Hybrid Learning in Neural Networks for Almond Classification

Arno Jooste
Department of Computer Science
The University of Pretoria
u21457451@tuks.co.za

***Abstract*—This research paper proposes a novel method for classifying almond species utilising a hybrid learning approach within a neural network architecture. Almonds are nutrient-dense nuts, and precise classification is crucial for agricultural quality control and pricing. Traditional classification approaches, such as visual inspection and genetic analysis, are labour-intensive and prone to errors. To address these issues, we hope to increase classification accuracy by combining two popular training algorithms: Resilient Backpropagation (RPROP) and Stochastic Gradient Descent (SGD). We optimised important parameters using a structured experimental setup that incorporates data preprocessing techniques including outlier removal and K-Nearest Neighbour imputation for missing values. The findings show that our hybrid learning technique employing RPROP and SGD did not increase performance measures, emphasising the need to explore more combinations of various training algorithms with similar weight update rules.**

***Index Terms*—Neural Networks, Classification, Hybrid Learning**

I. Introduction

Almonds are valuable nuts rich in protein, vitamins, and minerals, with health benefits like regulating blood sugar [1], [2]. Accurate classification of almond varieties plays a crucial role in the agricultural industry, impacting quality control and pricing. Traditionally, visual inspection or genetic analysis are used for classification, but these methods can be time-consuming, prone to errors, and expensive [3].

This study investigates a novel strategy for efficient and accurate almond classification using hybrid learning within a neural network. Neural networks have shown substantial effectiveness in image classification tasks. By combining different neural network training algorithms, we can potentially achieve improved classification accuracy.

This report details the development and evaluation of hybrid learning within a neural network for almond classification. We will first provide background information on almond classification and neural networks. Next, we describe the experimental setup, including the chosen hybrid architecture, training process, and evaluation metrics. We then present the research results and discuss the model's performance. Finally, we conclude by summarising the key findings.

II. Background

This study investigates the use of gradient-based training algorithms for optimising the parameters of a basic neural network model. The parameters include the number of epochs, activation function, loss function, learning rate, and number of hidden layers. Optimising these parameters is critical for increasing the model's performance. This study relies on two key algorithms: Resilient Propagation (RPROP) [4] and Stochastic Gradient Descent (SGD) [5]. Both play important roles in neural network training.

A. RPROP

Resilient Propagation (RPROP) is an effective optimisation technique that modifies the size of weight updates based on local gradient information. It optimises training by focussing on the gradient's direction rather than its size, resulting in faster convergence. If the gradient changes direction, RPROP reduces the update size to minimise overshooting. If the gradient stays the same, the step size increases to speed up convergence [4]. This strategy is especially effective for dealing with vanishing or expanding gradients.

B. SGD

Stochastic Gradient Descent (SGD) is an extremely popular optimisation strategy in machine learning, particularly for large-scale tasks. Unlike traditional gradient descent, which computes gradients across the entire dataset, SGD updates the model's weights progressively using a random selection of the input. This stochastic technique enables faster training and improved scalability on large datasets. SGD, despite introducing noise into the optimisation process, has proven to be effective in many practical applications due to its simplicity and efficiency [6].

C. Other Optimisation Techniques

While this study focusses on RPROP and SGD, other optimisation methods have demonstrated promise for increasing neural network performance. Mehmood et al. used complex methods including Adam-based optimisers, simplified RMSProp by deleting a hyperparameter, and maintained a constant learning rate to improve training efficiency [7].

Furthermore, Ilani et al. used deep convolutional neural networks (DCNNs) to perform image classification tasks such as almond kernel detection. Their findings proved the advantages of DCNNs for image-based challenges, due to characteristics such as convolutional layers for feature extraction and pooling layers for dimensionality reduction. The authors used image augmentation, transfer learning, and dropout strategies to improve model accuracy and resilience [8].

III. Experimental Setup

This sections outlines how the problem was approached and solved. We first look at the dataset used and then at other key factors such as the parameter values that were used.

A. Dataset

The dataset for this research was provided and is publicly available on Kaggle. It essentially consists of features taken from images using various image processing techniques, with additional features resulting from these initial extractions. The authors used a multi-step methodology to extract information from images. To capture more colour information, the images were first transformed from RGB to HSV colour space. A mask was subsequently developed to extract brown pixels, which are believed to be almond kernels. The Canny algorithm was used to determine the edges of these kernels, and contours were found within the mask to define the almond shapes. Finally, almond segments were isolated from the source images using this mask [9]. A detailed list of the features and their descriptions can be found in Table I.

Feature Name	**Feature Description**
Length (major axis)	Length of the almond in the image (pixels)
Width (minor axis)	Width of the almond in the image (pixels)
Thickness (depth)	Thickness of the almond in the image (pixels)
Area	Area of the almond region detected in the image
Perimeter	Total length of the almond boundary (pixels)
Roundness	Roundness $\left(4 \times \text{Area}/(\pi \times \text{Length}^2)\right)$
Solidity	Ratio of Area to Convex Hull Area
Compactness	Shape compactness $\left(\text{Perimeter}^2/(4 \times \pi \times \text{Area})\right)$
Aspect Ratio	Length-to-Width ratio (Length/Width)
Eccentricity	Elongation $\left(\sqrt{1-(\text{Width}/\text{Length})^2}\right)$
Extent	Area to Bounding Box Area ratio
Convex Hull (Convex Area)	Area of the smallest convex set
Type	Almond type (Mamra, Sanora, Regular)

TABLE I: Description of Features in The Dataset [9]

B. Data Preparation

Initially the dataset contained a lot of noise such as missing values, outliers, and even some irrelevant columns.

1) Outliers: The distribution of outliers is shown in Figure 1, containing a boxplot for each numerical feature. The outliers were handled based on the feature's distribution. If the feature had a normal/near-normal distribution, Z-scores were used to impute the outliers with that feature's average. However, if the feature had a skewed distribution, the outliers for that feature were identified using the interquartile range (IQR), and replaced using the median value of that feature.

2) Missing Values: The dataset contained a lot of missing values for some of its features. At first it was considered to impute missing values using mean/median imputation for the features such as *'Width (minor axis)', 'Length (major axis)', and 'Thickness (depth)'*. However, this resulted in the distributions of the updated features to introduce bias where the mean or median values formed the majority. To prevent this, the imputation strategy changed from using either mean or median imputation, to using a K-Nearest-Neighbour approach. This approach imputed the missing values using the 5 nearest neighbouring data points of a missing value in each feature, respectively. This approached proved to be a better choice as the distributions showed unbiased values. Note that this was only with respect to the three earlier mentioned features. Some of the other features were dependent on others (e.g. *'Aspect Ratio'* being dependent on *'Width (minor axis)'* and *'Length (major axis)*). Any missing values in such a feature were replaced using their formulas as shown in Table I. These choices led to more accurate imputations of the missing values than mean or median imputation.

3) Standardising the data: After the outliers were handled, there remained a few outliers in some of the features. Those outliers might have introduced some bias to the classification model. Therefore, it was important that we standardised the data to mitigate any potential biases. Robust scaling was chosen over other scaling techniques such as *Min-Max* scaling and the *StandardScaler*. The *StandardScaler* requires normally distributed data, and the *Min-Max* scaler assumes that physical limitations prevent features from surpassing specific values. Another limitation for these two scaling techniques is that outliers negatively impacts them. These limitations are overcome by the *RobustScaler* through the approach of scaling the features with the first and third quartiles rather than the minimum, mean, and maximum values. This allows the *RobustScaler* to be insensitive to outliers [10]. Therefore, we chose this scaling technique because some of our features still contained outliers.

4) Encoding the class labels: Two encoding techniques were considered in this study: One-hot encoding, and label-encoding. Label-encoding is most often used for ordinal categories, where each category is encoded as an integer value. In contrast to this, one-hot encoding is most often used for nominal data, where one creates a new feature and link each category to a binary variable (0 or 1). 0 indicates absence and 1 indicates presence of a category [11]. In this study we opted to use label-encoding. This may have been problematic in other datasets, since our categories are nominal. However, we did not want to increase the dataset's dimensions, and label-encoding save a bit more memory which may be ideal for the training process.

5) Splitting the data: The dataset was split into three subsets to ensure an unbiased evaluation of the model's performance. Initially, the data was separated into two sets: training (60%) and temporary (40%). The Training Set was used to train the model, while the Temporary Set was divided into two parts: Validation (20%) and Test (20%). The Validation Set was used to fine-tune hyperparameters and detect overfitting during training. The Test Set, maintained completely separate, gave a final, unbiased assessment of the model's performance. This strategy ensures that the model generalises well while avoiding overfitting to the training data.

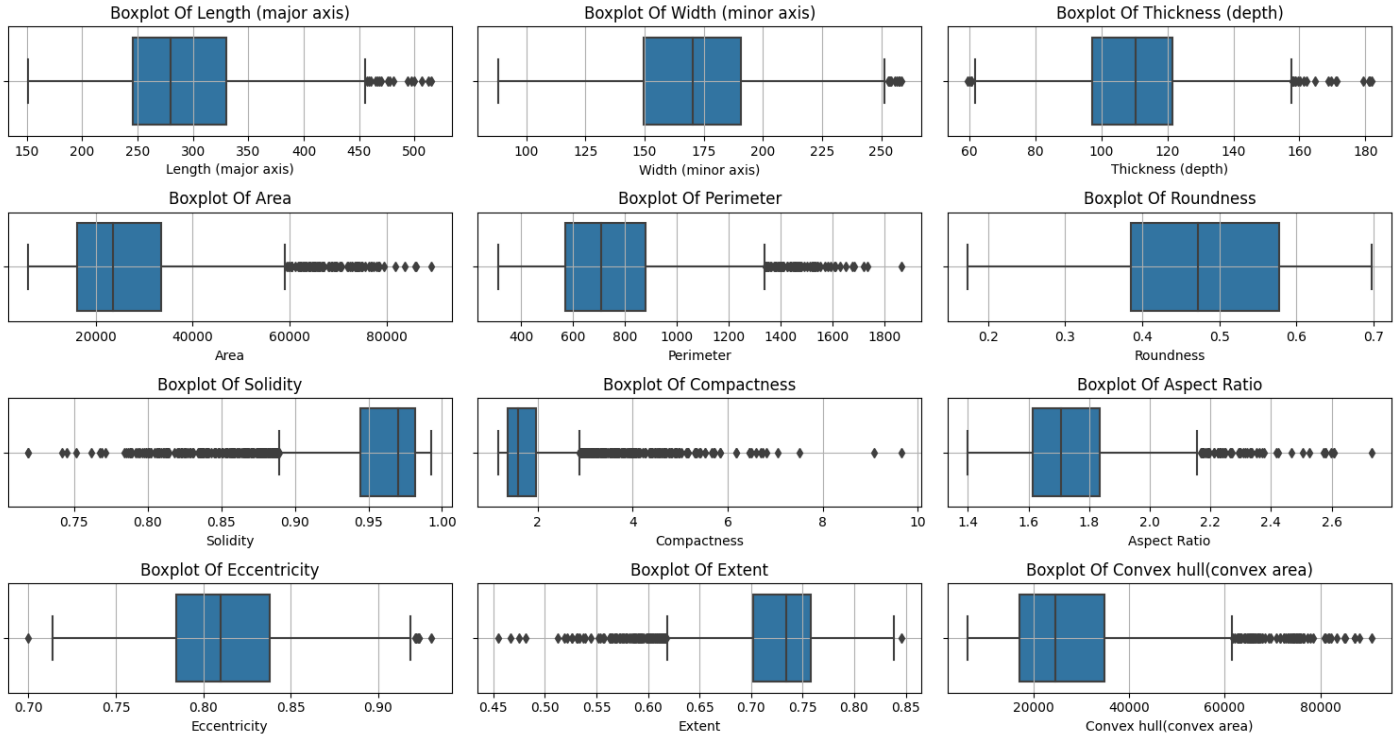

Fig. 1: Boxplot Showcasing Outliers in Numerical Features

C. Hyperparameters

This research involved several hyperparameters whereby at least two of them needed to be optimised. Table II provides a description for each hyperparameter, their values, and also highlights the two that were chosen to be optimised. A gridsearch was required to compare the combinations of the two chosen hyperparameters such that the combination producing the highest average accuracy would serve as the possible optimised values for those two hyperparameters.

1) Theoretical Insights: To justify the choice of each hyperparameter's value(s), we can consider theoretical insights. According to Jeff Heaton, who wrote the book *Introduction to Neural Networks with Java*, the input size should be equal to the amount of features from the dataset. For classification problems, as done for this study, the author also mentions that the output size should be equal to the amount of class labels [12]. We used powers of 2 for the number of neurons in the hidden layer to take advantage of matrix multiplication's efficiency. SIMD operations in CPUs are optimised for batch sizes that are powers of 2, which can boost computing speed by allowing for parallel processing. We chose to use Cross-Entropy as the loss function in the neural network model, simply because it performs better than the Mean-Square-Error loss function for classification tasks [13]. The ReLU activation function was chosen as a possible choice, because it is regarded as the standard activation function to use for many classification problems [14]. We used the Sigmoid activation function as our other option for the activation function, since both these activation function are the most commonly used activation functions, and each have their own benefits for classification problems [15]. The ranges for the number of epochs was chosen as displayed in Table II because a large number of epochs may lead to overfitting. Therefore, we also introduced a stopping criteria to help prevent overfitting by terminating the active run's fold when its accuracy stagnates. Finally, we used RPROP or SGD as the two choices for optimising the model's parameters. RPROP was a requirement, but we chose SGD because it is considered ene of the most useful strategies for online optimisation in machine learning [16].

2) Emperical Evidence: To justify the choice for the two hyperparameters that were optimised (i.e. Number of hidden layers and Learning rate), we provided a discussion on the comparison of the values used for these two parameters. The values used to experiment with these two hyperparameters are also provided in Table II.

D. Model Architecture

The neural network has a feedforward structure, with a given input layer, numerous hidden layers of various sizes, and a single output layer. For nonlinearity, the design uses ReLU or Sigmoid activation functions, and weights are initialised using Xavier to improve training stability. The hybrid learning approach uses the two optimisers (RPROP and SGD) to improve neural network training. During each epoch, the model computes the loss using a forward pass, then uses a backward pass to calculate gradients. The hybrid learning approach accumulates gradients for each parameter across two separate optimisers without requiring immediate updates. After processing mini-batches, the accumulated gradients are averaged. This enables shared effect on weight updates. The averaged gradient is then used to update the model's weights with each optimiser. By combining the

strengths of the two optimisers, this hybrid approach aims to improve convergence and overall model performance.

Hyperparameter	Description	Value(s)
Seed	Seed value used for reproducability	99
Input size	Number of neurons for the input layer	12
Hidden sizes	Number of neurons for each hidden layer	[256, 128, 64, 32] (if 4 hidden layers)
Output size	Number of neurons for the output layer	3
Hidden layers	Number of hidden layers	4, 3, or 2
Learning rates	Learning rates used	0.001, 0.01, or 0.02
Loss function	Measures prediction error	Cross-Entropy
Activation function	Introduces non-linearity	ReLU or Sigmoid
Number of epochs	Number of training iterations	$n \in \mathbb{Z}, 100 \leq n \leq 500$
Optimiser Algorithm	Optimises model parameters	RPROP or SGD

TABLE II: Hyperparameters' Description and Value(s)
The highlighted rows indicate the parameters chosen to be optimised

IV. Research Results

In this study we made use of two evaluation metrics, namely accuracy and loss, where the best performance was measured using accuracy. The accuracy is the percentage of correct predictions among all, determined using formula (1).

$$\text{Accuracy} = \frac{\text{number of correct predictions}}{\text{total number of predictions}} \tag{1}$$

The loss can be determined by using the Cross-Entropy(CE) loss function from PyTorch which uses formula (2).

$$\text{CE} = -\sum_{i}^{C} t_i \log f(s)_i \tag{2}$$

where:

- t represents the target class' probability
- $f(s)$ refers to the predicted output's probability
- C represents the amount of target classes

There were a total number of **10 independent runs** performed for each combination of the two hyperparameters.

A. Training Results

The combined statistical analysis (Tables III to VIII) and visual (Figures 2 to 6) analysis compares the three optimisers - SGD, RPROP, and a hybrid approach - in terms of neural network training performance over a range of hidden layer and learning rate configurations. The statistical analysis shows that both RPROP and SGD perform consistently across the explored hidden layer and learning rate combinations, with no statistically significant differences detected in most cases. Notably, the Hybrid optimiser was slightly sensitive to hidden layer configurations, with one significant difference observed when comparing four hidden layers to two, implying that the Hybrid approach may react more easily to structural changes in the model.

Heatmaps and line plots provide additional insights into these optimisers' performance dynamics. The heatmaps, namely Figure 3, show that the SGD optimiser performed the best overall. Increasing the number of hidden layers and learning rate enhanced SGD performance, resulting in a high average performance of 86% with 4 hidden layers and a learning rate of 0.02. This hyperparameter combination also resulted in the greatest performance for RPROP, obtaining 67% average performance, as seen in Figure 2. The Hybrid approach performed poorly with these parameters, as shown in Figure 4. These results are supported by line plots that examine the distribution of accuracy and loss scores across epochs for each optimiser. The plots indicate that the SGD optimiser outperformed the other two approaches. This is supported by the other two approaches having shorter line plots, indicating faster convergence or the stopping criteria was reached in fewer epochs due to fitness stagnation.

In Figure 5, SGD outperformed RPROP and the Hybrid method, with training accuracy scores of roughly 99%. The Hybrid approach did not perform well which may be related to its limited use of SGD and RPROP. Additionally, the poor performance may be due to SGD and RPROP using different weight update rules. This implies that more research is needed to see whether various combinations of optimisers could improve the Hybrid approach's performance.

Overall, the data reveals that the SGD optimiser is the most resilient and efficient of the three, particularly when tuning hyperparameters like hidden layers and learning rates. While RPROP and the Hybrid technique produce consistent results, their performance is significantly lower, especially the Hybrid approach.

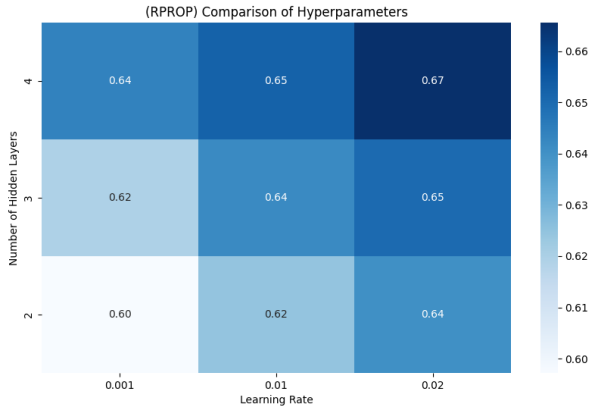

Fig. 2: RPROP: Heatmap Showing model Training Performance Between Hyperparameter Variations

Further comparison of results for the average accuracies and losses' mean and standard deviations are reported in Tables IX to XII. Each row in these tables represents the average score for that specific combination of hyperparameters across five runs. These tables strenghthe the earlier discussion where we concluded that SGD outperformed the other two optimisers.

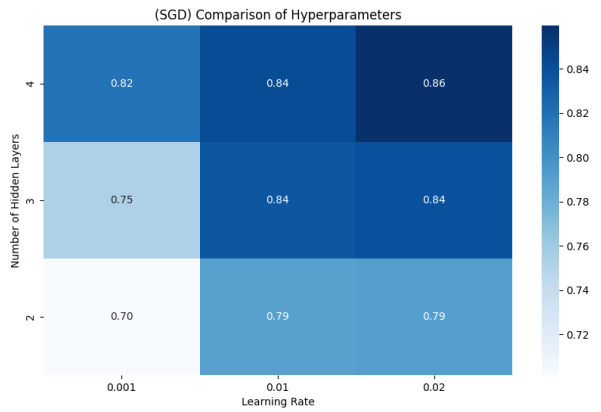

Fig. 3: SGD: Heatmap Showing model Training Performance Between Hyperparameter Variations

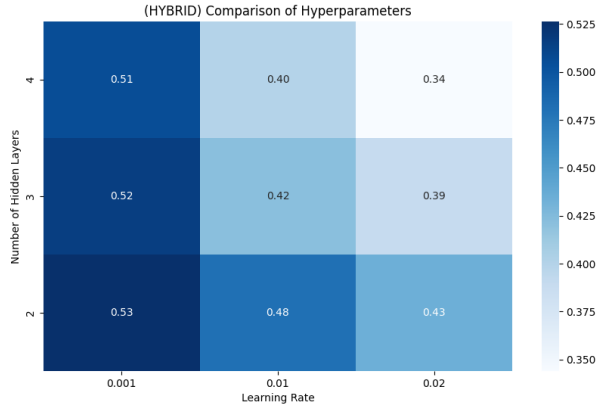

Fig. 4: Hybrid: Heatmap Showing model Training Performance Between Hyperparameter Variations

# HL 1	# HL 2	CO	T-stat	P-val	Sig. Diff.
4	3	FALSE	-1.511	0.205	FALSE
4	2	FALSE	-2.377	0.076	FALSE
3	2	FALSE	-1.067	0.346	FALSE

TABLE III: RPROP: Statistical Analysis of Neural Network Performance Across Hidden Layer Variations

Abbreviations: *# HL = Number of Hidden Layers, CO = Confidence Overlap, T-stat = T-statistic, P-val = P-Value, Sig. Diff. = Significant Difference.*

LR 1	LR 2	CO	T-stat	P-val	Sig. Diff.
0.001	0.01	FALSE	1.222	0.289	FALSE
0.001	0.02	FALSE	2.056	0.109	FALSE
0.01	0.02	TRUE	1.136	0.320	FALSE

TABLE IV: RPROP: Statistical Analysis of Neural Network Performance Across Learning Rate Variations

Abbreviations: *LR = Learning Rate, CO = Confidence Overlap, T-stat = T-statistic, P-val = P-Value, Sig. Diff. = Significant Difference.*

# HL 1	# HL 2	CO	T-stat	P-val	Sig. Diff.
4	3	TRUE	-1.006	0.371	FALSE
4	2	FALSE	-2.484	0.068	FALSE
3	2	FALSE	-1.184	0.302	FALSE

TABLE V: SGD: Statistical Analysis of Neural Network Performance Across Hidden Layer Variations

Abbreviations: *# HL = Number of Hidden Layers, CO = Confidence Overlap, T-stat = T-statistic, P-val = P-Value, Sig. Diff. = Significant Difference.*

LR 1	LR 2	CO	T-stat	P-val	Sig. Diff.
0.001	0.01	FALSE	1.731	0.159	FALSE
0.001	0.02	FALSE	1.844	0.139	FALSE
0.01	0.02	TRUE	0.281	0.792	FALSE

TABLE VI: SGD: Statistical Analysis of Neural Network Performance Across Learning Rate Variations

Abbreviations: *LR = Learning Rate, CO = Confidence Overlap, T-stat = T-statistic, P-val = P-Value, Sig. Diff. = Significant Difference.*

# HL 1	# HL 2	CO	T-stat	P-val	Sig. Diff.
4	3	FALSE	0.425	0.693	FALSE
4	2	FALSE	1.179	0.0304	TRUE
3	2	FALSE	0.828	0.454	FALSE

TABLE VII: Hybrid: Statistical Analysis of Neural Network Performance Across Hidden Layer Variations

Abbreviations: *# HL = Number of Hidden Layers, CO = Confidence Overlap, T-stat = T-statistic, P-val = P-Value, Sig. Diff. = Significant Difference.*

LR 1	LR 2	CO	T-stat	P-val	Sig. Diff.
0.001	0.01	FALSE	-0.593	0.585	FALSE
0.001	0.02	FALSE	-1.219	0.2898	FALSE
0.01	0.02	FALSE	-0.652	0.5497	FALSE

TABLE VIII: Hybrid: Statistical Analysis of Neural Network Performance Across Learning Rate Variations

Abbreviations: *LR = Learning Rate, CO = Confidence Overlap, T-stat = T-statistic, P-val = P-Value, Sig. Diff. = Significant Difference.*

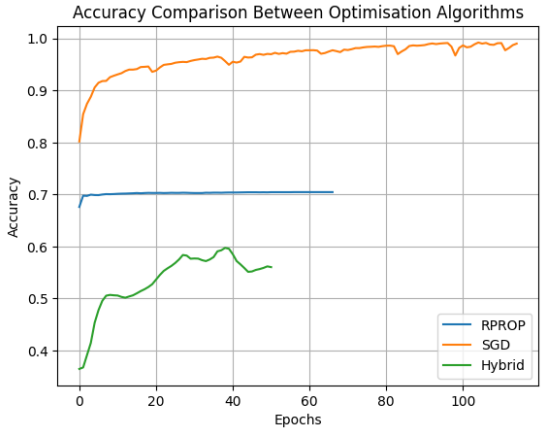

Fig. 5: Accuracy Comparison Between Optimisation Algorithms

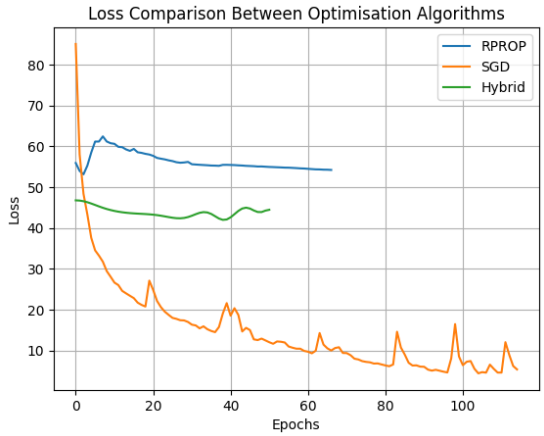

Fig. 6: Loss Comparison Between Optimisation Algorithms

# HL	LR	RPROP	SGD	Hybrid
4	0.001	0.6510	0.8183	0.5058
4	0.01	0.6603	0.8429	0.3998
4	0.02	0.6753	0.8596	0.3440
3	0.001	0.6351	0.7528	0.5159
3	0.01	0.6528	0.8358	0.4213
3	0.02	0.6616	0.8395	0.3897
2	0.001	0.6074	0.7016	0.5264
2	0.01	0.6318	0.7896	0.4810
2	0.02	0.6504	0.7914	0.4344

TABLE IX: Average Accuracy: Performance Comparison of RPROP, SGD, and Hybrid Optimisers
Abbreviations: *# HL = Number of Hidden Layers, LR = Learning Rate*

B. Validation Results

Following the training process, a grid search was used to determine the best hyperparameter combinations. The RPROP and SGD optimisers produced the same optimum configurations, whereas the Hybrid approach produced a different set. This distinction was also underlined in the previous heatmaps.
RPROP achieved its highest accuracy score of 60.61% with **4 hidden layers** and **a learning rate of 0.02**, while SGD scored 73.44%. The respective loss scores were **0.8964** for RPROP and **1.2314** for SGD. Thus, despite having a greater loss score, SGD achieved improved classification accuracy. In contrast, the Hybrid technique, which used **2 hidden layers** and **a learning rate of 0.001**, had an accuracy score of only **52.05%** and a loss score of **1.0253**.

# HL	LR	RPROP	SGD	Hybrid
4	0.001	0.0160	0.0767	0.0387
4	0.01	0.0216	0.0947	0.0460
4	0.02	0.0260	0.0957	0.0339
3	0.001	0.0205	0.0410	0.0359
3	0.01	0.0220	0.0806	0.0523
3	0.02	0.0215	0.0827	0.0452
2	0.001	0.0155	0.0218	0.0334
2	0.01	0.0183	0.0617	0.0472
2	0.02	0.0224	0.0636	0.0660

TABLE X: Average Accuracy Standard Deviation: Performance Comparison of RPROP, SGD, and Hybrid Optimisers
Abbreviations: *# HL = Number of Hidden Layers, LR = Learning Rate*

# HL	LR	RPROP	SGD	Hybrid
4	0.001	1.6495	0.4764	1.0546
4	0.01	0.8371	0.5656	1.4242
4	0.02	0.7553	0.5131	2.2320
3	0.001	0.7941	0.5718	1.0559
3	0.01	0.7712	0.5827	1.5535
3	0.02	0.9189	0.5760	2.0310
2	0.001	0.9027	0.6680	1.2508
2	0.01	1.0118	0.6656	2.3207
2	0.02	0.7711	0.7896	2.1703

TABLE XI: Average Loss: Performance Comparison of RPROP, SGD, and Hybrid Optimisers
Abbreviations: *# HL = Number of Hidden Layers, LR = Learning Rate*

# HL	LR	RPROP	SGD	Hybrid
4	0.001	1.0414	0.1853	0.1543
4	0.01	0.1632	0.4046	0.2075
4	0.02	0.0570	0.4205	0.8536
3	0.001	0.0423	0.0803	0.1253
3	0.01	0.0415	0.3664	0.2959
3	0.02	0.2626	0.3818	0.6259
2	0.001	0.1034	0.0435	0.2447
2	0.01	0.2549	0.2316	0.6634
2	0.02	0.0371	0.3318	0.6689

TABLE XII: Average Loss Standard Deviation: Performance Comparison of RPROP, SGD, and Hybrid Optimisers
Abbreviations: *# HL = Number of Hidden Layers, LR = Learning Rate*

C. Testing Results

Finally, the optimal configurations for each optimiser were used to evaluate the models on the testing set. RPROP had an accuracy of **59.18%** and a loss score of **0.8909**. This performance was again worse to that of SGD, which had an accuracy score of **73.797%** and a loss score of **1.4669**. Even with the optimum configuration, the Hybrid technique could not surpass the other two optimisers, with accuracy and loss scores of **54.72%** and **0.9659** respectively.

Despite the fact that some algorithms showed a good or poor performance, we observe, from the training results, that all models performed relatively consistent across multiple independent runs for all hyperparameter configurations. Overall, the validation and testing results were much lower than the training results. This shows that the model may be overfitting, even with the optimal configurations. To address this issue, further exploration of additional combinations and ranges of hyperparameter values is necessary.

V. Conclusion

This research offers significant insights into the classification of almond varieties utilising neural network training techniques. While both RPROP and SGD produced competitive results, the hybrid technique did not improve accuracy for the specific problem at hand. This emphasises the significance of properly choosing training algorithms and hyperparameters depending on the dataset's specific properties. Thus, we conclude that we did not achieve our goal of improving accuracy. This may be due to the Hybrid approach utilising two algorithms with different weight update rules. The SGD utilises a global learning rate across all weights, while the RProp method updates learning rates according on the gradient's sign [17]–[19]. Additionally, averaging distinct weight update directions may also have led to slower convergence and poor performance. Future research could explore new algorithm combinations, specifically algorithms with similar weight update rules, or feature engineering techniques to improve classification results.

References

[1] K. Sze-Tao and S. Sathe, "Functional properties and in vitro digestibility of almond (prunus dulcis l.) protein isolate," *Food Chemistry*, vol. 69, no. 2, pp. 153–160, 2000, ISSN: 0308-8146. DOI: https://doi.org/10.1016/S0308-8146(99)00244-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0308814699002447.

[2] D. Barreca, S. M. Nabavi, A. Sureda, *et al.*, "Almonds (prunus dulcis mill. d. a. webb): A source of nutrients and health-promoting compounds," *Nutrients*, vol. 12, no. 3, 2020, ISSN: 2072-6643. DOI: 10.3390/nu12030672. [Online]. Available: https://www.mdpi.com/2072-6643/12/3/672.

[3] M. Yurdakul, İ. Atabaş, and Ş. Taşdemir, "Almond (prunus dulcis) varieties classification with genetic designed lightweight cnn architecture," *European Food Research and Technology*, vol. 250, no. 10, pp. 2625–2638, Oct. 2024, ISSN: 1438-2385. DOI: 10.1007/s00217-024-04562-4. [Online]. Available: https://doi.org/10.1007/s00217-024-04562-4.

[4] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The rprop algorithm," in *IEEE International Conference on Neural Networks*, 1993, 586–591 vol.1. DOI: 10.1109/ICNN.1993.298623.

[5] S.-i. Amari, "Backpropagation and stochastic gradient descent method," *Neurocomputing*, vol. 5, no. 4, pp. 185–196, 1993, ISSN: 0925-2312. DOI: https://doi.org/10.1016/0925-2312(93)90006-O. [Online]. Available: https://www.sciencedirect.com/science/article/pii/092523129390006O.

[6] D. Newton, F. Yousefian, and R. Pasupathy, "Stochastic gradient descent: Recent trends," in *Recent Advances in Optimization and Modeling of Contemporary Problems*, ch. 9, pp. 193–220. DOI: 10.1287/educ.2018.0191. eprint: https://pubsonline.informs.org/doi/pdf/10.1287/educ.2018.0191. [Online]. Available: https://pubsonline.informs.org/doi/abs/10.1287/educ.2018.0191.

[7] F. Mehmood, S. Ahmad, and T. K. Whangbo, "An efficient optimization technique for training deep neural networks," *Mathematics*, vol. 11, no. 6, 2023, ISSN: 2227-7390. DOI: 10.3390/math11061360. [Online]. Available: https://www.mdpi.com/2227-7390/11/6/1360.

[8] M. A. Ilani, S. M. Tehran, A. Kavei, and A. Radmehr, *Automatic image annotation (aia) of almondnet-20 method for almond detection by improved cnn-based model*, 2024. arXiv: 2408.11253 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2408.11253.

[9] S. Moradi, *Almond types classification*, 2024. [Online]. Available: https://www.kaggle.com/datasets/sohaibmoradi/almond-types-classification.

[10] O. J. Osogba, "Machine learning for subsurface data analysis: Applications in outlier detection, signal synthesis and core & completion data analysis," Ph.D. dissertation, 2020.

[11] M. K. Dahouda and I. Joe, "A deep-learned embedding technique for categorical features encoding," *IEEE Access*, vol. 9, pp. 114 381–114 391, 2021. DOI: 10.1109/ACCESS.2021.3104357.

[12] J. Heaton, *Introduction to Neural Networks with Java*. Heaton Research, 2008, ISBN: 9781604390087. [Online]. Available: https://books.google.co.za/books?id=Swlcw7M4uD8C.

[13] L. Hui and M. Belkin, *Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks*, 2021. arXiv: 2006.07322 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2006.07322.

[14] K. Eckle and J. Schmidt-Hieber, "A comparison of deep networks with relu activation function and linear spline-type methods," *Neural Networks*, vol. 110, pp. 232–242, 2019, ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2018.11.005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608018303277.

[15] S. Mastromichalakis, "Sigmorelu: An improvement activation function by combining sigmoid and relu¡strong¿ ¡/strong¿," *Preprints*, Jun. 2021. DOI: 10.20944/preprints202106.0252.v1. [Online]. Available: https://doi.org/10.20944/preprints202106.0252.v1.

[16] S. Eslamian and F. Eslamian, *Handbook of HydroInformatics: Volume I: Classic Soft-Computing Techniques*. Elsevier Science, 2022, ISBN: 9780128219706. [Online]. Available: https://books.google.co.za/books?id=m4meEAAAQBAJ.

[17] P. Zhou, J. Feng, C. Ma, C. Xiong, S. C. H. Hoi, and W. E, "Towards theoretically understanding why sgd generalizes better than adam in deep learning," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., Curran Associates, Inc., pp. 21 285–21 296. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/f3f27a324736617f20abbf2ffd806f6d-Paper.pdf%7D,%20volume%20=%20%7B33%7D,%20year%20=%20%7B2020%7D.

[18] W. Saputra, A. P. Windarto, and A. Wanto, "Analysis of the resilient method in training and accuracy in the backpropagation method," *International Journal of Informatics and Computer Science (IJICS)*, vol. 5, no. 1, pp. 11–15, 2021. DOI: 10.30865/ijics.v5i1.2922.

[19] L. M. Saini, "Peak load forecasting using bayesian regularization, resilient and adaptive backpropagation learning based artificial neural networks," *Electric Power Systems Research*, vol. 78, no. 7, pp. 1302–1310, 2008, ISSN: 0378-7796. DOI: https://doi.org/10.1016/j.epsr.2007.11.003%7D. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S037877960700258%7D.