Hybrid Learning in Neural Networks for 3-Class Almond Classification

Jake Weatherhead
University of Pretoria
jakeweatherhead1@gmail.com

Abstract— **This report investigates the effectiveness of various neural network configurations for optimising model performance on a three-class almond classification problem using a hybrid-learning implementation. A series of experiments were conducted to fine-tune both the architecture and hyperparameters, exploring a range of training algorithms. Notably, hybrid-learning significantly improved the learning process on unseen test cases versus using only a single training algorithm. The impact of the Cosine Annealing Warm Restarts (CAWR) learning rate scheduler was also tested leading to further performance improvements. An analysis is presented of the conducted experiments, detailing the interactions between architectures, optimization strategies and hyperparameter configurations as well as the observed benefits of incorporating synthetic data in training.**

I. INTRODUCTION

The given dataset contains 2,803 almond instances in CSV format, each described by length, width, and other size-related metrics recorded from a previous collection of almond images. The task was to build a neural network that could accurately classify almond instances into one of three types—Mamra, Sonora, and Regular. This required iterative experimentation and fine-tuning to minimize loss and overfitting and maximize accuracy on unseen test cases.

After reviewing the findings in [1], which detail the effectiveness of large-scale datasets for training neural networks, it was decided to augment the original dataset with 20,000 additional rows of synthetic data. As will be shown, this augmentation resulted in significant improvements in the classifier's performance when tested on unseen almonds from the original dataset of real-world observations.

Initially, the characteristics of the original dataset were analyzed and it was found that the three almond types are evenly represented, with virtually no skew toward any particular almond-type. Then, the per-feature value distributions were visualized to establish a baseline to adhere to when imputing missing values. Additionally, the number of missing values and outliers per feature was recorded.

The chosen method of mitigating outliers was the Z-score method, which was found to have the greatest benefit on performance amongst the options for handling outliers. Missing values were handled using K-nearest neighbours imputation, which proved effective in providing the model with more complete training information while also preserving the underlying distribution of the affected feature.

After preprocessing the data, an optimal architecture was selected via cross-validation and the hyperparameters were fine-tuned to maximize performance on the validation set.

At the core of this report, a hybrid learning implementation was developed that allowed the averaging of two or more optimization algorithms used in tandem. By averaging their outputs, updates to the network proved more effective than when any single optimization algorithm was used on its own. As will be discussed in a subsequent section, using Adam and RProp in this hybrid implementation maximized the model's performance on unseen almond cases.

A grid search was performed across various learning rates and epoch counts to gain insight into configurations that might yield optimal performance on unseen data.

A final round of manual fine-tuning led to the best observed performance accuracy on unseen test data, reported on in a later section.

II. BACKGROUND

Accurate and automated almond classification helps to streamline agricultural sorting processes, directly impacting the quality of products delivered to consumers. By leveraging neural networks, it is possible to classify almond types—Mamra, Sanora, and Regular—with significantly higher accuracy and efficiency than a human. Neural networks can not only increase operational speed but also reduce human errors in harvesting and distribution.

Neural networks are ideal for almond-classification due to their capacity to model complex, non-linear relationships between input features and target labels. This study explores the use of a feedforward neural network for almond classification, utilising ten numeric features derived from a previous dataset of two-dimensional almond images. These features capture geometric properties of the almonds, such as length, width, and thickness, which form the basis for classification. Given the challenges posed by missing values and the variety of feature scales, data preprocessing played a crucial role in ensuring the neural network could effectively learn meaningful patterns from the data.

The primary objective of this report is to evaluate and compare three gradient-based training algorithms: Resilient Backpropagation (RProp) [2], Adam (Adaptive Moment Estimation) [3], and Root Mean Square propogation (RMSProp)

[4] when used individually and together in hybrid-learning pairs.

RMSProp adjusts the learning rate for each parameter based on the average of recent squared gradients, effectively preventing the oscillations that can occur with standard gradient descent. This allows for better convergence, particularly in problems with non-stationary objectives.

RProp focuses on the sign of the gradient rather than its magnitude, adjusting the step size for each weight individually based on whether the gradient's sign changes. This approach avoids issues related to vanishing or exploding gradients, improving the training process for certain types of networks.

Adam combines the benefits of RMSProp and momentum by computing both running averages of the gradient and its square. It adapts the learning rate for each parameter, leading to faster convergence and better performance in practice across a wide range of problems.

A crucial aspect of building an effective neural network is the initialisation of weights and biases. Proper initialisation can significantly impact the model's ability to converge and avoid issues such as vanishing or exploding gradients, which can hinder learning in deeper networks. In this study, we uniform-Xavier weights initialisation was implemented, which ensures that the initial values are scaled to avoid large or small gradients during backpropagation. Bias terms are initialised to small non-zero values to provide an offset for the hyperplane that the model iteratively constructs and improves, ensuring that the model does not prematurely squash activations in the early stages of training. This careful initialisation is essential for setting the network on a path to stable and efficient learning, reducing the likelihood of poor convergence or suboptimal performance and had great bearing on the performance results observed.

To enable the neural network to differentiate between the three almond types, CrossEntropy loss was employed as the loss function. CrossEntropy is well-suited for multi-class classification problems as it measures the performance of a classification model whose output is a probability value between 0 and 1. It penalises incorrect classifications more harshly than other loss functions, encouraging the network to improve over successive iterations by optimising the weight adjustments.

The network's architecture uses the sigmoid activation function for its hidden layers. sigmoid is a common activation function in feedforward networks, which transforms input values into an output range between 0 and 1. Its non-linearity allows the network to learn complex relationships, but its tendency to cause vanishing gradients must be carefully managed, especially when deeper networks are used. For the final layer, the softmax activation function is employed. softmax transforms the outputs of the network into a probability distribution over the three almond classes, ensuring that the sum of the outputs is equal to 1. This makes it particularly effective for multi-class classification, as it allows the model to assign higher confidence scores to the correct class and lower probabilities to the incorrect ones.

In this study, the performance of the neural network across different configurations is assessed, focusing on both training and testing accuracy, as well as the stability of convergence. A critical part of the evaluation involves running multiple independent simulations to compute averages and standard deviations for each optimizer and the hybrid approach. This analysis will include K-fold cross-validation to ensure that the model generalizes well across different subsets of the data.

By combining RProp and Adam in a hybrid-learning approach, along with leveraging CrossEntropy loss and appropriate activation functions like sigmoid and softmax, we aim to improve the model's classification performance. The comparison of these optimizers and their hybrid will provide insights into their relative strengths and weaknesses, potentially leading to more efficient training procedures in neural network models applied to agricultural classification tasks.

III. EXPERIMENTAL SET-UP

The described neural network was built using PyTorch version 2.4.0+cu121 and trained on a local RTX GeForce 3080 Ti graphics processing unit. The Pandas module was used for data manipulation and, as will be explained later, SciKit-Learn's K-Nearest Neighbors imputer was used to impute the missing values. To ensure reproducibility of results across simulations, a random value of 42 was used to seed PyTorch before any operations were executed in code.

A. Data Preprocessing

The first task was to gain an understanding of the characteristics and limitations of the dataset in order to be able to preprocess the data. Missing values were detected with the following frequencies:

Feature	Missing Values	Proportion (%)
Length (major axis)	857	30.57%
Width (minor axis)	942	33.61%
Thickness (depth)	1004	35.82%
Area	0	0%
Perimeter	0	0%
Roundness	857	30.57%
Solidity	0	0%
Compactness	0	0%
Aspect Ratio	**1799**	**64.18%**
Eccentricity	**1799**	**64.18%**
Extent	0	0%
Convex hull (convex area)	0	0%
Type	0	0%

Due to the high frequency of missing values in **Aspect Ratio** and **Eccentricity**, these two features were dropped from the dataset.

Missing values in the remaining features were imputed using a K-Nearest Neighbour imputer provided out-of-the-box by SciKit-Learn, which preserved the underlying ditributions of each feature as verified by plots and kurtosis values generated before and after imputation.

Outliers were detected using the inter-quartile range method, which identifies outliers as data points greater than

1.5 times the interquartile range (IQR) above the third quartile or less than 1.5 times the IQR below the first quartile. The following table enumerates the frequency of outliers per feature:

Feature	Value	Proportion (%)
Length (major axis)	16	0.57%
Width (minor axis)	9	0.32%
Thickness (depth)	22	0.78%
Area	87	3.10%
Perimeter	61	2.18%
Roundness	0	0%
Solidity	208	7.42%
Compactness	209	7.46%
Extent	112	3.99%
Convex hull (convex area)	89	3.17%

The influence of outliers in all features was mitigated using the Z-score scaling method where each feature-value X_f in all features F underwent the following transformation:

$$X_f \rightarrow Z_f = \frac{X_f - \mu_F}{\sigma_F}$$

The almond-type label was encoded using three-way class indexing where 0 represented Mamra almonds, 1 represented Sonora almonds, and 2 represented Regular almonds.

Class-indexing was chosen because it directly aligns each of the three output layer neurons with a specific almond type, making class indexing ideal for generating clear and interpretable probabilities with the softmax function.

B. Architecture Selection

The chosen network architecture followed a pyramidal structure, gradually compressing data across its layers on each forward pass.

The input layer contained ten neurons, one for each input feature in the preprocessed dataset, followed by a hidden layer with 20 neurons, a second hidden layer with 29 neurons, a third hidden layer with 19 neurons and a final fourth hidden layer with 13 neurons. Finally, the output layer had three neurons, corresponding to the three almond-types.

Each hidden layer neuron was activated using the sigmoid activation function and the output layer was activated using the softmax activation function.

Although this architecture demonstrated effectiveness, it is likely that with additional experimentation, computational resources and a more sophisticated procedure of architecture selection, a more optimal design could be found.

C. Weights and Biases

All weights in the network were initialized using PyTorch's Xavier uniform initialization function and a random state of 42. All biases were set to a small positive constant of 0.01.

D. Activation Function

Sigmoid was utilized in the hidden layers to introduce non-linearity and differentiability for backpropogation. It scales the output of each neuron to a range between zero and one, which is useful for models that rely on probabilistic interpretations of neuronal outputs.

For the output layer, the softmax converted the raw output logits into three probabilities, one for each almond-type, that sum to one. The output neuron with the highest probability score was taken as the model's classification for the almond instance.

E. Learning Rate & Number of Epochs

The learning rate (η), a hyperparameter that influences how a model updates its free parameters during training, was evaluated under three different setups.

Initially, a constant $\eta = 0.007$ was used across all epochs. Subsequently, PyTorch's exponential decay learning rate scheduler was used, decreasing η as the number of epochs increased.

However, the most effective strategy proved to be the Cosine Annealing Warm Restarts (CAWR) provided by PyTorch, which periodically resets η, with the first reset at the tenth epoch and subsequent resets doubling the epoch count ($T_0 = 10$, $T_{mult} = 2$).

The optimal initial η for CAWR, set at 0.007, was determined through grid search and fine-tuning.

Regarding epoch counts, a grid search indicated that fewer epochs might be more effective. After further trials and adjustments, the optimal number of epochs was established at 600.

F. Data Augmentation (Gretel.AI)

Having read [1], it was deemed worthwhile to trial the augmentation of the training set with 20,000 additional rows of synthetic data using the online tool from Gretel AI: Navigator Fine-Tuning [5].

The synthetic data were generated over two separate runs using the same preprocessed version of the original dataset as context, where the first run generated 5,000 new rows of data and the second run generated 15,000.

The synthetic data were used exclusively for training and never for validation or testing as the synthetic dataset likely deviates from reality in its representations of almonds.

G. Train, Validation, and Test Data

Splitting the dataset into training, validation and test sets was performed using SciKit-Learn's train_test_split function using a random seed of 711.

First, the preprocessed dataset, excluding synthetic data, was split along an 80%-20% partition.

The 20% portion was then divided equally among the validation and test sets each of which comprised 10% of the preprocessed dataset.

Finally, the synthetic data were vertically concatenated below the original 80% partition to form a training dataset of 22,242 rows, a validation set of 280 ± 1 rows and a test set of 280 ± 1 rows. No further splitting or data augmentation occurred. A batch-size of 64 was used. All claims of configuration-superiority will be statistically motivated in the subsequent section.

IV. RESEARCH RESULTS

This section presents the experimental results obtained from various configurations of the neural network, including hyperparameter optimization, comparisons of training algorithms, and the performance of the hybrid learning algorithm.

All experiments were evaluated using a 80(+ synthetic data)-10-10 train-validation-test split, with accuracy and loss on unseen data as the primary performance metrics.

Reproducibility of results was an early focus in the implementation of this study; given the same seed and configuration, the model repeats the same results across all epochs. For this reason, each trial of ten independent samples cycled, in order, through the following integers, [42, 43, 44, 45, 46, 47, 48, 49, 50, 51] where the first integer was used as the global seed for the first sample, the second integer for the second sample, and so on. This ensured that results were not repeated while also preserving the model's underlying configuration for experimental validity.

Additionally, results from paired t-tests, ANOVA (Analysis of Variance) and Tukey-HSD (Honest Significant Difference) tests of statistical significance are provided to substantiate claims of configuration-superiority. When conducting tests of statistical significance, all variables except the independent variable were held constant according to the specification outlined in the experimental set-up section.

Paired t-tests are used to compare the means of two related groups, typically measuring the same subjects before and after the independent variable is modified. It calculates the difference between each pair of observations and assesses whether the mean difference is significantly different from zero. The test produces a t-statistic (t_{stat}), which measures the size of the difference relative to the variability in the data. A p-value is then derived to determine whether the observed difference is likely due to chance, with a p-value below a chosen threshold (in this study, $\alpha = 0.5$) indicating a significant difference between the paired groups.

The ANOVA test is used to determine if there are significant differences between the means of three or more groups. It calculates an F-statistic, which is the ratio of the variance between group means to the variance within groups, indicating how much the group means differ relative to random variation. A higher F-statistic suggests greater differences between groups. The p-value measures the probability that the observed differences occurred by chance, and if it is below a chosen significance level ($\alpha = 0.5$), the null hypothesis of equal means is rejected and we conclude that the differences observed are statistically significant.

The Tukey-HSD test is a post-hoc analysis (used after an initial statistical test) that is applied after ANOVA to identify which specific group means are significantly different from each other. It calculates the mean differences between pairs of groups and compares them to a critical value, accounting for multiple pairwise comparisons and controlling for Type I error (incorrectly rejecting a true null hypothesis). A larger difference than the critical value indicates a significant difference between those group means. The test provides adjusted p-values for each comparison to control the family-wise error rate (FWER - the probability of making at least one Type I error), helping ensure the results are statistically valid.

The *meandiff* column shows the difference in means between two groups, while *p-adj* is the adjusted p-value accounting for multiple comparisons, indicating the significance of the observed difference. The *lower* and *upper* columns provide the bounds of the confidence interval, showing the range within which the true mean difference likely falls. Together, these values assess the statistical significance and precision of the mean difference between groups.

A. Grid Search: Number of Epochs by Learning Rate

Preliminary tests identified the ranges $\eta = [0.004, 0.1]$ and number of epochs = $[450, 750]$ as being worth further investigation via a grid-search.

K-fold validation was used with $k = 3$ and involved splitting the dataset into three equal folds. During the grid-search the model was trained on two of these folds in addition to all 20,000 rows of synthetic data and tested on the remaining fold. Each of the three independent runs on a configuration used a different rotation of the three folds.

The following plot visualizes the mean validation accuracies obtained across three independent runs for all 49 ($\eta\times$ number of epochs) pairs where the total number of simulations conducted for the grid search was 147:

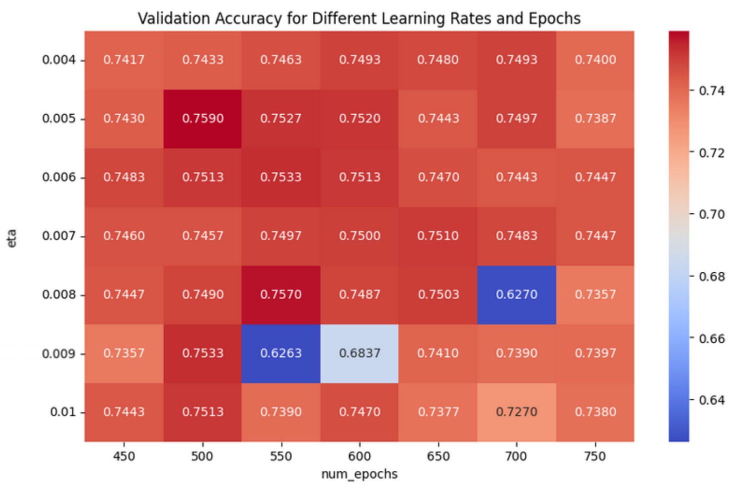

Fig. 1. Grid search: mean results on accuracy of three independent runs per configuration.

The grid-search results highlighted several promising configurations whereafter the epoch range $[550, 600]$ was investigated. After further cross-validation and fine-tuning of the model, the highest accuracy was observed using 600 epochs with $\eta = 0.007$.

B. Dropout Usage

The usage of dropout applied to each hidden layer, with probability of dropout $p = 0.5$, and its effect on the model's performance was tested using a paired t-test ($\alpha = 0.05$).

The mean accuracy achieved when dropout was *enabled* was 0.644 (σ = 0.018) with a mean loss of 0.784 (σ = 0.019).

The mean accuracy achieved when dropout was *disabled* was 0.757 (σ = 0.02) with a mean loss of 0.628 (σ = 0.048).

The paired t-test results on accuracy ($t_{\text{stat}} = -12.547$, p-value = 5.261×10^{-7}) indicate that there is a significant difference between the accuracies observed when dropout was enabled versus when it was disabled.

The paired t-test results on loss (t_{stat} = 8.192, p-value = 1.831×10^{-5}), indicate that there is a significant difference between the losses observed when dropout was enabled versus when it was disabled.

Therefore, for this study's implementation, using dropout significantly diminished the model's performance on unseen almond-cases in terms of accuracy and loss.

C. Batch Size

The model was tested using three batch-size configurations—32, 64, and 128. The observed performance metrics are as follows:

Batch Size	**Accuracy**		**Loss**	
	μ	σ	μ	σ
32	0.757	0.020	0.628	0.048
64	0.776	0.014	0.564	0.040
128	0.780	0.013	0.531	0.022

The ANOVA results on accuracy (F-stat = 5.229, p-value = 0.012) indicate that there is a significant difference between the observed accuracies between the batch-sizes.

The Tukey-HSD (FWER = 0.05) results on accuracy are as follows:

Batch Size 1	Batch Size 2	Mean Diff	p-adj	Lower	Upper	Reject
32	64	0.0189	0.0465	0.0003	0.0375	True
32	128	0.0227	0.0147	0.0041	0.0413	True
64	128	0.0038	0.8694	-0.0148	0.0224	False

These results indicate that there are significant differences in the observed accuracies between batch-sizes 32 and 64, and between 32 and 128 but no significant difference between batch-sizes 64 and 128.

The ANOVA results on losses (F-stat = 14.836, p-value = 4.497×10^{-5}) indicate that there is a significant difference between the observed losses between the batch-sizes.

The Tukey-HSD (FWER = 0.05) results on losses are as follows:

Batch-Size 1	Batch-Size 2	Mean Diff	p-adj	Lower	Upper	Reject
32	64	-0.0633	0.0044	-0.1080	-0.0186	True
32	128	-0.0967	0.0	-0.1414	-0.0520	True
64	128	-0.0334	0.1722	-0.0781	0.0113	False

The Tukey-HSD results indicate that there were significant differences in loss between batch-sizes 32 and 64, and between 32 and 128 but no significant difference between batch-sizes 64 and 128.

D. Weight Initialization

Two built-in PyTorch weight initialization strategies were tested, namely He-Kaiming uniform and Xavier uniform. He-Kaiming initializes weights to values sampled from a uniform distribution scaled by the inverse square root of the number of input units, optimized for ReLU activations. Xavier uniform initializes weights to values sampled from a uniform distribution scaled by the inverse square root of the sum of input and output units, optimized for sigmoid or hyperbolic-tangent (tanh) activations.

Both strategies were tested in isolation using a paired t-test ($\alpha = 0.05$) to assess the statistical significance of their effect on the model's performance.

The mean accuracy achieved when using Xavier uniform was 0.757 (σ = 0.02) with a mean loss of 0.628 (σ = 0.0048).

The mean accuracy achieved when using He-Kaiming uniform was 0.768 ($\sigma = 0.017$) with a mean loss of 0.607 (σ = 0.07)

The paired t-test results on accuracies ($t_{\text{stat}} = -1.029$, p-value = 0.331) indicate that there is no significant difference between the observed accuracies.

The paired t-test results on losses (t_{stat} = 0.749, p-value = 0.473) indicate that there is no significant difference between the observed losses.

E. Hybrid-Learning

A comparison of the RProp, Adam, and RMSProp training algorithms was conducted to determine the performance effects of using each algorithm individually. The observed performance metrics are as follows:

Algorithm	**Accuracy**		**Loss**	
	μ	σ	μ	σ
Adam	0.776	0.014	0.564	0.040
RProp	0.601	0.024	0.872	0.024
RMSProp	0.776	0.013	0.552	0.036

The ANOVA results on accuracies (F-stat = 297.998, p-value = 3.961×10^{-19}) indicate that there is a significant difference between the accuracies observed.

The Tukey-HSD (FWER = 0.05) results on accuracy are as follows:

Group 1	Group 2	Mean Diff	p-adj	Lower	Upper	Reject
Adam	RProp	-0.1747	0.0	-0.1952	-0.1542	True
Adam	RMSProp	0.0003	0.9993	-0.0202	0.0208	False
RProp	RMSProp	0.1750	0.0	0.1545	0.1955	True

The Tukey-HSD results on accuracies indicate that a significant difference in accuracy exists between Adam and RProp, and between RMSProp and RProp but not between Adam and RMSProp.

The ANOVA results on losses (F-stat = 256.021, p-value = 2.795×10^{-18}) indicate that there is a significant difference between losses observed.

The Tukey-HSD (FWER = 0.05) results on losses are as follows:

Group 1	Group 2	Mean Diff	p-adj	Lower	Upper	Reject
Adam	RProp	0.3072	0.0	0.2675	0.3469	True
Adam	RMSProp	-0.0125	0.7177	-0.0522	0.0272	False
RProp	RMSProp	-0.3197	0.0	-0.3594	-0.2800	True

The Tukey-HSD results on losses indicate that significant differences in loss exist between Adam and RProp, and between RMSProp and RProp but not between Adam and RMSProp.

Then, a hybrid-learning implementation was used to compare RProp, Adam, and RMSProp in pairs. The observed performance metrics are as follows:

Algorithm Pair	**Accuracy**		**Loss**	
	μ	σ	μ	σ
RProp + RMSProp	0.778	0.013	0.542	0.024
Adam + RMSProp	0.767	0.016	0.599	0.444
RProp + Adam	0.784	0.018	0.534	0.034

The ANOVA results on accuracy (F-stat = 2.862, p-value = 0.075) indicate that there is no significant difference between the observed accuracies.

The Tukey-HSD (FWER = 0.05) results on accuracy are as follows:

Group 1	Group 2	Mean Diff	p-adj	Lower	Upper	Reject
RProp + RMSProp	Adam + RMSProp	-0.0115	0.2924	-0.0301	0.0071	False
RProp + RMSProp	RProp + Adam	0.0062	0.6905	-0.0124	0.0248	False
Adam + RMSProp	RProp + Adam	0.0177	0.0647	-0.0009	0.0363	False

The Tukey-HSD results on accuracies reiterate the findings of the ANOVA test, namely that no significant difference exists between the mean observed accuracies.

The ANOVA results on losses (F-stat = 9.314, p-value = 0.00084) indicate that a significant difference exists between the losses observed.

The Tukey-HSD (FWER = 0.05) results on losses are as follows:

Pair 1	Pair 2	Mean Diff	p-adj	Lower	Upper	Reject
RProp + RMSProp	Adam + RMSProp	0.0573	0.0048	0.0164	0.0982	True
RProp + RMSProp	RProp + Adam	-0.0079	0.8817	-0.0488	0.033	False
Adam + RMSProp	RProp + Adam	-0.0652	0.0014	-0.1061	-0.0243	True

The Tukey-HSD results indicate that there is a significant difference between the losses of [*RProp*, *RMSProp*] and [*Adam*, *RMSProp*], that no significant difference exists between the losses of [*RProp*, *RMSProp*] and [*RProp*, *Adam*] and that there was a significant difference between the losses of [*Adam*, *RMSProp*], and [*RProp*, *Adam*].

According to these results, [*Adam*, *RMSProp*] performs the worst in this study's hybrid-learning implementation.

Finally, a comparison is made between the individual performance effects of the individual training algorithms versus their effects when used in hybrid-learning pairs.

The ANOVA results on accuracy between the six test groups (F-stat = 164.712, p-value = 2.085×10^{-31}) indicate that there is a significant difference between the mean observed accuracies.

The Tukey-HSD (FWER = 0.05) results on accuracy are as follows:

Group 1	Group 2	Mean Diff	p-adj	Lower	Upper	Reject
Adam	R + RMS	0.0022	0.9998	-0.0211	0.0255	False
Adam	A + RMS	-0.0093	0.8455	-0.0326	0.0140	False
Adam	R + A	0.0084	0.8935	-0.0149	0.0317	False
RProp	R + RMS	0.1769	0.0	0.1536	0.2002	True
RProp	A + RMS	0.1654	0.0	0.1421	0.1887	True
RProp	R + A	0.1831	0.0	0.1598	0.2064	True
RMSProp	R + RMS	0.0019	0.9999	-0.0214	0.0252	False
RMSProp	A + RMS	-0.0096	0.8274	-0.0329	0.0137	False
RMSProp	R + A	0.0081	0.9073	-0.0152	0.0314	False

The Tukey-HSD results indicate that a significant improvement in accuracy was gained by using any of the investigated hybrid-learning pairs when compared to the performance when using the RProp training algorithm alone.

By the ANOVA results on losses between the six groups, we reject the null hypothesis (F-stat = 127.851, p-value = 1.181×10^{-28}), i.e. there is a significant difference between the mean losses observed between pairs of training algorithms and individually-used training algorithms.

The Tukey-HSD (FWER = 0.05) results on losses are as follows:

Group 1	Group 2	Mean Diff	p-adj	Lower	Upper	Reject
Adam	R + RMS	-0.0224	0.7394	-0.0704	0.0256	False
Adam	A + RMS	0.0349	0.2791	-0.0131	0.0829	False
Adam	R + A	-0.0303	0.4345	-0.0783	0.0177	False
RProp	R + RMS	-0.3296	0.0	-0.3776	-0.2816	True
RProp	A + RMS	-0.2723	0.0	-0.3203	-0.2243	True
RProp	R + A	-0.3375	0.0	-0.3855	-0.2895	True
RMSProp	R + RMS	-0.0099	0.9899	-0.0579	0.0381	False
RMSProp	A + RMS	0.0474	0.0548	-0.0006	0.0954	False
RMSProp	R + A	-0.0178	0.8811	-0.0658	0.0302	False

The Tukey-HSD results indicate that a significant reduction in loss occurred when using any of the investigated hybrid-learning pairs when compared to the performance of using the RProp training algorithm alone.

F. Data Augmentation

20,000 sythetic rows of additional training data were generated to augment the original almond dataset of 2,803 rows. The performance effects of incorporating the synthetic data during training were tested via the ANOVA and Tukey-HSD statistical significance tests. The observed performance metrics are as follows:

Synthetic Utilisation	**Accuracy**		**Loss**	
	μ	σ	μ	σ
0 (0%)	0.742	0.023	0.779	0.074
10,000 (50%)	0.765	0.017	0.651	0.064
20,000 (100%)	0.776	0.014	0.564	0.040

The ANOVA results on accuracy (F-stat = 8.227, p-value = 0.001) indicate that there is a significant difference between the observed accuracies.

The Tukey-HSD (FWER = 0.05) results on accuracy are as follows:

Util. 1	Util. 2	Mean Diff	p-adj	Lower	Upper	Reject
0%	50%	0.023	0.0325	0.0017	0.0443	True
0%	100%	0.0342	0.0013	0.0129	0.0555	True
50%	100%	0.0112	0.4059	-0.0101	0.0325	False

The Tukey-HSD results on accuracy indicate that there is a significant difference in accuracies observed when using none of the synthetic data versus using half of the synthetic data.

The results also indicate that there is a significant difference in observed accuracies between using half of the synthetic data versus using all 20,000 rows.

However, no significant difference exists in accuracy between when the training set is augmented with half of the synthetic data versus when the training set is augmented with all of the synthetic data. This is useful information because training a model on 10,000 rows of CSV data is far less computaionally expensive than training a model on 20,000 rows of CSV data.

The ANOVA results on loss, (F-stat = 33.794, p-value = 4.463×10^{-8}) indicate that there is a significant difference between the observed losses.

The Tukey-HSD (FWER = 0.05) results are as follows:

Util. 1	Util. 2	Mean Diff	p-adj	Lower	Upper	Reject
0%	50%	-0.1482	0.0001	-0.2197	-0.0767	True
0%	100%	-0.2342	0.0	-0.3057	-0.1627	True
50%	100%	-0.0860	0.0159	-0.1575	-0.0145	True

The Tukey-HSD results on observed losses indicate that there is a significant difference in losses observed when using none of the synthetic data versus half of the synthetic data. They also indicate that there is a significant difference in losses observed when using half of the synthetic data versus all.

However, there is no significant difference in loss between when the training set is augmented with half of the synthetic data versus when the training set is augmented with all of the synthetic dataset.

G. Learning Rate Scheduler

The model was tested using three learning-rate scheduler configurations—a constant learning rate ($\eta = 0.007$), exponential decay, and CAWR. The observed performance metrics are as follows:

Learning Rate Schedule	**Accuracy**		**Loss**	
	μ	σ	μ	σ
Constant ($\eta = 0.007$)	0.748	0.012	0.594	0.029
Exponential Decay	0.748	0.011	0.599	0.011
CAWR	0.757	0.020	0.628	0.048

The ANOVA results on accuracy ($t_{\text{stat}} = 1.089$, p-value = 0.351) indicate that there is no significant difference between the observed accuracies.

The Tukey-HSD (FWER = 0.05) results on accuracies are as follows:

LRS 1	LRS 2	Mean Diff	p-adj	Lower	Upper	Reject
Constant	Exp.	-0.0001	0.9999	-0.0173	0.0171	False
Constant	CAWR	0.0088	0.4232	-0.0084	0.0260	False
Exp.	CAWR	0.0089	0.4153	-0.0083	0.0261	False

The Tukey-HSD results indicate that there are no significant differences in observed accuracies between the three learning-rate schedulers tested in this study.

The ANOVA results on losses ($t_{\text{stat}} = 2.784$, p-value = 0.082) indicate that no significant differences exists between the observed losses.

The Tukey-HSD (FWER = 0.05) results are as follows:

LRS 1	LRS 2	Mean Diff	p-adj	Lower	Upper	Reject
Constant	Exp.	0.0055	0.9329	-0.0329	0.0439	False
Constant	CAWR	0.0338	0.0921	-0.0046	0.0722	False
Exp.	CAWR	0.0283	0.1793	-0.0101	0.0667	False

The Tukey-HSD results indicate that there are no significant differences in observed losses between any of the learning-rate schedulers when tested in this study.

H. Key Findings

The highest accuracy achieved on unseen test data was 82.2% with a loss of 0.49. Notably, training loss was higher than validation loss and validation loss higher than test loss when all 20,000 rows of synthetic data were added to the training set.

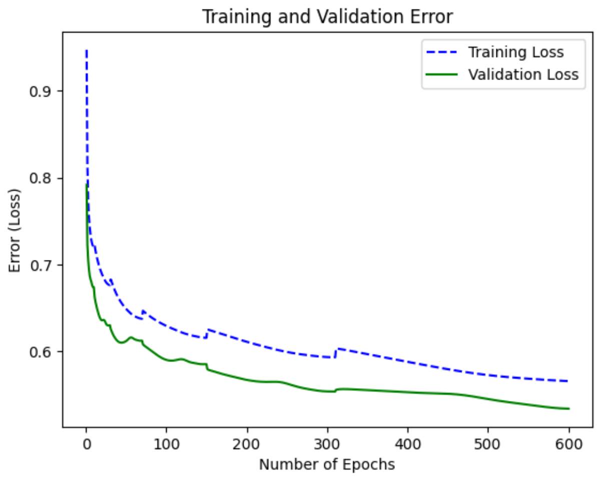

Fig. 2. Loss by Number of Epochs on Best Run

A reason for this may be because the synthetic data introduced noise that the model struggled to generalize during training, leading to overfitting on the training set but improved performance on the unseen validation and test sets due to the added diversity of the additional data.

V. CONCLUSION

While the Adam and RMSProp training algorithms performed comparitively well when used individually, all of the investigated hybrid-learning strategies significantly improved accuracy and reduced loss when compared to the isolated use of RProp.

The results in this study also support the findings in [1], that more data is effective in improving the performance of a neural network. It was shown that adding 10,000 rows of synthetic data to the training process significantly improved the accuracy and reduced the loss of the model implemented in this study versus using no synthetic data.

However, this study also found that augmenting the training set with 10,000 rows of synthetic CSV data was statistically similar to using 20,000 rows of synthetic data.

Having achieved an unseen accuracy of 82.2% with a loss of 0.42, the hyperparameter tuning and search for optimal model configuration, motivated by evidence, paid dividends.

References

[1] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," 2017. [Online]. Available: https://arxiv.org/abs/1707.02968

[2] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: the rprop algorithm," in *IEEE International Conference on Neural Networks*, 1993, pp. 586–591 vol.1.

[3] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017. [Online]. Available: https://arxiv.org/abs/1412.6980

[4] G. Hinton, "Lecture 6e rmsprop: Divide the gradient by a running average of its recent magnitude," https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf, 2012, neural Networks for Machine Learning, Coursera.

[5] [Online]. Available: https://docs.gretel.ai/create-synthetic-data/models/synthetics/gretel-navigator-fine-tuning