COS711 Assignment 2

1st Robert Officer
U20431122
University of Pretoria

***Abstract*—This report investigates the optimization of neural networks (NNs) for classifying almonds into three categories: Mamra, Sanora, and Regular, utilizing Resilient Backpropagation (RPROP) and Adaptive Moment Estimation (Adam) as gradient-based training algorithms. An EDA of the given dataset was performed and found that preprocessing of the dataset was required to account for missing values, outliers, and categorical encoding. Hyperparameter optimization focused on the number of hidden layers and learning rate, make use of a grid search approach in order to find these optimal values. Experimental results revealed that RPROP generally outperformed Adam across various average and standard deviations metrics, including accuracy, precision, recall, and F1 score. However, Adam was shown to have better metrics for accuracy and F-score between the two optimisers best runs. Furthermore, a hybrid learning approach combining RPROP and Adam resulted in generally sub-optimal performance, but produced results just below those produced by Adam and RPROP in its best run, indicating potential instability due to conflicting gradient updates. This study aimed to emphasize the importance of careful optimizer selection and optimisation of hyper-parameters.**

***Index Terms*—Neural network, classification, RPROP, Adam, Hyper-parameter optimisation, Hybrid-learning**

I. Introduction

Neural networks (NN) have demonstrated significant potential in solving complex classification problems across various domains. In a comparative study study done in [1] on the performances of NN compared to Logistic Regression models, NN outperformed LR in 62% of the collected papers. Although Issitt et al. state that the differences between the two models were often insignificant. However, these results still show that NN models are able to perform at similar levels of accuracy as LR models. This report focuses on optimizing a neural network (NN) to classify almonds into three distinct categories: Mamra, Sanora, and Regular. The classification task is performed of the data set found in [2]. The main goals of this assignments include: preprocessing this data, optimizing the NN's hyperparameters, and comparing various gradient-based training algorithms to enhance the model's classification performance.

The classification problem being solved in this report required a selection of supervised models and learning techniques in order to classify the given dataset of almonds into the following categories: Mamra, Sanora, or Regular. This report focuses on the model optimisation techniques known as Resilient Backpropagation (RProp) and Adaptive Moment Estimation (Adam).

This report will follow the following structure. Section II will describe the algorithms and techniques used in the implementation of this report. Section III will run through the experimental setup of the implementation. This section includes the setup used to develop, run and test the implementation. Section **??** will show the results collected from running the implemented classifiers. This section also includes a critical analysis subsection whereby the various chosen gradient-based training algorithms are compared against each other.

II. Background

NNs can be briefly explained as a information processing technique that is modelled after the biological structure of the brain. Standard NN make use of interconnected neurons with weightings and activation functions in order to process incoming data and produce results that can be interpreted by humans. In the case of this report, the NN takes in a vast set of numerical features and the goal is to classify the given almond instances into the three previously mentioned categories. Back-propagation neural networks (BPNN) make use of gradient calculations based off of the results produced by the model in-order to adjust the weights of the neuron connections in the model to improve these results. This report makes use of optimisation techniques to further improve the results of the model as well as improve the efficiency in which these improvements take place.

A. Data preprocessing

The dataset [2] collected data from the Kaggle dataset [3] using image processing techniques. Using these techniques, Moradi was able to extract various numerical features from the images relating to almonds. A series of preprocessing steps were applied to the dataset before being feed to the chosen optimisers and hybrid implementation. The steps taken were decided based on a brief Exploratory Data Analysis (EDA). This EDA revealed that not only were there outlying and missing data values, but also that an unnamed column was present in the data. This unnamed column was simply the index of each data instances in the dataset.

The EDA resulted in the following processes being run against the dataset. Missing values were handled by replacing them with the mean of their respective columns, ensuring the dataset remained intact without creating outliers within the dataset. Outliers, which can skew model performance, were addressed by removing any instances falling in the upper 95th

and lower 5th percentiles. The data was also normalized using the standard deviation and mean technique shown in **??**. This step is crucial in ensuring that the different scales of numerical data do not skew the models performances. Furthermore, the categorical "Type" column was one-hot encoded to allow the network to treat the class labels as distinct numerical categories. Normally converting categorical to numerical is done to make use of this categorical data, however since the "Type" column in this dataset shows us the true classification, this conversion was done simply to make it easier to see if the classification done by the model is correct.

$$Z_{i,p}^{MV} = \frac{Z_{i,p} - \bar{Z}_i}{\sigma_{Z_i}} \tag{1}$$

Figures 1 and 2 show the box plots of the dataset before and after preprocessing has occurred. The empty plots were formed automatically but have no relevance to the dataset.

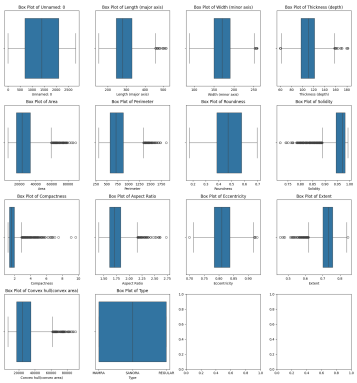

Fig. 1. Box plots of the dataset attributes before preprocessing

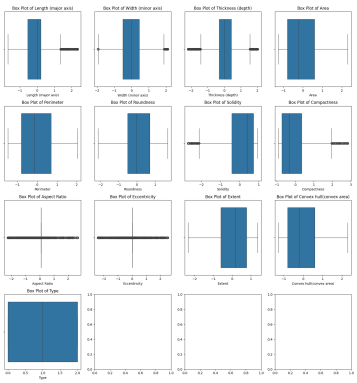

Fig. 2. Box plots of the dataset attributes after preprocessing

B. Model optimisers

As stated in the introduction, the SGD and RPROP optimisation techniques were chosen to apply to the standard NN model implemented.

The RPROP optimiser was introduced in [4]. This optimiser's aim is to improve the shortfalls that are attached to standard gradient-descent variants of NN. These improvements include improved convergence speed by looking at previous updates as well as the current direction of the gradient. Riedmiller and Braun state that this approach ensures that blurred adaptivity, whereby the magnitude of the gradient affects the weight step taken by the adjustment protocols directly, does not occur.

Stochastic Gradient Descent is an optimiser for NN that aims to reduce the cost that comes with very large datasets and difficulty in adding new data by updating the gradient information of the model after each instance in the dataset or after each mini-batch. [5]. The Adam optimiser is an extension to this introduced by Diederik Kingma and Jimmy Ba [6]. Adam makes use of only the first order gradient information that adjusts parameters based on the first and second moments (the mean of the gradients and the variance of the gradients respectively [8]) [7]. Due to this, Adam is able to perform more efficiently than its predecessors and is less costly in terms of memory.

III. Experimental Setup

This report focuses on the *Accuracy*, *Precision* and *Recall*, *F-Score* of the trained models. Because this system is being used to solve a ternary classification problem, macro-averaging was used to calculate a single value across all three classes. This was decided appropriate since the EDA discovered that there was an equal distribution of class instances in the dataset. The equations for these measurements are listed in 2, 3, 4, 5 respectively. Accuracy measures the proportion of correctly classified data instances out of the total instances. Precision measures the proportion of class specific positive classifications that are true. Recall/Sensitivity measures the proportion of correct positives classified by the model. The F-Score is a balanced value that uses both Precision and Recall in its calculation. It's primary purpose is to balance the values of these metrics and give a more precise score.

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Samples}} \tag{2}$$

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{3}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{4}$$

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5}$$

Both the development and experimentation runs were completed using Python (v3.11.3) on an Asus Vivobook Pro 14X with a AMD Ryzen 7 5800H CPU 3.2 GHz (Boosted: 4.4 GHz) with 16GB 3200 MHz DDR4. Although this report has a focus on hyper-parameters, there are still some set values that were chosen. These are as follows: *Epochs = 5000; Test set proportion size = 0.15*.

A. Testing and training data

The pre-processed dataset was split into two subsets: testing data and training data. As stated in the previous section, this split had a ratio of *85:15* between the training and testing sets. The training set was feed into the model and optimisers in order to produced the optimal weightings and hyper-parameters for the model. At each epch, the training set was shuffled. This was to avoid the model from learning any non-existent or erroneous data patterns that may be in the training dataset. The testing data set was then used to see if the trained model does not over- or under-fit the dataset.

B. NN model

This implementation made use of a standard NN model. The structure of this model is as follows:

- Input layer neurons: This layer has 12 neurons, correlating the number of attributes in the dataset after preprocessing
- Hidden layer neurons: Each hidden layer has 12 neurons
- Activation function: The relu activation function was chosen for all layers in the standard NN model.
- Initial weightings: The initial weightings of the standard NN model are automatically generated on creation of a new NN model. These values make use of a seed value for the random generator to allow for reproducable results.
- Loss function: This model made use of the Cross Entropy Loss function. This was chosen as the loss function due to its ability to handle multi-class classification [9].

C. Hyper-parameter optimisation implementation

Hyper-parameter optimisation was performed on the number of hidden layers and the learning rate of the models (represented as *num_hidden_layers* and *learning_rate* respectively in the implementation). A grid search was performed over two arrays containing multiple values for both hyper-parameters. The arrays are as follows:

- Number of hidden layers: 2, 4, 6, 8, 10, 12, 14
- Learning rate: An array of 10 floating point numbers, evenly spaced between 0.01 and 0.15

The hyper-parameter optimisation was run 10 times to ensure that particular seed values were nto responsible for high performing parameter values.

D. Hybrid learning implementation

The hybrid learning approach makes use of the RPROP and Adam optimisation patterns in its implementation. A single classifier is used between both optimisers and the gradient update rule makes use of the calculated mean between these two methods. This average gradient rule is then fed back into the optimiser where it is used in recalculating the weights for the classifier.

The implemented setup can be found here [1].

[1]Full URL address: https://github.com/Rob-0ff/COS711_assignments/tree/main/Assignment1

IV. Research Results

This section will present the results obtained by the implemented system. This section will first run through hyper-parameter optimisation in order to present the optimal values for the hyper-parameters that were chosen. These collected values were then used for the runs used in the comparative study between the RPROP and Adam models.

A. Hyper-parameter optimisation

During the grid-search performed by the hyper-parameter optimisation, both the loss and accuracy were collected and used to determine the optimal values. Figures 3 and 4 show the heat-maps of loss and accuracy produced by the HPO for the RPROP optimiser whilst 5 and 6 show these heat-maps produced for the Adam optimiser.

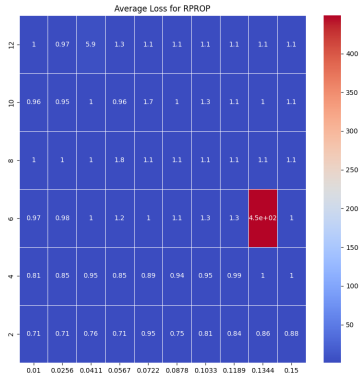

Fig. 3. Heat-map showing optimal values for the hyper-parameters learning rate (X-xis) and number of hidden layers (Y-axis) in terms of loss for RPROP.

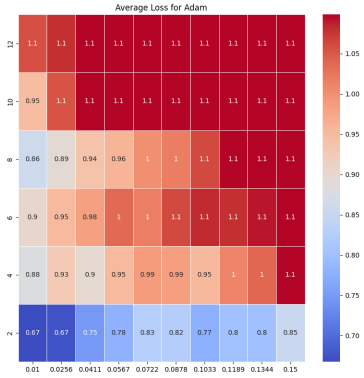

Fig. 4. Heat-map showing optimal values for the hyper-parameters learning rate (X-xis) and number of hidden layers (Y-axis) in terms of loss for Adam.

The heat-maps produced revealed that the most optimal values for the hyper-parameters number of hidden layers and learning rate in terms of loss and accuracy. The two values returned were then averaged to produce a single result that was then used for the rest of the runs. The resulting values for RPROP were *Number of hidden layers = 2, learning rate = 0.0.0723* and for Adam were *Number of hidden layers = 2, learning rate = 0.0178*.

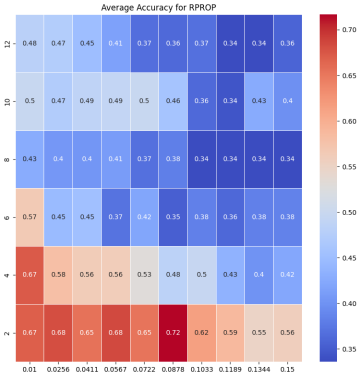

Fig. 5. Heat-map showing optimal values for the hyper-parameters learning rate (X-xis) and number of hidden layers (Y-axis) in terms of accuracy for RPROP.

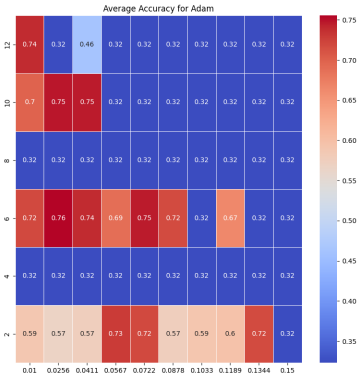

Fig. 6. Heat-map showing optimal values for the hyper-parameters learning rate (X-xis) and number of hidden layers (Y-axis) in terms of accuracy for Adam.

B. Optimiser results

The following section show the results of the RPROP and Adam optimiser collected over ten runs. This was to account for the stochastic nature of the optimisers in using seed values to initialise initial weights for the models. The results displayed are the averages collected over ten runs.

TABLE I
COMPARISON OF RPROP AND ADAM OVER 10 RUNS

Metric	RPROP	Adam
Average Accuracy	0.6216	0.6145
Average Precision	0.6235	0.6148
Average Recall	0.5560	0.5408
Average F1 Score	0.5693	0.5530
Standard Deviation of Accuracy	0.1324	0.1383
Standard Deviation of Precision	0.1335	0.1382
Standard Deviation of Recall	0.2161	0.2181
Standard Deviation of F1 Score	0.1939	0.1949
Best Run Seed	1727895892	1727895949
Accuracy (Best Run)	0.7672	0.7815
F1 Score (Best Run)	0.7686	0.7782

The results in I provide insight into the performances of the RPROP and Adam optimising algorithms. In all averaging and standard deviation metrics, RPROP outperformed Adam. The differences between the two optimisers are insignificant. However, Adam's best run produced the better results in relation to the two optimisers best runs. In terms of accuracy of these best runs, it can be seen that the Adam has a greater classification accuracy and the higher F1-score shows that Adam is better at maximising true positive cases whilst minimising false positives.

The large standard deviations for both Adam and RPROP show that improvements in consistency should be looked into.

The seed values are provided in the table to reproduce the results. Take note that the results are based off of the testing dataset formed. However, in general, Adam and PROP could not consistently perform well as seen by the averages in terms of accuracy.

C. Hybrid learning results

This section presents the results collected over ten runs for the hybrid learning implementation. Again this was to account for the stochastic nature of the optimisers. Table II lays out these results.

TABLE II
PERFORMANCE METRICS FROM 10 RUNS

Metric	Value
Average Accuracy	0.5264
Average Precision	0.5112
Average Recall	0.4348
Average F1 Score	0.4269
Standard Deviation of Accuracy	0.1776
Standard Deviation of Precision	0.1932
Standard Deviation of Recall	0.2920
Standard Deviation of F1 Score	0.2725
Best Run Seed for RPROP	1727896463
Best Run Accuracy	0.7387
Best Run F1 Score	0.7453

The average accuracy results generated from the hybrid approach being just greater than 50% shows that this approach has a poor handling on classifying the dataset into the three in general. The combination of gradient updates between RPROP and Adam could be the result of this. The updates from one optimiser maybe resulting in instability during the training of the model. This instability may come from the loss of advantages that come from dynamic learning rates.

Although the average score of all metrics for the hybrid learning approach are poor, it must be noted that the best run for the hybrid approach produced results that are only just worse than the standard RPROP and Adam optimisers.

It is interesting to note that some runs that on 4 of the 6 runs performed poorly (below 70%), the metric values collected for these runs. This potentially indicates that there is some factor that hindered these particular runs.

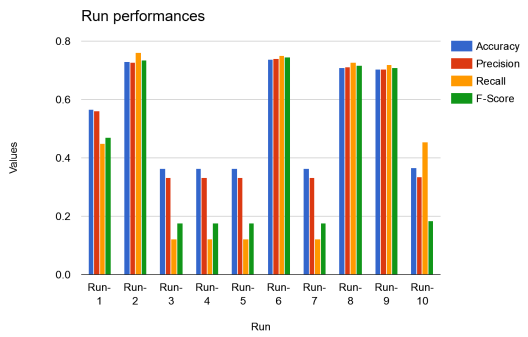

Fig. 7. Bar graph showing the results of the ten runs for the hybrid learning approach

V. Conclusion

In conclusion, based on the results, from this report we can see that generally, Resilient Back-propagation (RPROP) optimiser outperforms the Adam optimiser in both, however, in top runs the Adam optimiser comes on top performing manners in *accuracy*, *recall*, *precision* and *F-score*. The experiment run for the hybrid learning approach revealed that by combining RPROP and Adam the model generally yielded sub-optimal results, suggesting that the differing gradient update rules from these optimizers may introduce instability during training. However, its best run provided insight into the hybrid approach's potential. In future work an exploration with other optimisers that have complementary gradient schemes should be explored.

References

[1] Issitt RW, Cortina-Borja M, Bryant W, Bowyer S, Taylor AM, Sebire N. Classification Performance of Neural Networks Versus Logistic Regression Models: Evidence From Healthcare Practice. Cureus. 2022 Feb 21;14(2):e22443. doi: 10.7759/cureus.22443. PMID: 35345728; PMCID: PMC8942139

[2] Almond Type Classification. (2024). Sohaib Moradi. Kaggle. https://www.kaggle.com/datasets/sohaibmoradi/almond-types-classification

[3] Choudhary, Chetan; Kale, Atharva ; Rajput, Jaideep; Meshram, Vishal; Meshram, Vidula (2023), "Dry Fruit Image Dataset", Mendeley Data, V1, doi: 10.17632/yfhgn8py5f.1, https://www.kaggle.com/datasets/warcoder/dry-fruits-image

[4] Riedmiller, M. and Braun, H., 1993, March. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In IEEE international conference on neural networks (pp. 586-591). IEEE.

[5] Stanford.edu. (2019). Unsupervised Feature Learning and Deep Learning Tutorial. [online] Available at: http://deeplearning.stanford.edu/tutorial/supervised/OptimizationStochasticGradientDescent/.

[6] Jason Brownlee (2017). Gentle Introduction to the Adam Optimization Algorithm for Deep Learning. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/.

[7] Kingma, D.P., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[8] Mehdi Ghasri (2024). Adaptive moment estimation (Adam) (https://www.mathworks.com/matlabcentral/fileexchange/136679-adaptive-moment-estimation-adam), MATLAB Central File Exchange. Retrieved October 2, 2024.

[9] Brownlee, J. (2019). A Gentle Introduction to Cross-Entropy for Machine Learning. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/cross-entropy-for-machine-learning/.