Active Learning in Neural Networks

DT Nicolay 26296918
Computer Science Division
Stellenbosch University
Stellenbosch, South Africa
26296918@sun.ac.za

***Abstract*—Passive learning approaches are sometimes inefficient, since they assume that all training observations contribute equally. Active learning provides a solution to this by deciding which observations are important and rather learning on those first. This paper compares two active learning approaches, output sensitivity and uncertainty sampling, against passive learning. A single hidden-layer neural network is considered across six problems of varying complexity. Three classification problems and three function approximations are run over 50 trial runs to analyse performance. Active learning shows mixed results which depend on the nature and complexity of the dataset. For small datasets, all methods perform equivalently. For larger datasets, the passive learning and output sensitivity approaches outperform uncertainty-based methods. The uncertainty-based methods do however show more efficient performance in early stages of training. Computational overhead of active learning approaches must be weighed against sample efficiency gains. This paper provides empirical evidence for when active learning approaches are more beneficial than passive learning.**

***Index Terms*—active learning, neural networks, uncertainty sampling, output sensitivity analysis, sample efficiency, supervised learning, stochastic gradient descent, pattern informativeness, entropy-based selection, ensemble methods**

I. Introduction

Traditional passive learning approaches in supervised learning assume that all training observations contribute equally to the learning process. However, this assumption does not always hold in practice, leading to inefficient computations. Active learning is a method of obtaining predictive models with high precision at a limited cost through the adaptive selection of samples for labelling [1]. This study implements and compares two active learning approaches, output sensitivity analysis [2] and uncertainty sampling [3], against traditional passive learning. The comparison of the performance of these approaches is performed on three classification and three function approximation problems of varying complexity.

The passive and active learning approaches are implemented using a neural network architecture with one hidden layer. The control parameters of the neural network are optimised through a grid search. The performance of the various algorithms is evaluated based on accuracy, convergence rate, and sample efficiency. Notably, the models are compared based on the number of training observations seen during training. This is done to illustrate the efficiency of the active learning approaches.

This paper contributes to the understanding of active learning's practical applicability by providing an empirical comparison of selection strategies across diverse problem domains. While active learning theory suggests significant efficiency gains through intelligent sample selection, its real-world effectiveness depends critically on problem characteristics, dataset size, and computational constraints. Through comprehensive statistical analysis including omnibus testing, pairwise comparisons with Bonferroni correction, and effect size measurements, this study reveals when active learning provides genuine advantages over traditional passive approaches and when the added complexity fails to justify the computational overhead.

The theoretical background of active learning is presented in section 2, while section 3 walks through the implementation of the two active learning approaches with a neural network architecture. Section 4A describes the dataset preparation, followed by the performance criteria in section 4B. Section 4C and 4D outline the experimental design and framework for analysis. Section 5 summarises the results of the various algorithms and discusses their comparative performances with statistical analyses.

II. Background

The active learning paradigm and various approaches to active learning are discussed in this section. Active learning is described by assessing how it is useful and the manner in which it should be applied.

A. Passive Learning

Traditional supervised learning approaches perform passive learning during training where the algorithm has no control over which training samples it receives. That is, when techniques such as Stochastic Gradient Descent (SGD) are used, samples of the data are taken at random to update model parameters iteratively. SGD represents a form of passive learning since the model does not *choose* the data points it learns from. This approach is interpretable and easy to implement, especially since there is no overhead required for observation selection.

Mathematical updates during SGD in passive learning can be expressed as:

$$w_{t+1} = w_t - \eta \nabla_w L(x_i, y_i; w_t)$$

where η is the learning rate, (x_i, y_i) is a randomly sampled observation from the training data at iteration t and L represents the loss function.

Random sampling is unbiased, but potentially inefficient in the presence of high noise and variance [4]. Passive learning assumes that the distribution of the training data matches that of the test distribution, and that all the training observations contribute equally. This assumption does not always hold in practice, leading to inefficient computations. More specifically, the algorithm may waste effort on uninformative observations.

B. Active Learning Paradigm

In supervised learning, acquiring labelled training data for a predictive model can be very costly, but acquiring a large amount of unlabeled data is often quite easy. Active learning is a method of obtaining predictive models with high precision at a limited cost through the adaptive selection of samples for labelling [1]. Rather than passively accepting training examples from the teacher, the network is allowed to use its current knowledge about the problem to have some deterministic control over training examples, and to guide the search for informative patterns [2]. This may result in shorter training times due to fewer observations needing to be considered by the algorithm. That is, if the complexity of the observation selection approach does not exceed the reduction in training time achieved by considering fewer observations. Therefore, careful consideration is required to decide on an approach to observation selection.

C. Output Sensitivity Analysis

For this approach, begin by considering the entire training set and iteratively remove observations that are shown to be less informative. The neural network uses its learned knowledge of the distribution at each selection interval to do so.

The core of this algorithm is based on pattern informativeness. The following definitions and the corresponding mathematical formulation follow the approach introduced by Engelbrecht [2]. An informative pattern is defined as one that has a strong influence of the neural network outputs, whilst an uninformative pattern has a negligible effect. That is, the informativeness of a pattern is the sensitivity of the neural network output vector to small perturbations in the input vector. Denote the informativeness of pattern p as $\Phi^{(p)}$. Then,

$$\Phi^{(p)} = \|\mathbf{S}_o^{(p)}\|, \tag{1}$$

where $\mathbf{S}_o^{(p)}$ is the output sensitivity vector for pattern p (defined in (3)), and $\|\cdot\|$ is any suitable norm. Consider the maximum-norm:

$$\Phi(_\infty p) = \|\mathbf{S}_o^{(p)}\|_\infty = \max_{k=1,\dots,K}\{\left|S_{o,k}^{(p)}\right|\}, \tag{2}$$

where $S_{o,k}^{(p)}$ refers to the sensitivity of a single output unit o_k to changes in the input vector $\mathbf{z}$.

The output sensitivity vector is defined as

$$\mathbf{S}_o^{(p)} = \|\mathbf{S}_{oz}^{(p)}\|, \tag{3}$$

where $\mathbf{S}_{oz}^{(p)}$ is the output–input layer sensitivity matrix. Assuming sigmoid activation functions in both the hidden and output layers, each element $S_{oz,ki}^{(p)}$ of the sensitivity matrix is computed as

$$S_{oz,ki}^{(p)} = o_k^{(p)}\big(1-o_k^{(p)}\big)\sum_{j=1}^{J} w_{kj}\, y_j^{(p)}\big(1-y_j^{(p)}\big)v_{ji}, \tag{4}$$

where w_{kj} is the weight between output unit o_k and hidden unit y_j, v_{ji} is the weight between hidden unit y_j and input unit z_i, $o_k^{(p)}$ is the activation value of output o_k, $y_j^{(p)}$ is the activation of hidden unit y_j, and J is the total number of hidden units (including the bias unit to the output layer).

Suitable norms for calculating the output sensitivity vector include the sum-norm or the Euclidean norm, i.e.,

$$S_{o,k}^{(p)} = \|S_{oz}^{(p)}\|_1 = \sum_{i=1}^{I}\left|S_{oz,ki}^{(p)}\right|, \tag{5}$$

or

$$S_{o,k}^{(p)} = \|S_{oz}^{(p)}\|_2 = \sqrt{\sum_{i=1}^{I}\big(S_{oz,ki}^{(p)}\big)^2}, \tag{6}$$

where I is the total number of input units (including the bias unit to the hidden layer).

Using Equation 2, a pattern is considered *informative* if one or more of the output units is sensitive to small perturbations in the input vector. The larger the value of $\Phi_1^{(p)}$, the more informative the pattern p is.

To illustrate, assume gradient descent is used to find optimal weight values:

$$\Delta w_{kj},\, \Delta v_{ji} \propto \big(t_k^{(p)} - o_k^{(p)}\big), \tag{7}$$

where t_k is the target value for output unit o_k for pattern p. Each new pattern can be viewed as a perturbation of a previously presented pattern. Let $\Phi_\infty^{(p)} = \left|S_{o,k}^{(p)}\right|$. If $\Phi_\infty^{(p)}$ is large, the output value of o_k changes significantly compared to the previous presentation, making pattern p highly informative. Conversely, if $\Phi_\infty^{(p)}$ is small, no significant change in o_k occurs, and pattern p is an insignificant contributor to the gradient direction, thus uninformative for the learning process.

D. Uncertainty Sampling

Uncertainty sampling, a frequently utilised active learning strategy, selects instances about which the model is uncertain but it does not consider the reasons for why the model is uncertain [3]. There are two reasons that a model may be uncertain about an observation. Considering a classification scenario, *conflicting-evidence uncertainty* is where there is strong conflicting evidence for multiple classes. On the other hand, *insufficient-evidence uncertainty* describes the situation where there is not enough evidence to classify an observation to any class. Uncertain instances often lie close to the decision boundary. Understandably, a model's variance on *conflicting-evidence uncertainty* is higher than *insufficient-evidence uncertainty*.

In contrast with the output sensitivity approach, begin with an empty set of observations and iteratively include the most uncertain observations. That is, to select an informative instance $\langle x^*, ?\rangle \in U$ and incorporate the new labelled instance $\langle x^*, y\rangle$ into L. More formally, Algorithm 1 was presented by Sharma and Bilgic [3], and describes the process of building the subset.

Algorithm 1 Pool-Based Active Learning

```
1: Input: $U$ - unlabeled data, $L$ - labeled data, $\theta$ - classification model, $B$ - budget
2: repeat
3:     for all $\langle x^{(i)}, ?\rangle_i \in U$ do
4:         compute utility$(x^{(i)}, \theta)$
5:     end for
6:     pick highest utility $x^*$ and query its label
7:     $L \leftarrow L \cup \{\langle x^*, y^*\rangle\}$
8:     $U \leftarrow U \setminus \{\langle x^*, y^*\rangle\}$
9:     Train $\theta$ on $L$
10:    $B = B - 1$
11: until $B == 0$
```

This study considers two ways to quantify uncertainty. First, given a probabilistic classifier that outputs a distribution over classes $P(y|x, \mathbf{w})$, the predictive entropy of an observation x is defined as

$$H(x) = -\sum_{c=1}^{C} P(y = c \mid x, \mathbf{w}) \log P(y = c \mid x, \mathbf{w}). \quad (8)$$

This approach has been widely used in literature [5]–[7], however often along with other techniques.

A second approach to measuring uncertainty involves using model ensembles. Instead of relying on a single model's output, construct an ensemble of models (via bootstrap sampling, random initialisations, or subspace sampling), and measure the disagreement among them. That is, for k models, compute

$$\mathrm{Var}_k(P(y \mid x, \mathbf{w}_k)), \quad (9)$$

where $\bar{P}(y \mid x)$ is the mean predictive distribution across the models. Empirically, ensemble methods often provide more reliable uncertainty estimates than single-model entropy, especially in settings with model misspecification or limited data [8].

III. Implementation

The approach to implementing the algorithms is described in this section.

A. Neural Network Architecture

The neural network architecture is composed of one hidden layer, of which the number of hidden units is determined through a grid search for each problem. Figure 1 illustrates an overview of the architecture. Each hidden layer has a sigmoid activation function. However, only the classification tasks use a sigmoid activation function for the output units. Weight decay, specifically L2 regularisation, is considered along with various values for momentum.

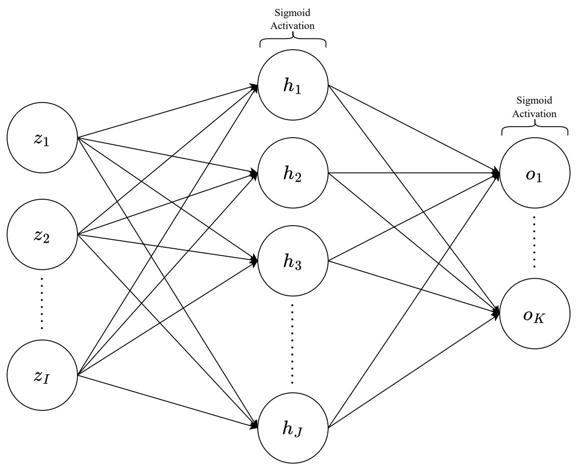


Fig. 1: Neural network architecture. Here, z_i represents input units, h_i represents hidden units, and o_i represents output units.

The grid search is only performed once using the MSE Loss given in Equation 10 in the passive learning setting. Here, N represents the number of training patterns, K represents the number of output units, $t_k^{(i)}$ is the target value for output unit k in pattern i, $o_k^{(i)}$ is the network output of unit k in pattern i, w_j is the weight parameter and λ is the weight decay coefficient. The same parameter values obtained from the passive learning grid search are used in the active learning procedures in order to maintain the focus of the comparison on the algorithms and not the parameters. SGD is used to determine the optimal parameter values.

$$L_{\text{MSE+WD}} = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}\left(t_k^{(i)} - o_k^{(i)}\right)^2 + \lambda\sum_{j} w_j^2 \quad (10)$$

B. Output Sensitivity Algorithm

The output sensitivity active learning implementation follows the background derivations previously described. A conservative pattern selection constant, $\beta = 0.9$ is used. That is, the algorithm only picks the most informative samples, keeping the training subset small at first.

The algorithm operates by repeatedly training the neural network on training subset D_T until a termination criterion on D_T is triggered. In this implementation, a budget of 1000 epochs is used. The exact gradient-based output sensitivity is computed using Equation 4. The pattern informativeness is then computed using both sum-norm and max-norm metrics in Equation 1 and 2. The average pattern informativeness is calculated and multiplied by $(1 - \beta)$ to determine the selection threshold. The patterns with informativeness above this threshold are selected from the candidate set. A safety mechanism was implemented here to ensure that at least one sample per class is kept. Initially these calculations were performed linearly, however vectorising them results in a massive improvement in performance. A selection interval parameter is used to determine after how many epochs the selection takes place, this is set to one unless otherwise stated.

C. Uncertainty Sampling Algorithm

Both entropy and ensemble-based uncertainty sampling were implemented. The Pool-Based Active Learning Framework described in Algorithm 1 is implemented. First, the algorithm starts with a small labelled set L and large unlabelled pool U. The initial labelled set is defined as the smaller of 10% of the training data or 10 observations. Then, for each query iteration, compute the utility (uncertainty) for all the samples in U. This is computed using the softmax of the output units,to obtain probabilities for use in the equation Equation 8. Further normalise the raw entropy obtained from Equation 8 by dividing by $\log(C)$, this is to ensure that the entropy is in the range [0, 1]. The highest utility sample is selected and its label queried, then moved from U to L. Now, simply retrain the model on the updated L set and repeat this process until the budget is exhausted. Here, the budget is defined as the number of remaining samples.

For the ensemble-based approach, three independent neural networks are trained simultaneously with different random weight initialisations to create diversity in their learned representations. Rather than using entropy of a single model's output distribution as the uncertainty metric, the disagreement among ensemble members is exploited. Specifically, for each unlabelled sample $\mathbf{x}_i \in U$, perform forward passes through all ensemble members to obtain softmax-normalised probability distributions $\{\mathbf{p}_i^{(1)}, \mathbf{p}_i^{(2)}, ..., \mathbf{p}_i^{(n_{ensemble})}\}$. The uncertainty is then quantified as the total predictive variance across the ensemble using Equation 9. Higher variance indicates greater disagreement among models, suggesting the sample lies in an ambiguous region of the feature space where additional labelled data would be most informative. During training, all ensemble members are updated synchronously on the same labelled set L using independent optimisers, and final predictions are made via majority voting across the ensemble. This approach trades increased computational cost (linear in $n_{ensemble}$) for potentially better-calibrated uncertainty estimates, particularly valuable when dealing with limited labelled data where single-model confidence may be poorly calibrated.

D. Software and Hardware Specifications

All experiments were performed on a Fedora Linux system with a 13th Gen Intel Core i3-13100 x8 CPU and 16GiB of memory. The code for this paper is available on GitHub [9].

IV. Empirical Process

The datasets along with their respective pre-processing measures are presented in this section, followed by a summary of the performance criteria and the experimental design. Finally, the statistical analysis framework is described.

A. Dataset Selection and Pre-processing

Three classification and regression problems were selected. One simple, one medium, and one hard problem were selected for each to illustrate how the algorithms apply to varying problem complexities.

The iris dataset was used as the simplest classification problem with four predictors and three target classes. The distribution of the classes is noted in Table I. The wine cultivar dataset was used as a dataset of medium complexity. It contains 13 feature variables and a target variable of three classes representing different wine cultivars. The fashion-MNIST dataset is a more complex dataset containing pixel information from 28×28 grey scale images. There are 10 equally distributed target classes each representing different articles of clothing.

An artificial function approximation task was created using

$$y = \sin(2\pi x) + \text{noise}$$

where the noise is Gaussian noise centred at 0 with a standard deviation of 0.1. The California housing dataset is a medium complexity function approximation problem with nine features and the target variable is the median house value. The energy efficiency dataset is the final function approximation task considered with eight features used to predict the heating load of a building.

TABLE I: Dataset Descriptions

Dataset	**Distribution / Notes**	**Source**
Iris	setosa – 33.3% versicolor – 33.3% virginica – 33.3%	[10]
Wine	cultivar 0 – 33.1% cultivar 1 – 39.9% cultivar 2 – 27.0%	[11]
Fashion-MNIST	10 equally distributed classes	[12]
Sine + Noise	sine function + Gaussian noise	synthetic
California Housing	-	[13]
Energy Efficiency	-	[14]

Only the California housing dataset contained 207 missing values in the `total_bedrooms` column. These missing values were imputed with the median for a safe estimate that is less sensitive to outliers. None of the other datasets contained missing values.

The iris dataset contained an ID feature which was dropped since it provides no predictive power to the model.

All the features were scaled to the range $[-1, 1]$ since the hidden layer uses the sigmoid activation function. If unscaled inputs were fed into the model, the linear outputs could be very large for positive or negative numbers. This would saturate the sigmoid function leading to vanishing gradients. The function approximation tasks' target variables were scaled to bring the values into a range consistent with the network's activations and makes optimisation much more stable. Obviously, at prediction time, the scaling is inverted.

For the iris, wine, and fashion-MNIST datasets the target variables were one-hot encoded so the neural network can provide a probability distribution across classes. The housing dataset had one categorical feature, `ocean_proximity`, which was one-hot encoded so that the neural network could interpret the feature levels without assuming an ordinal relationship.

In order for the frameworks to be applied to the various problems, the variables that were changed were: the input and output dimensions of the neural network, number of epochs, and for regression the output activation function was removed. The size of the datasets and number of epochs are presented in Table II. A test train split of 20/80 was considered for all approaches and datasets.

TABLE II: Dataset sizes and number of epochs used for training

Dataset	**Size**	**Epochs**
Iris	150	1000
Wine	178	2000
Fashion-MNIST	70,000	1000
Sine + Noise	500	1000
Housing	20,640	200
Energy	768	200

B. Performance Criteria

For the classification tasks, accuracy was the primary metric used for model performance evaluation across all learning strategies. The F1-Score was also recorded and considered for a more comprehensive, balanced metric. The precision and recall were also tracked. In each training iteration, the number of training observations seen by the model are recorded as it trains. The training set size reduction for the active learning methods was recorded across the epochs. Epochs to convergence was recorded for each method, that is, the number of training epochs required for the model to achieve a stable performance. The convergence rate was computed by calculating the proportion of the trials achieving the convergence threshold.

For the function approximation tasks, Mean Squared Error (MSE) served as the primary performance metric, directly measuring the average squared deviation between predicted and actual target values. Similar to classification tasks, the number of training observations presented to the model was tracked throughout training to assess sample efficiency. Training and validation losses were monitored across epochs to evaluate convergence behaviour and detect potential overfitting or underfitting patterns.

C. Experimental Design

Due to the stochastic nature of the weight initialisation process, 50 trial runs were performed for each method for statistical reliability. Each run included five-fold stratified cross-validation. That is, the distribution of the target classes in the original data is maintained in the training and validation sets. In order to optimise the control parameters, a grid search was performed. First, the hidden layer size was varied across $\{64, 128, 256, 512\}$, allowing the model capacity to range from relatively compact (64 units) to more expressive (512 units). The learning rate was chosen from $\{0.01, 0.05, 0.1, 0.5\}$, where smaller values yield slower but stable convergence, and larger values speed up training at the risk of overshooting minima. To control overfitting, the following weight decay values $\{0.0, 0.001, 0.01, 0.1\}$ were considered, where higher values apply stronger L_2 regularisation on the weights. Finally, momentum was introduced with values $\{0.0, 0.5, 0.9, 0.95\}$, ranging from standard stochastic gradient descent (no momentum) to strongly accelerated updates (high momentum). Since each parameter is considered independently, the total number of grid search configurations is the Cartesian product of these sets, i.e., $4 \times 4 \times 4 \times 4 = 256$ distinct control parameter combinations. The grid search was performed on the passive learning model setup, and the parameters obtained were then applied to the active learning approaches since the primary focus of this study is to compare the active learning techniques, and not the control parameters.

D. Statistical Analysis Framework

To ensure a rigorous comparison of the learning approaches, a comprehensive statistical framework was implemented across multiple levels of analysis.

1) Normality Assessment: Normality was first assessed to determine whether the model results come from a normal distribution. That is, the Shapiro-Wilk test provided insight as to consult parametric or non-parametric tests. Some of the further statistical tests considered have a normality assumption, and therefore cannot be considered for non-normal results.

2) Omnibus Testing: The Friedman test [15] is the non-parametric equivalent of repeated measures ANOVA and has the null hypothesis that all the methods performed identically. The test ranked methods within each trial and then compared the average rank with a significance level of $\alpha = 0.05$. If $p < 0.05$, the analysis proceeded to post-hoc pairwise tests, if $p \geq 0.05$, no significant differences existed.

3) Pairwise Comparisons: The Wilcoxon signed-rank test was applied as a non-parametric paired comparison test. Since four methods were compared, this resulted in $\binom{4}{2} = 6$ pairwise comparisons. To control for Type I error inflation due to multiple comparisons, the Bonferroni correction was applied [16], adjusting the significance threshold to $\alpha_{adjusted} = 0.05/6 \approx 0.0083$. Without this correction, the probability of at least one false positive across all comparisons would be approximately $1 - 0.95^6 \approx 0.26$. Methods were considered significantly different only if $p < \alpha_{adjusted}$.

4) Critical Difference Analysis: Critical difference plots were created to provide a visual representation of all the pairwise comparison tests simultaneously. The methods are ranked from one to four within each trial and averaged over all 50 trials. Methods with a lower average rank are said to have better performance. The critical difference (CD) was calculated using the Nemenyi post-hoc test [18] as $\text{CD} = q_\alpha \sqrt{k(k+1)/(6N)}$, where $k = 4$ methods, $N = 50$ trials, and q_α is the critical value from the Studentized range distribution at $\alpha = 0.05$. Methods differing by more than CD in average rank are considered significantly different, with connected horizontal bars indicating non-significant differences.

5) Significance Levels and Multiple Testing: All hypothesis tests employed a significance level of $\alpha = 0.05$. The family-wise error rate across the six pairwise comparisons was controlled through Bonferroni correction, ensuring the overall Type I error probability remained at 0.05. This conservative approach reduces the risk of false positive findings when conducting multiple statistical tests on the same dataset.

V. Results & Discussion

The metrics obtained from the various approaches on the six datasets are presented in this section. The performance of the passive and active learning approaches are also compared by explaining the accuracies and training statistics.

A. Control Parameter Optimisation Results

The grid search obtained a variety of results as exhibited in Table III. Surprisingly, the weight decay parameter was only used in two of the datasets. The momentum values remained quite similar at around 0.9 whilst most of the learning rate parameters also remained at 0.5 except for the wine dataset. The number of hidden units was often small due to the power of neural networks being able to approximate functions even with a smaller number of hidden units, especially when the functions were less complex.

TABLE III: Parameters obtained from grid search

Dataset	**Hidden Units**	**Weight Decay**	**Learning Rate**	**Momentum**
Iris	512	0.1	0.5	0.9
Wine	64	0	0.01	0.9
Fashion-MNIST	512	0	0.5	0.95
Sine + Noise	64	0.1	0.5	0.95
Housing	64	0	0.5	0.95
Energy	64	0	0.5	0.9

B. Classification Problem Results

For the simple iris dataset, all the methods performed similarly, as shown in Table V. All the approaches maintained a stable performance with the standard deviations remaining low. Slightly more drop off in performance is seen as the complexity of the dataset was increased. That is, for the fashion-MNIST dataset the entropy and the ensemble uncertainty active learning approaches had a small, but noticeable, decrease in performance compared to the passive learning approach.

Figure 2 indicates that both the passive learning and SASLA approaches were stable during training and not exhibiting any noticeable overfitting behaviour. The entropy and ensemble uncertainty sampling approaches were slightly less stable, but still converged well after the full 1000 epochs had been performed. This is understandable, since the training sets start off small and therefore more variability can be expected. Similar trends are observed in Figure 3, however, with slightly less variability for the uncertainty sampling approaches. For the fashion-MNIST dataset in Figure 4, the validation losses appear to land higher than the training losses after the 1000 epochs. This gap suggests that the models may be under-regularised or struggling to generalise fully to the more complex and diverse patterns in the fashion-MNIST dataset.

The strength of the entropy uncertainty sampling can be seen in Figure 8, where the generalisation factor exceeds that of the passive learning approach at roughly 2000 epochs. This indicates that this uncertainty sampling approach was efficient at selecting useful samples for training at early stages. This highlights the advantage of active learning, as it is able to prioritise informative samples that accelerate model improvement compared to random selection. However, this is not the case for all datasets, as Figure 9 and Figure 10 show that for bigger datasets, considering all samples is more effective at achieving a better overall performance.

Again in Figure 13, the entropy uncertainty sampling performs well at a low number of epochs. Figure 14 shows that the entropy uncertainty sampling found some super informative observations early on which lead to it improving performance, however as it added more less informative observations, the F1-score then decreased again to align with the other approaches. The SASLA approach can get led in the wrong direction and the performance can decrease, this is seen in Figure 15 where the F1-score decreases noticeably after 500 epochs.

C. Function Approximation Results

For function approximation tasks, the SASLA and entropy uncertainty sampling approaches performed worse when applied to smaller datasets, specifically the synthetic dataset in Table VI. The passive approach achieved the lowest error, while SASLA and entropy both introduced a noticeable increase in MSE, with entropy uncertainty in particular exhibiting a very high variance across runs. Ensemble uncertainty fell in between, outperforming SASLA and entropy but still lagging behind passive learning.

For the housing dataset, the performance of all methods converged, with only minor differences in MSE and relatively low variance across runs. This suggests that for a moderately sized real-world dataset, the choice of active learning strategy had limited impact compared to passive sampling.

In contrast, for the energy dataset, clear divergence emerged between the strategies. While SASLA and passive approaches remained close in performance, entropy uncertainty once again performed poorly, showing both high error and instability. Ensemble uncertainty also underperformed relative to passive learning, but achieved significantly better results than entropy-based sampling.

In general, few epochs were required for the regression tasks to converge clearly. Figure 5a shows that all the methods converged in about 200 epochs. The trend is seen more closely in Figure 6a, where the entropy and ensemble uncertainty approaches took about double the number of epochs to converge compared to the passive and SASLA approaches. That is, they converged after about 100 epochs compared to 50 for the other two approaches. The SASLA approach varied significantly during early stages of training in Figure 7a. This instability in SASLA's early training phase likely stems from its reliance on the adaptive sampling, which can introduce high variance in gradient updates before the model has established a stable representation.

The uncertainty approaches showed their true power in Figure 12 where we see they achieved an impressive generalisation factor early in training when only a few patterns had been presented to them. The passive and SASLA approaches more observations to achieve a similar performance. This is expected, as the models were able to find informative samples early on through the active sampling process.

D. Comparative Analysis

Computational efficiency varied considerably across methods and datasets, as shown in Table VII. Ensemble uncertainty consistently required the longest training times due to maintaining multiple models, particularly evident in the housing dataset (12.64s vs 0.60s for passive). SASLA incurred moderate overhead from sensitivity calculations, approximately 1.5× slower than passive learning on average. Entropy uncertainty sampling showed mixed results, where it was faster than passive on some datasets (synthetic: 0.95s vs 0.97s) but substantially slower on others (housing: 2.43s vs 0.60s), likely due to the cost of computing entropy over large unlabelled pools. For smaller datasets like iris and wine, the computational overhead of active learning was minimal in absolute terms, though proportionally significant. These results suggest that while active learning methods can reduce sample complexity, practitioners must weigh potential gains in sample efficiency against increased computational costs, particularly when using ensemble-based approaches or processing large unlabelled pools.

For small datasets, the models had equivalent performance, based on the critical difference plot in Figure 16 for the iris dataset. The critical difference plot shows that while ensemble uncertainty achieved the best average rank (2.39) and uncertainty sampling the worst (2.64), all four methods are connected by horizontal lines indicating no statistically significant differences between them at the 0.05 level. This means that despite numerical differences in rankings, all methods perform equivalently from a statistical perspective.

In contrast, the plot for the fashion-MNIST dataset in Figure 17 shows the simpler approaches (passive learning and SASLA) significantly outperform the more complex active learning methods (entropy uncertainty sampling and ensemble uncertainty), with ensemble uncertainty performing worst overall. Here, the passive learning and SASLA approaches are not significantly different from each other, since the difference of 0.09 is less than the critical difference of 0.663. The SASLA and passive learning approaches exhibited similar behaviour in the energy efficiency dataset seen in Figure 20.

For the synthetic dataset, the passive learning showed to be different from the active learning approaches. This is observed in Figure 18.

Table IV confirms significant differences exist among the compared methods across all four datasets (fashion, synthetic, housing, and energy). Both parametric tests (ANOVA F, Friedman χ^2) and non-parametric tests (Kruskal-Wallis H) consistently yield p-values below 0.001, which is well below the $\alpha = 0.05$ significance threshold. These results justify proceeding with post-hoc pairwise comparisons (like the critical difference plots) to identify which specific methods differ from each other, as the omnibus tests confirm that not all methods perform equally.

TABLE IV: Omnibus Test Results

Dataset	**Test**	**Statistic**	**p-value**
Fashion	Friedman χ^2	125.80	< 0.001
	ANOVA F	27.87	< 0.001
Synthetic	Friedman χ^2	62.38	< 0.001
	Kruskal-Wallis H	–	< 0.001
Housing	Friedman χ^2	85.75	< 0.001
	Kruskal-Wallis H	104.07	< 0.001
Energy	Friedman χ^2	110.33	< 0.001
	Kruskal-Wallis H	140.71	< 0.001

All tests indicate significant differences among methods at $\alpha = 0.05$.

VI. Conclusions

Active learning approaches did not consistently outperform traditional passive learning with regards to key metrics. These approaches did however show efficient training characteristics. Training samples are selected in a more informed fashion which provides benefits during training. Passive learning remains competitive and simpler to implement for most scenarios. SASLA proved somewhat of a middle ground with moderate overhead required. Ensemble uncertainty incurs significant computational costs which struggle to outweigh training benefits.

A single hidden-layer architecture may not accurately reflect more complex deep learning approaches and a fixed selection parameter of 0.9 may not be optimal for all problems. Future work could include considering adaptive selection thresholds and extension into deep neural network architectures.

Active learning is not a universal solution and the overheads should be carefully considered before electing an active learning strategy over traditional passive learning.

References

[1] H. Hino, "Active learning: Problem settings and recent developments," *ArXiv*, vol. abs/2012.04225, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:227745938

[2] A. Engelbrecht, "Sensitivity analysis for selective learning by feedforward neural networks," *Fundam. Inform.*, vol. 45, pp. 295–328, 08 2001.

[3] M. Sharma and M. Bilgic, "Evidence-based uncertainty sampling for active learning," *Data Mining and Knowledge Discovery*, vol. 31, no. 1, pp. 164–202, 2017. [Online]. Available: https://doi.org/10.1007/s10618-016-0460-3

[4] M. Mahdavi and R. Jin, "Passive learning with target risk," *Journal of Machine Learning Research*, vol. 30, 02 2013.

[5] J. Zhu, H. Wang, T. Yao, and B. Tsou, "Active learning with sampling by uncertainty and density for word sense disambiguation and text classification." 01 2008, pp. 1137–1144.

[6] T. He, S. Zhang, J. Xin, P. Zhao, J. Wu, X. Xian, C. Li, and Z. Cui, "An active learning approach with uncertainty, representativeness, and diversity," *The Scientific World Journal*, vol. 2014, p. 827586, 2014. [Online]. Available: https://doi.org/10.1155/2014/827586

[7] Y. Yang and M. Loog, “Active learning using uncertainty information,” 2017. [Online]. Available: https://arxiv.org/abs/1702.08540

[8] T. Yin, G. Panapitiya, E. D. Coda, and E. G. Saldanha, “Evaluating uncertainty-based active learning for accelerating the generalization of molecular property prediction,” *Journal of Cheminformatics*, vol. 15, no. 1, p. 105, 2023. [Online]. Available: https://doi.org/10.1186/s13321-023-00753-5

[9] D. Nicolay, “Ml441 assignments,” https://github.com/Voltzz9/ml441_assignments, 2025.

[10] R. A. Fisher, “Iris,” 1936. [Online]. Available: https://archive.ics.uci.edu/dataset/53/iris

[11] S. Aeberhard and M. Forina, “Wine,” 1991. [Online]. Available: https://archive.ics.uci.edu/dataset/109/wine

[12] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms,” arXiv preprint arXiv:1708.07747, 2017, available at https://github.com/zalandoresearch/fashion-mnist.

[13] R. K. Pace and R. Barry, “Sparse spatial autoregressions,” 1990, dataset obtained from Kaggle. [Online]. Available: https://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html

[14] A. Tsanas and A. Xifara, “Energy Efficiency,” UCI Machine Learning Repository, 2012, DOI: https://doi.org/10.24432/C51307.

[15] M. Friedman, “The use of ranks to avoid the assumption of normality implicit in the analysis of variance,” *Journal of the American Statistical Association*, vol. 32, no. 200, pp. 675–701, 1937.

[16] C. Bonferroni, “Teoria statistica delle classi e calcolo delle probabilità,” *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commericiali di Firenze*, vol. 8, pp. 3–62, 1936.

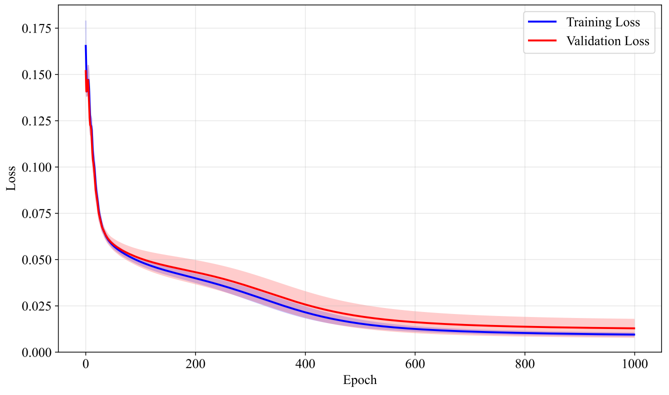
[17] J. Cohen, *Statistical power analysis for the behavioral sciences*, 2nd ed. Lawrence Erlbaum Associates, 1988.

[18] P. B. Nemenyi, “Distribution-free multiple comparisons,” Ph.D. dissertation, Princeton University, 1963.

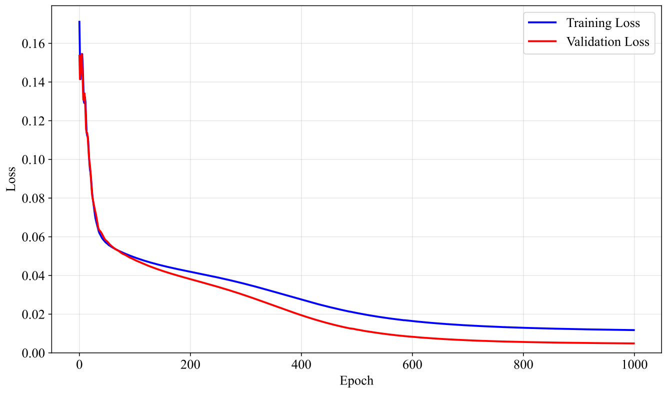
[19] B. Efron and R. J. Tibshirani, “An introduction to the bootstrap,” *CRC press*, 1994.

Appendix A
Additional Results

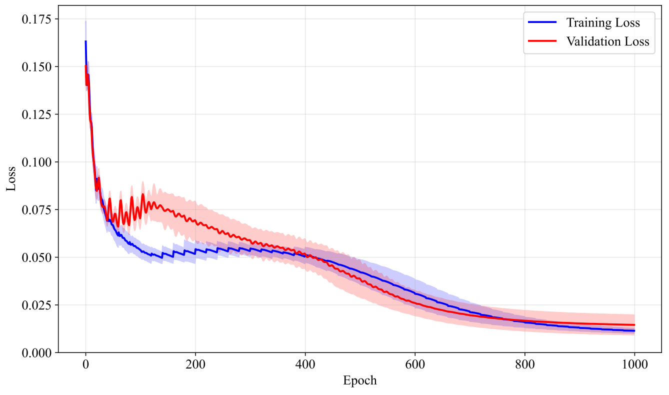
Fig. 2: Losses for training of frameworks with the iris dataset.



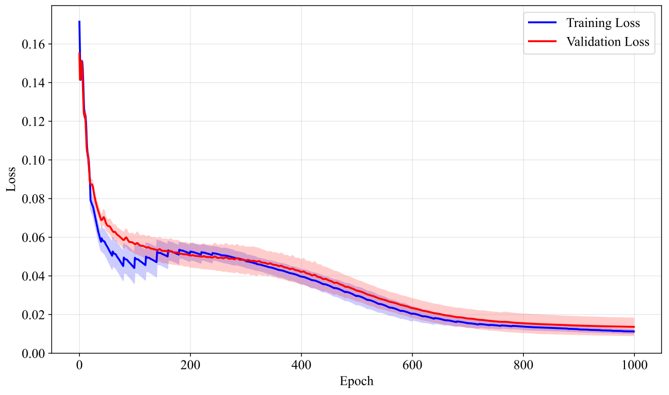
(a) Passive learning losses.



(b) SASLA losses.



(c) Entropy uncertainty sampling losses.



(d) Ensemble uncertainty sampling losses.

TABLE V: Comparison of classification accuracies $\pm$ standard deviation

Dataset	**Passive**	**SASLA**	**Entropy Uncertainty**	**Ensemble Uncertainty**
Iris Train	0.9819 ± 0.0066	0.9803 ± 0.0073	0.9692 ± 0.0105	0.9780 ± 0.0084
Iris Test	0.9819 ± 0.0292	0.9653 ± 0.0298	0.9613 ± 0.0322	0.9660 ± 0.0302
Wine Train	0.9834 ± 0.0050	0.9834 ± 0.0047	0.9799 ± 0.0059	0.9693 ± 0.0114
Wine Test	0.9750 ± 0.0237	0.9728 ± 0.0238	0.9722 ± 0.0255	0.9506 ± 0.0339
Fashion-MNIST Train	0.9030 ± 0.0023	0.9033 ± 0.0027	0.8688 ± 0.0034	0.8513 ± 0.0029
Fashion-MNIST Test	0.8414 ± 0.0109	0.8409 ± 0.0113	0.8309 ± 0.0110	0.8246 ± 0.0100

TABLE VI: Comparison of regression MSE $\pm$ standard deviation

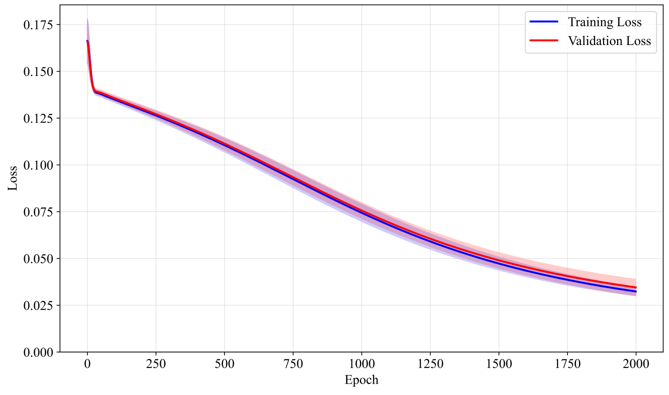
Dataset	**Passive**	**SASLA**	**Entropy Uncertainty**	**Ensemble Uncertainty**
Synthetic Train	0.0301 ± 0.0550	0.1148 ± 0.0770	0.1402 ± 0.3900	0.0752 ± 0.0877
Synthetic Test	0.0321 ± 0.0609	0.1146 ± 0.0758	0.1389 ± 0.3713	0.0746 ± 0.0872
Housing Train	0.0835 ± 0.0202	0.0921 ± 0.0080	0.1022 ± 0.0194	0.1023 ± 0.0179
Housing Test	0.0842 ± 0.0217	0.0926 ± 0.0088	0.1027 ± 0.0198	0.1027 ± 0.0188
Energy Train	0.0442 ± 0.0267	0.0455 ± 0.0320	0.2522 ± 0.2416	0.1232 ± 0.0411
Energy Test	0.0349 ± 0.0258	0.0459 ± 0.0315	0.2519 ± 0.2431	0.1210 ± 0.0409

TABLE VII: Computation time (in seconds) $\pm$ standard deviation for different methods

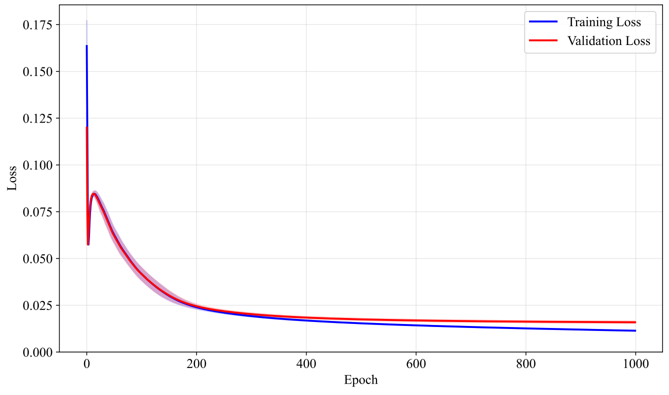
Dataset	**Passive**	**SASLA**	**Entropy Uncertainty**	**Ensemble Uncertainty**
Iris	4.664 ± 0.185	6.731 ± 0.337	5.512 ± 0.158	1.630 ± 0.047
Wine	9.754 ± 0.745	12.991 ± 0.454	9.860 ± 0.234	3.565 ± 0.263
Fashion	8.757 ± 0.291	21.378 ± 0.815	12.397 ± 0.768	23.825 ± 0.705
Synthetic Functions	0.968 ± 0.046	1.094 ± 0.064	0.950 ± 0.041	2.251 ± 0.101
Housing	0.596 ± 0.051	0.491 ± 0.107	2.431 ± 0.079	12.642 ± 0.577
Energy	1.376 ± 0.149	1.444 ± 0.084	1.438 ± 0.054	4.501 ± 0.134

Fig. 3: Losses for training of frameworks with the wine dataset.

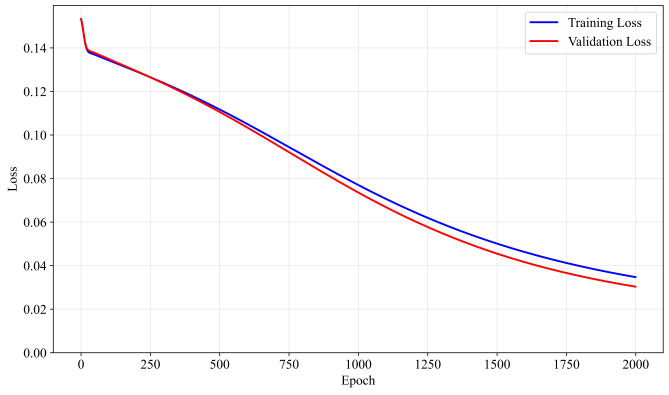
Fig. 4: Losses for training of frameworks with the fashion-MNIST dataset.



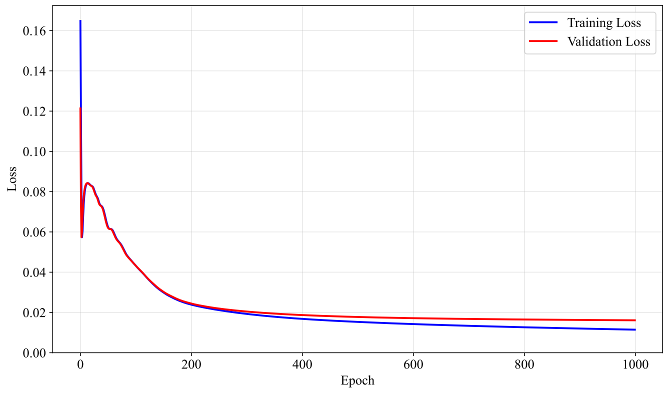
(a) Passive learning losses.



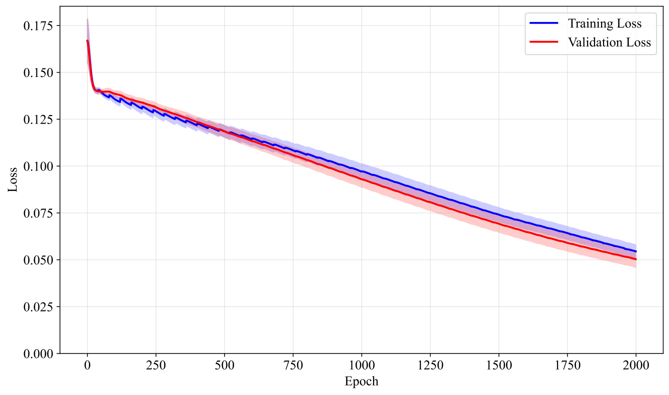
(a) Passive learning losses.



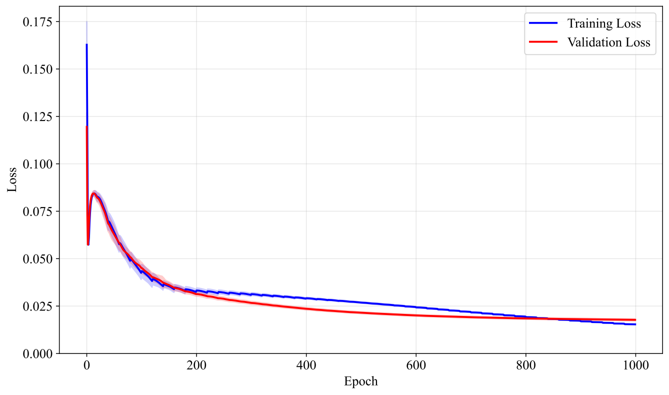
(b) SASLA losses.



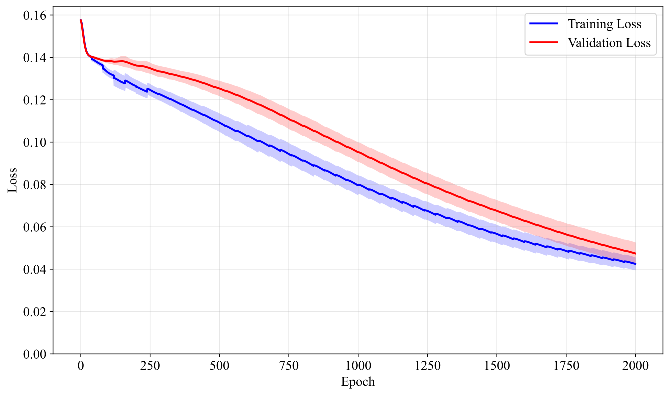
(b) SASLA losses.



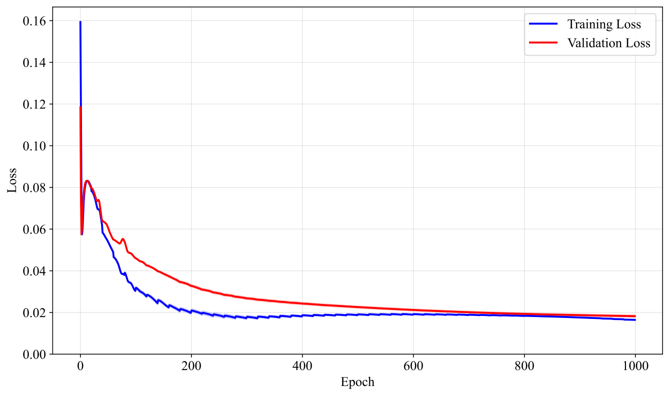
(c) Entropy uncertainty sampling losses.



(c) Entropy uncertainty sampling losses.

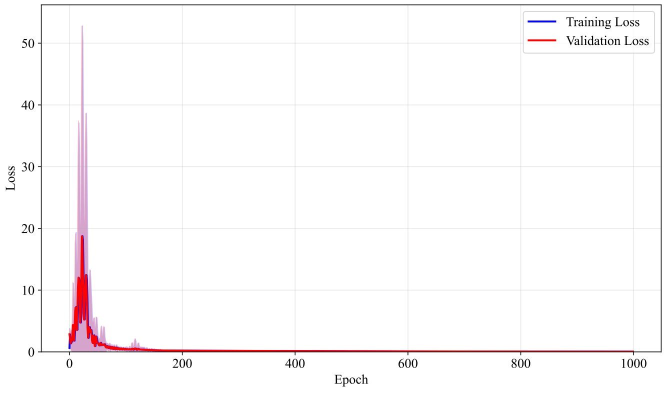


(d) Ensemble uncertainty sampling losses.

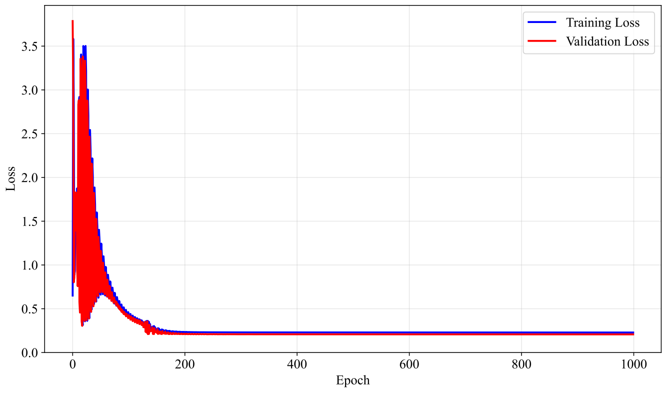


(d) Ensemble uncertainty sampling losses.

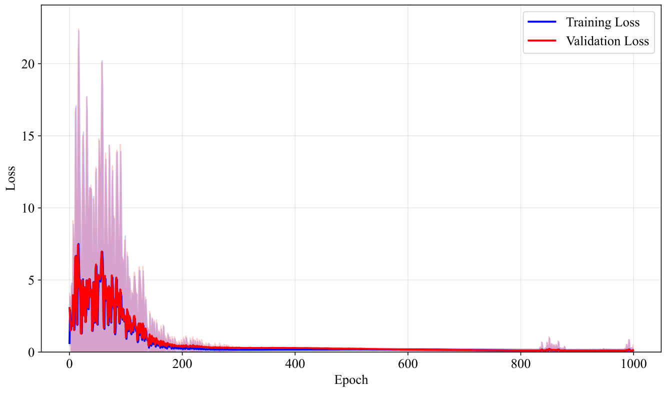
Fig. 5: Losses for training of frameworks with the synthetic function dataset.



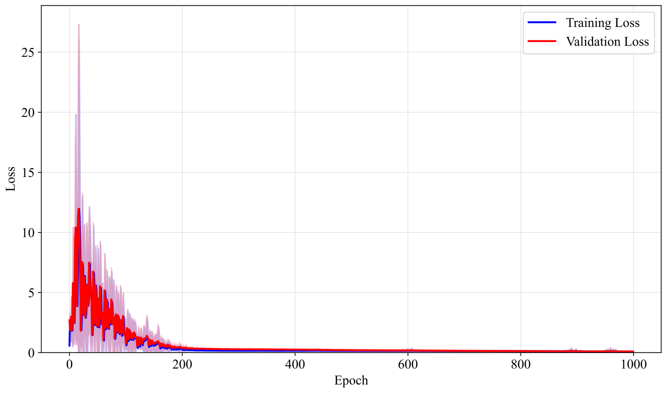
(a) Passive learning losses.



(b) SASLA losses.

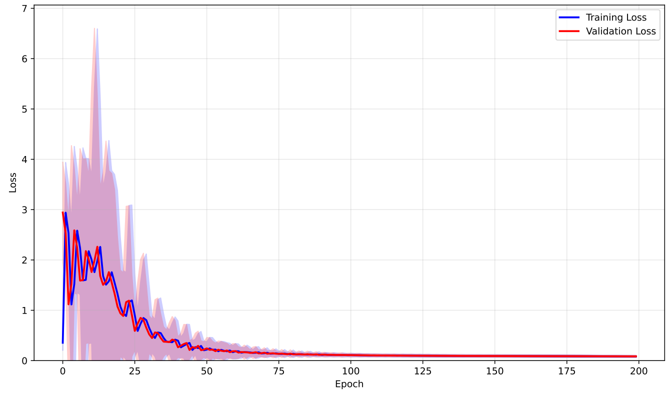


(c) Entropy uncertainty sampling losses.

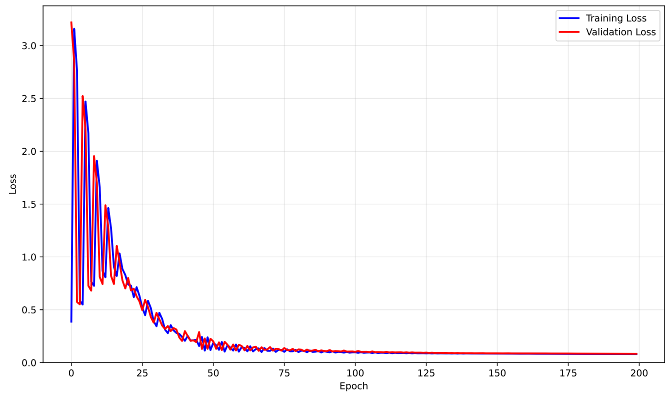


(d) Ensemble uncertainty sampling losses.

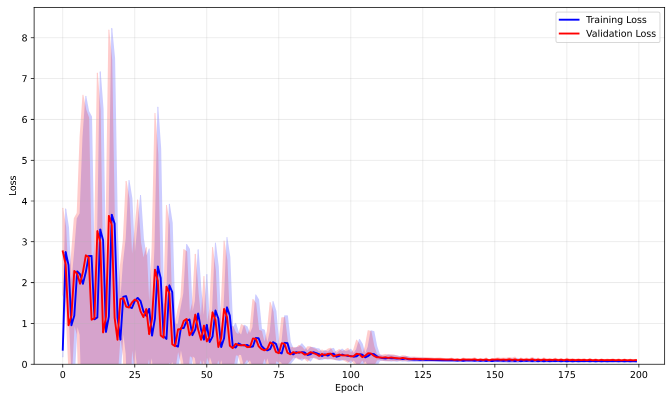
Fig. 6: Losses for training of frameworks with the California housing dataset.



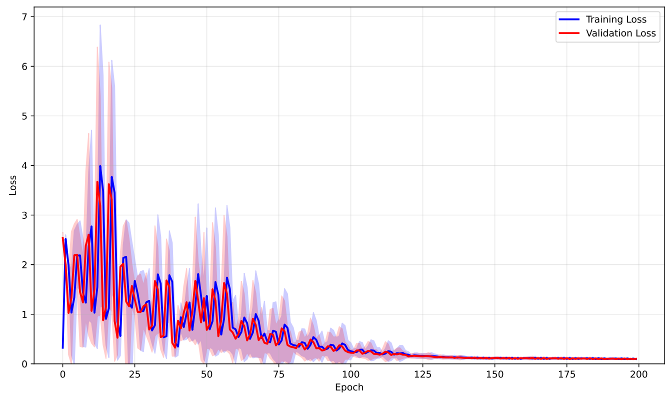
(a) Passive learning losses.



(b) SASLA losses.

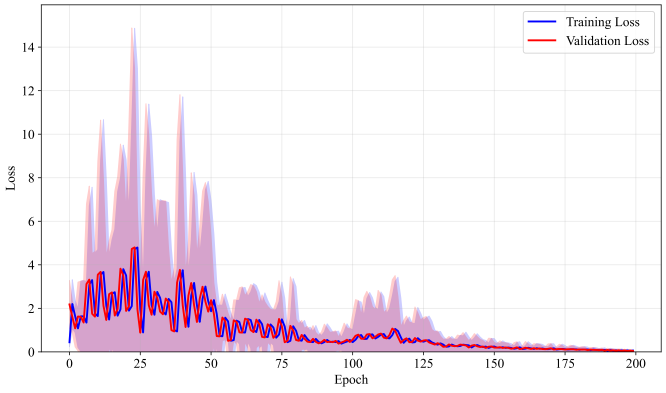


(c) Entropy uncertainty sampling losses.

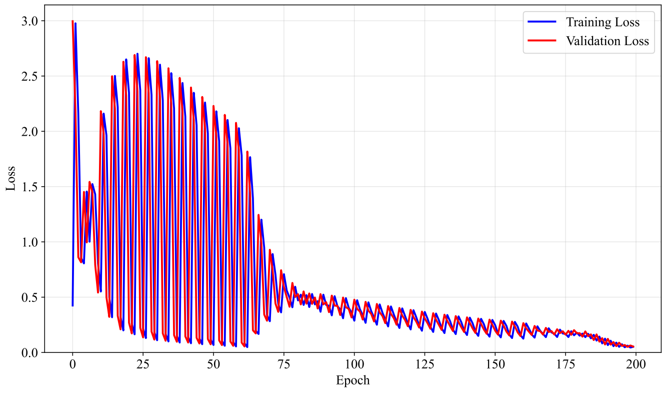


(d) Ensemble uncertainty sampling losses.

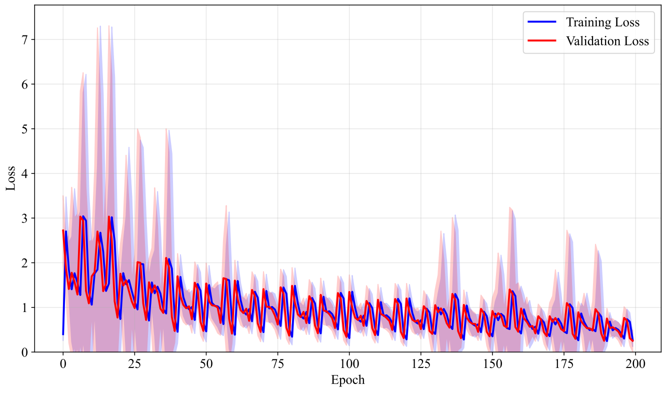
Fig. 7: Losses for training of frameworks with the energy efficiency dataset.



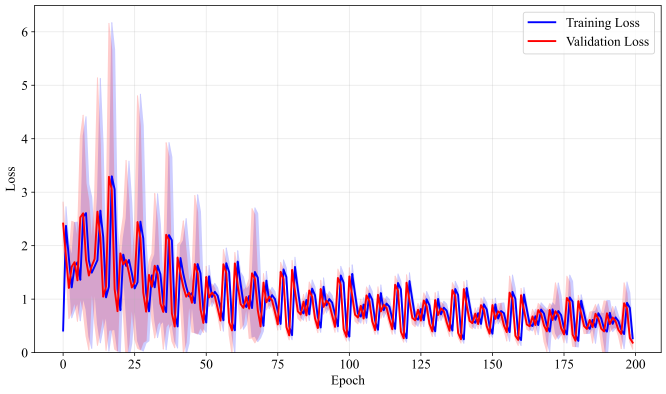
(a) Passive learning losses.



(b) SASLA losses.



(c) Entropy uncertainty sampling losses.



(d) Ensemble uncertainty sampling losses.

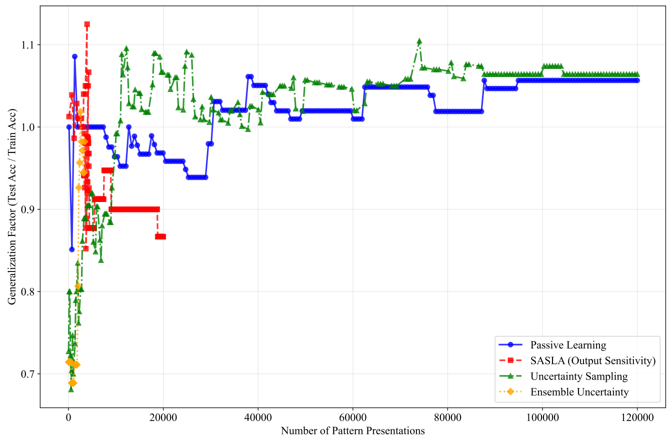


Fig. 8: Generalisation factor versus number of pattern presentations for the iris dataset.

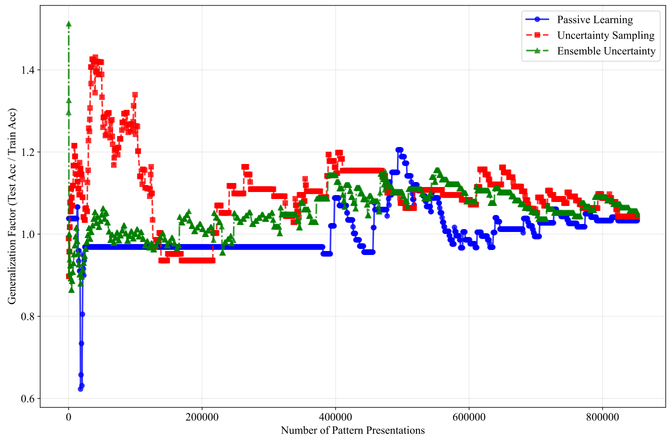


Fig. 9: Generalisation factor versus number of pattern presentations for the wine dataset.

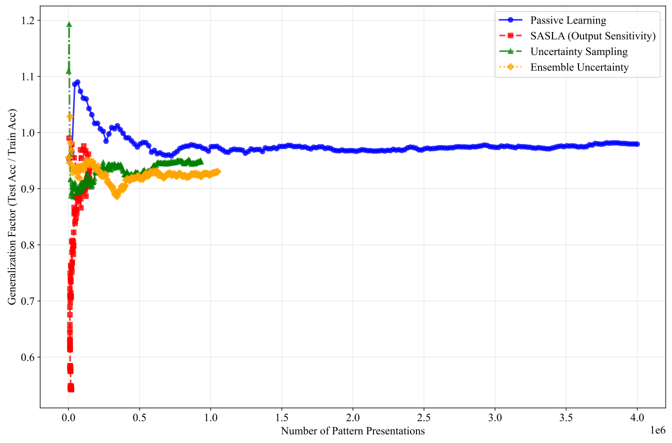


Fig. 10: Generalisation factor versus number of pattern presentations for the fashion-MNIST dataset.

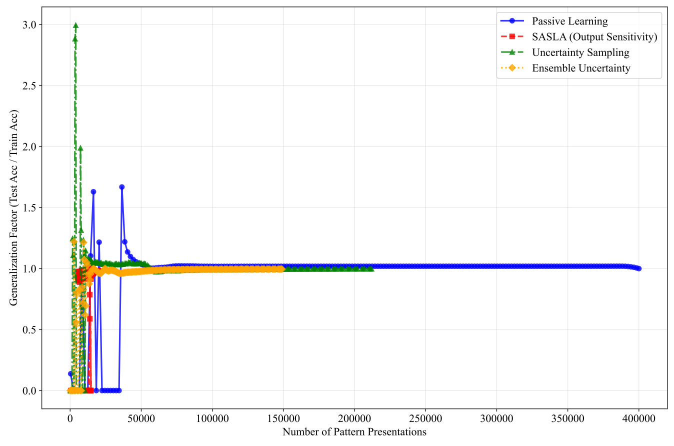


Fig. 11: Generalisation factor versus number of pattern presentations for the synthetic function dataset.

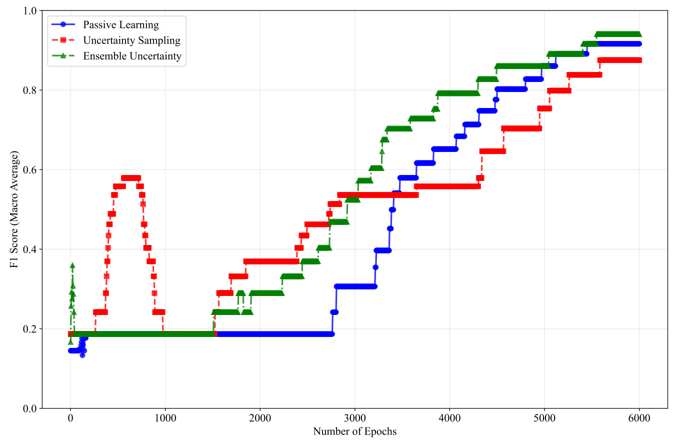


Fig. 14: F1 score versus number of epochs for the wine dataset.

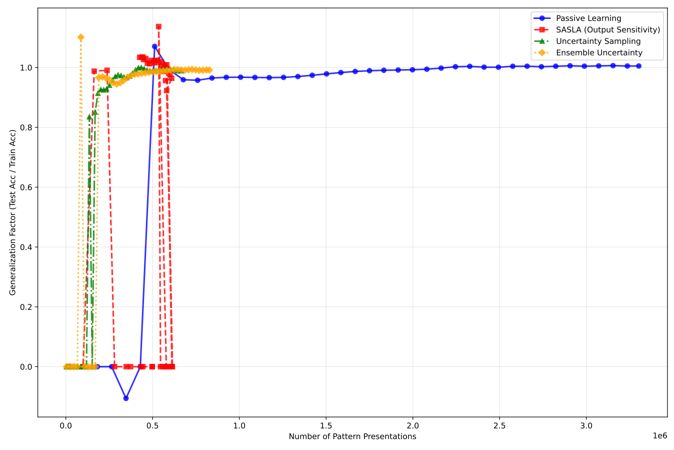


Fig. 12: Generalisation factor versus number of pattern presentations for the California housing dataset.

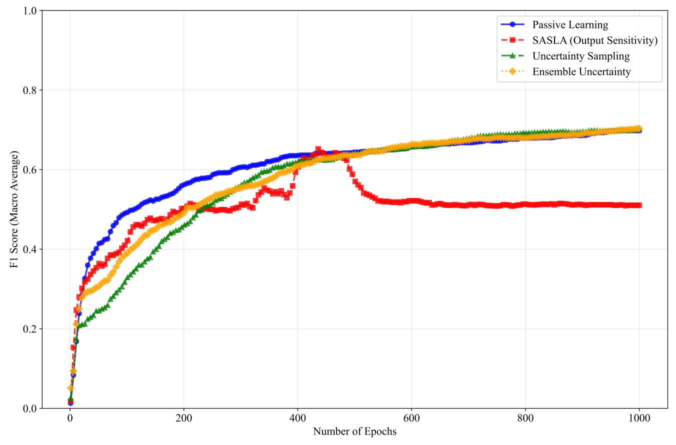


Fig. 15: F1 score versus number of epochs for the fashion-mnist dataset.

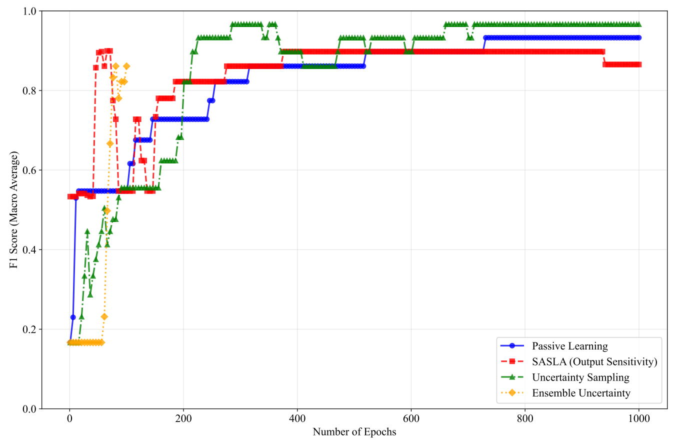


Fig. 13: F1 score versus number of epochs for the iris dataset.

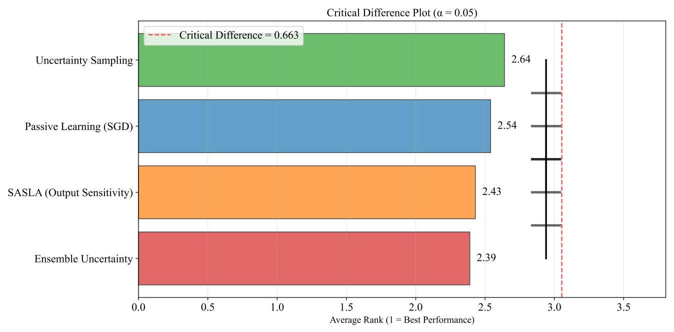


Fig. 16: Critical difference plot for the iris dataset.

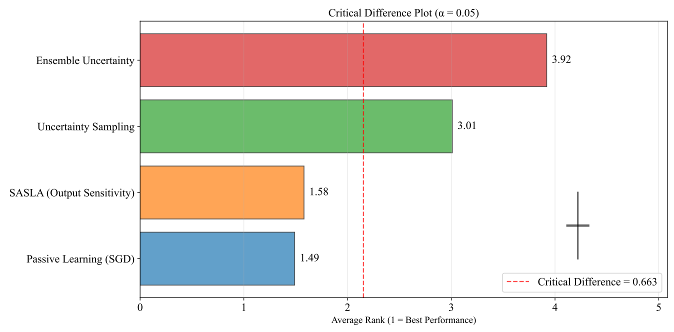


Fig. 17: Critical difference plot for the fashion-mnist dataset.

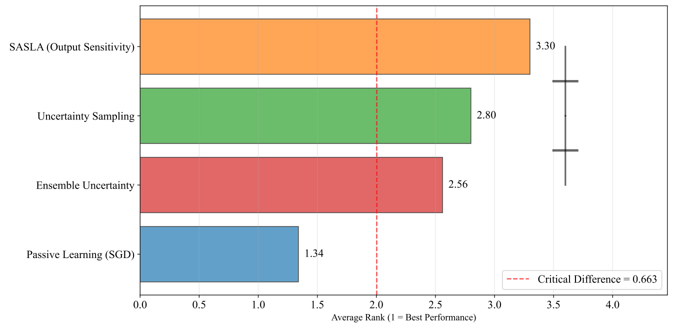


Fig. 18: Critical difference plot for the synthetic function dataset.

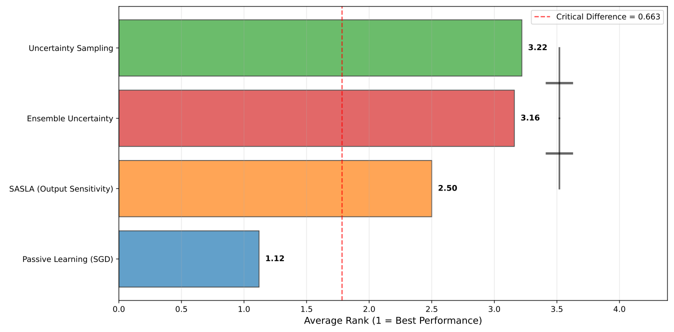


Fig. 19: Critical difference plot for the California Housing dataset.

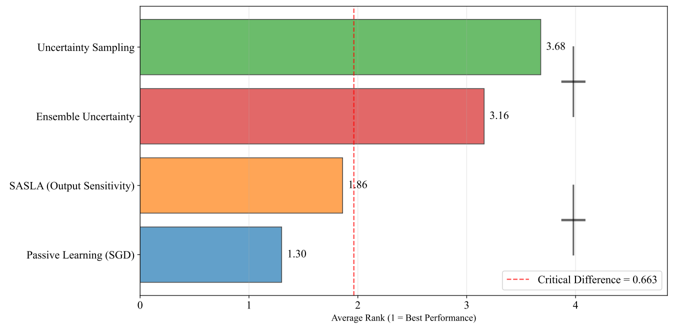


Fig. 20: Critical difference plot for the energy efficiency dataset.