0.00013
SASLA	0.16284	0.01493

Across all datasets, passive learning had negligible query time (≈ 0.00006 s per query on average), as it involves no model and simply samples chooses at random. The ALUS strategy was significantly more computationally demanding than the passive baseline, with an average query time of 0.0052 s. ALUS still remains efficient enough for real-time use and scales well with pool size.

In contrast, the SASLA approach is the most computationally expensive with an average query time of 0.163 s per query, approximately 30 times times slower than ALUS and nearly 2 700 times slower than random selection. This overhead is mainly due to the computation of a full Jacobian for each candidate sample, which requires separate gradient evaluations across all outputs and input features. The sensitivity analysis performed by SASLA thus provides more information about the model's output landscape but at a substantially higher computational cost.

The results confirm this pattern consistently. For example, SASLA required 0.29 s per query on the breast cancer dataset and 0.19 s on the synthetic regression problem, while ALUS remained around 0.004-0.006 s per query on the classification datasets. The lower variance of ALUS across datasets also indicates stable computational scaling compared to SASLA, whose cost grows more rapidly with both dimensionality and model size.

Overall, these findings demonstrate a clear trade-off between computational cost and selection detail. While SASLA offers potentially more informative queries due to its gradient-based sensitivity analysis, its computational cost might limit its use in cases where large-scale data processing is required. ALUS achieves a balanced compromise between efficiency and informativeness, making it more suitable for practical learning applications where time constraints are present.

D. Statistical Evaluation

To compare the performance of the active learning strategies (SASLA and ALUS) against the passive learning baseline, the Wilcoxon signed-rank test is employed for pairwise comparisons.

This non-parametric test is selected for two primary reasons based on the experimental design:

1) Paired Samples: The experiment compares the final performance scores from multiple trials (N=10). As is standard practice in machine learning comparisons, each trial for an active learning strategy is run under the same initial conditions as the corresponding trial for the passive baseline. This creates a paired sample design, making the Wilcoxon test the preferred choice over independent sample tests (such as the Mann-Whitney U test or independent t-test).
2) Non-Parametric Nature: Given the relatively small number of trials, it cannot reliably be assumed that the distribution of performance differences between the paired scores follows a normal distribution. The Wilcoxon test is non-parametric, meaning it makes no such distributional assumptions, providing the most robust and statistically sound conclusion for this scale of experiment.

Statistical testing provided mixed support for the observed performance differences. The results reaching statistical significance ($p<0.05$) confirm the reliability of the improvements on those respective datasets, such as ALUS achieving significant performance gains over passive learning and SASLA on the breast cancer dataset. Both active learning strategies achieved statistically significant results over passive learning on the iris dataset. However, most other comparisons failed to reach statistical significance despite consistent performance trends, which often reflects the high inherent variability in neural network training combined with the limited sample size. Larger-scale experiments would be required to establish clear statistical conclusions for marginal performance differences.

All results for the statistical test are provided in Table XI. The regression datasets only show one pairwise comparison, between SASLA and passive learning.

VII. CONCLUSION

The investigation compared passive learning with two separate active learning strategies, the informativeness-based SASLA and the uncertainty-based ALUS, for training feedforward neural networks. The evaluation across three classification and three regression benchmark datasets produced several important findings.

Table XI
RESULTS OF THE WILCOXON SIGNED-RANK TEST.

Dataset	Comparison	p-Value	$p<0.05$
Breast Cancer	SASLA vs. Passive	0.1816	No
Breast Cancer	ALUS vs. Passive	0.0039	Yes
Breast Cancer	ALUS vs. SASLA	0.0195	Yes
Diabetes	SASLA vs. Passive	0.0488	Yes
Friedman1	SASLA vs. Passive	0.2754	No
Iris	SASLA vs. Passive	0.0156	Yes
Iris	ALUS vs. Passive	0.0039	Yes
Iris	ALUS vs. SASLA	0.1230	No
Synth. Regression	SASLA vs. Passive	0.8457	No
Wine	SASLA vs. Passive	0.8125	No
Wine	ALUS vs. Passive	0.4258	No
Wine	ALUS vs. SASLA	0.5156	No

The effectiveness of active learning proved highly context-dependent, with no single strategy demonstrating superiority across all experiments. The finding aligns with the *No Free Lunch* theorem and emphasises the importance of matching query strategies to specific problem characteristics. Simple tasks with well-separated classes or smooth functions showed smaller benefit from advanced sample selection, as random sampling provided sufficient representation of the training data. Complex tasks with non-trivial decision boundaries or extreme function variation demonstrated more significant improvements from intelligent query strategies.

The appropriate pairing of the strategy to the task emerged as the critical factor determining active learning success. The ALUS strategy excelled on classification problems, particularly binary tasks where conflicting-evidence decisions provided valuable information for decision boundary evolution. The strategy achieved statistically significant improvements on breast cancer classification and demonstrated consistent performance advantages on multi-class wine classification. The SASLA strategy proved effective for regression tasks, where sensitivity analysis successfully identified high-gradient regions that required detailed function approximation. The success of SASLA is however less consistent and should be interpreted for each dataset individually.

The efficiency gains tended to be most evident during early training phases, where intelligent sample selection accelerated initial model development. The learning curves consistently showed steeper improvement rates for active strategies on challenging tasks, confirming the main active learning hypothesis.

Statistical significance testing provided mixed support for the observed performance differences. Clearly significant results were achieved only on the iris and breast cancer classification tasks as well as the diabetes regression task. The limited statistical power reflects the inherent variation found in neural network training and the constraints of the scale of the experiment. However, consistent directional improvements over independent trials suggest actual strategy benefits and not just random variation.

The implementation of active learning strategies is highly unlikely to lead to less efficient learning. The inductive biases of SASLA and ALUS ensure new information is maximised at each step. However, the efficiency increase varies per dataset and hyperparameter configuration. It is likely that the implementer will consider a tradeoff between learning efficiency and runtime. For this reason, all criteria need to be considered before deciding on a solution. In the case of runtime being a lesser priority, active learning should be strongly considered. However, if runtime minimisation is a central objective, the choice of learning strategy depends strongly on the specific dataset. SASLA provided the most detailed sensitivity-based query selection but ran for up to 30 times longer than ALUS.

Future work could explore hybrid strategies that adaptively combine uncertainty and sensitivity-based criteria, as well as extensions to deeper architectures or convolutional networks. Additionally, incorporating cost-aware query selection could make active learning more practical for real-world applications where labelling costs vary.

In conclusion, the study confirmed that active learning can provide meaningful improvements in neural network training efficiency, but success depends on appropriate strategy selection and problem characteristics. The informativeness-based SASLA strategy proved to be valuable for complex regression tasks, while the uncertainty-based ALUS strategy performed well on specific classification problems. However, the benefits must be considered against computational overhead and implementation complexity, with the most attractive applications being in domains where labelling costs are significantly higher than computation costs. ALUS should be preferred over SASLA for runtime-sensitive implementations.

References

[1] M. Sharma and M. Bilgic, "Evidence-based uncertainty sampling for active learning," *Data Mining and Knowledge Discovery*, vol. 31, pp. 164–202, 2017.

[2] A. P. Engelbrecht, "Sensitivity Analysis for Selective Learning by Feedforward Neural Networks," *Fundamenta Informaticae*, vol. 45, pp. 295–328, 2001.

[3] B. Settles, "Active learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 6, no. 1, 2012.

[4] M. F. Møller, "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning," *Neural Networks*, vol. 6, pp. 525–533, 1993.

[5] D. Cohn, L. Atlas, and R. Ladner, "Improving Generalization with Active Learning," *Machine Learning*, vol. 15, pp. 201–221, 1994.