# Image Classification Using CIFAR-10 Dataset

## CP322 Final Project Report

Abigail Lee
Maheep Jain
CP322-B: Machine Learning
Sukhjit Singh Sehra
April 6th, 2023

# Table of Contents

# Abstract

The focus of this project is to examine the accuracy of different models using the CIFAR-10 dataset, a widely used dataset used for machine learning research. The models focused on for this research includes a Logistic Regression model, a Random Forest model, and a Convolutional Neural Networks (CNN) model. We trained each model on the same dataset and evaluated their performance using various metrics such as accuracy and loss. The objective was to find a model that would return an accuracy level above 90%. We found that the CNN model outperformed the two simpler models, based on the comparison of their accuracy scores. As we were trying to determine whether it was possible to use a simplistic model to attain an efficient solution to the classification problem, we saw that the deep learning algorithms can better predict the class of an image.

# Introduction to Project

## Overview

Image classification is the process of assigning a label of a class to an entire image, where images are expected be classified into one predefined class based on visual content. Image classification essentially takes an image as an input and returns a prediction on which class it belongs to. Image classification plays a vital role in various fields and accurate image classification models can significantly improve the functions and efficiency.

This project will examine the accuracy of different classification models by using the CIFAR-10 data. This dataset is a condensed version of the CIFAR-100 dataset that was created by the Canadian Institute for Advanced Research. This image dataset consists of 60,000, 32x32 images with 10 classes of images and 6000 images per class (see Figure 1).

We will be focusing on Logistic Regression, Random Forest, and CNN models and determine whether there is potential to improve the accuracy of image classification using these three models. We will then discuss and compare the performance of these models to determine whether there is a best suited model for this specific dataset.

## Existing System

Today there exist several image classification systems that have been developed over the last decade. These include deep neural networks systems such as VGG Image Classification (VIC) Engine which is an open-source project developed at the Visual Geometry Group and released under the BSD-2 clause. VIC is a web application that serves as a web engine to perform image classification queries over a user-defined image dataset (Arbelaez, 2020). Other systems such as the ResNet-50 is a pretrained Deep Learning model for image classification of the CNN, which is a class of deep neural networks and most applied to analyzing visual imagery. This system is 50 layers deep and has over 23 million trainable parameters which indicates a deep architecture that makes it better for image recognition (*Christensen et al., 2019*). Both of these models have achieved high performance on a variety of image classification datasets. These models, however, can be complex and computationally expensive to train, which may limit their practical use in certain applications.

Simpler models such as Logistic Regression and Random Forest have been useful for image classification, however, may not be as accurate as some of the deep learning networks. Nonetheless, these models have the advantage and potential to be computationally efficient and require less training time than the deep learning networks that exist today. Therefore, this project aims to understand the functionality and efficiency of simpler models.

### Objectives of Project

The objective of this project is to examine the accuracy of different classification models using the CIFAR-10 dataset. The aim of the project is to develop a fast and high accuracy image classification model of over 90% that can aid in object recognition, surveillance systems, quality control, medical diagnosis, and other related digital image recognition applications. The project is set to focus on three recognized classification models, namely, Logistic Regression, Random Forest, and Convolutional Neural Networks (CNN), to determine the most efficient and accurate model to classify the CIFAR-10 dataset. The project intends to contribute to the development of efficient and accurate image classification models that can aid in digital image analysis applications.

## Pre-Processing and Exploratory Data Analysis

### Dataset Collection

A description of the data and what goals there were for the project (particularly for this data set) etc. If the data are publicly available, give the source.

CIFAR-10 dataset is a condensed version of the CIFAR-100 dataset which was created by the Canadian Institute for Advanced Research. The dataset consists of 60,000 images of 32x32 size with 10 classes of images and 6000 images per class. This dataset will help us in training and testing our classification models for Image Classification (see Figure 1)

### Data Pre-processing

In the dataset we chose, there are 10 different classes of color images of size 32x32. To train our model faster, we need to normalize the image. The pixel range of a color image is 0-225. We divided each pixel of the image by 255 so that the pixel range will be between 0-1.

Doing so to our train and test data, we can train our models faster since the data is now normalized. In the output of the shape, we see 4 values e.g. (50000, 32, 32, 3). The first value shows the number of images. The second and third value shows the image size, i.e., image height and width. Here the image size is 32x32. The fourth value is 3, which shows the RGB format since the images we are using are color images.

### Exploratory Data Analysis and Visualizations

While working with this dataset, the first model we worked on was Logistic Regression. Since this image classification problem is a multiclass classification problem, we converted our data to One-Hot Encode data, where we converted our categorical data variables so they can be provided to our model to improve predictions.

For Random Forest, we converted our image dataset from multi-dimensional image to one-dimensional input. We did this by flattening our test and train data. Doing so, each image is represented by a one-dimensional array. Hence, we matched our input data to match our output.

## Methodology

### Platform and Machine Configurations Used

To compile this project, we used Google Colab and Visual Studio Code (VSC).

### Data Split

To evaluate the performance of our models, we split the data in two parts. The first is test data which contains 50,000 images and the rest 10,000 images were a part of test data. Once we trained our models with train dataset, we evaluated their performances by the test dataset.

## Model Planning

In terms of model planning, selecting appropriate machine learning or deep learning models are a critical step. With the image classification problem at hand, it is important to evaluate different models and select ones that are best suited for the problem. There are multiple techniques that can be used to solve classification problems. This includes logistic regression, classification trees, discriminant analysis, neural networks, random forests, k-nearest neighbours, among others.

We chose to create a Logistic Regression model as this is a statistical model that is well-suited for binary classification. The goal is to classify a given data point (image) into one of two possible classes. Although in this project we are looking at the CIFAR-10 dataset, which consists of 10 classes of images, we can still use logistic regression to determine whether a particular image belongs to a class or not. Logistic regression is computationally efficient and requires less training data compared to deep learning networks such as CNN, therefore we can examine whether this simplistic model is beneficial for this problem in terms of speed and efficiency.

A Random Forest is another powerful image classification technique that can be used for image classification. It is a very versatile model that combines multiple decision trees to make accurate predictions. Random forest trains data quickly and can efficiently handle large datasets which is essential for image classification as their applications always have incoming data that requires a lot of time in the training process. Random forest models can also handle high dimensionality effectively which is known to be a large problem when it comes to image classification. This model has the potential to achieve a high accuracy (>90%) making it a suitable choice as a model for this problem.

CNNs are deep learning models that are specifically designed for image classification tasks. Since they can handle complex images with high dimensionality, CNN is an ideal candidate for this image classification problem. While a CNN model may take more computational time than the other models to train the data, it can learn to recognize complex patterns and features in the images. We would expect that a CNN model would perform very well and achieve a high accuracy (>90%) in this image classification problem.

## Model Training

To train our models, we use the train data which we created while splitting the data. Training the models means adjusting the model parameters to improve its accuracy. Every model is trained differently with different parameters and different methods.

For Logistic Regression, we created an empty sequential model object (see Figure 4). Then we added a fully connected dense layer to the model. The number of units we chose was 10, which specifies the number of output units in the layer. In our case, it is 10 which is one for each class in the CIFAR-10 dataset. The next parameter we specified was the activation parameter. Here we chose the parameter to be softmax because we are working on a multiclass classification problem. The next input will be the input shape. While feature building, we converted our data to one-hot encode which transformed our data to 1D array of length 3072 (32x32x3).

For Random Forest, to train our model, we first flattened our dataset so that our input matches the output. To find the best parameters, we used grid search (see Figure 6). Grid search helps in finding the best parameters for our dataset by running various permutations. We passed a dictionary of hyperparameters to be tuned using a grid search. The n_estimators parameter specifies the number of trees in the forest. The criterion parameter specifies the quality of thesplit, the max_depth parameter specifies the maximum depth of the tree. The

min_sample_leaf parameter specifies the minimum number of samples required to be at a leaf node, and the bootstrap parameter specifies whether or not to bootstrap the samples when building trees.

For CNN, to train our model, we added various layers to build the model. We started with convolutional layers and pooling layers. The use of convolutional layers with pooling helps capture local features in the input image while down sampling its size. We also added two dense layers. Adding these dense layers to the flattened output helped in capturing the higher-level features in the image.

## Model Evaluation

After training our model with training data, its time to find which tuning parameters are the best for our model. By testing the data with test data, we were able to see following results for our models:

For Logistic Regression, we already chose the best hyper tuning parameters while building the model, hence we got the evaluation with already the best parameters possible for this case.

For Random Forest, we were able to find the best hyper tuning parameters as can be shown in the figure (see Figure 2). We can then optimize our model with these hyperparameters.

For CNN, while testing, we ran into an issue of overfitting. This can be seen in the graph figure that our entropy loss was very high (see Figure 9). Hence, a need for optimization raised.

## Model Optimization

From the evaluations, we now will optimize our model based on the results of model evaluation. We need to optimize the models to improve its performance.

For Random Forest, to increase the performance of our model, we will tune our parameters with the results we got from Grid Search. Grid search gave us the best parameters for

the inputs we provided (see Figure 2). After tuning our model, we should be seeing the best performance from our model for this problem of image classification.

For CNN, we ran into an issue of overfitting, which gave us a very high value for entropy loss and a low accuracy rate than what we expected. To fix this issue, we added a dropout layer. This layer randomly drops 20% of the input units to prevent overfitting. This will help in increasing the performance of our model while reducing the entropy loss, and the issue of overfitting simultaneously (see Figure 5).

## Final Model Building

After optimizing our models, we can fit our models to predict the results and report the accuracies. Fitting the model means compiling our models after optimizing them. It is done differently for all our models.

For Logistic Regression, we compile our model by updating the weights of the model to minimize the categorical cross-entropy loss function using stochastic gradient descent (sgd) optimizer. The number of epochs used for training is set to 10, which means that the model will be trained on the entire training dataset 10 times. The model's performance is evaluated on the test dataset after each epoch. This is done to monitor the model's progress and to prevent overfitting. The training history contains information on the model's loss and accuracy values during each epoch of training. After fitting this model, we got a test accuracy of 35.09%.

For Random Forest, after tuning our model with the best hyper tuning parameters from the Grid Search, we will fit our model to predict our target class on the test data. We trained the model on the training data and their corresponding labels. Then, our trained model is used to predict the class labels for the test data. After predictions, the accuracy of the model is calculated

by comparing the predicted labels with the true labels. Doing this, our model calculated the prediction accuracy which turned out to be 42.86%.

For CNN, after adding the dropout layer, we will compile the model using the Adam optimizer and Sparse Categorical Cross-Entropy loss function. The model is then trained on the training data for 24 epochs and validated using the testing data. The model's performance is evaluated on the test dataset after each epoch. This is done to monitor the model's progress and to prevent overfitting. The training history contains information on the model's loss and accuracy values during each epoch of training. After fitting this model, we got a test accuracy of 81.84%.

# Results

## Description of the Models

### Model 1: Logistic Regression

The purpose of the Logistic Regression model is to predict the probability that a given image belongs to each of the 10 classes or not. The model estimates the coefficients of the linear equation that models the relationship between the three input features and the target variable, which in this case is the class label.

### Model 2: Random Forest

The Random Forest model combines multiple decision trees to make accurate predictions. The purpose of the model is to predict the class label of a given input image based on its features. The model works by constructing multiple decision trees on randomly selected subsets of the data and feature, then combines the predictions of each tree to make final prediction of which class the image belongs to. The model has several parameters such as the

number of trees, the maximum depth, and the minimum sample per leaf which affects its performance.

### Model 3: CNN

The CNN model uses a deep learning algorithm that learns from data which is useful for classification tasks. The purpose of the CNN model is to predict the class label of a given input image by extracting features and automatically learning from it. The model applies convolutional and pooling layers to the input and reduces the size to pull out useful features. The layers are then connected to give the final prediction of the image class. The hyperparameters including the six layers including the flattened and dense layers, affect the ultimate performance and accuracy of the model.

## Performance Metrics

To evaluate the models, we used several metrics to understand their performance. We calculated the accuracy levels for each model as well as used a confusion matrix and ROC curve to understand the performance of the Random Forest model.

Accuracy is commonly used to evaluate the performance of a model as it measures the proportion of correct predictions made by the model from all the predictions made. It is the ratio of the number of correct predictions to the total number of predictions. For this image classification problem, the accuracy levels tell us how well the models were able to predict the class of a given image. The goal is to achieve high accuracy as this means the model is doing a good job at predicting the classes of a large portion of the images. Although accuracy alone is not always reflective of a model's overall performance, it is a very useful metric that we can use to understand and compare the three model's performance.

The confusion matrix is a table of the different outcomes of the prediction and results of a classification problem that can aid in visualizing its predicted outcomes versus its actual outcomes. It shows the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) for each of the classes. With the given confusion matrix, we can calculate accuracy, precision, recall, and F1 score to measure the model's performance.

The Receiver Operating Characteristic (ROC) curve is a graph that shows the performance of a classification model at all the classification thresholds by comparing the true positive rate against the false positive rate. We are specifically looking for a high area under the curve (AUC) as this indicates higher model performance. In this case, the ROC curve can be used to evaluate the performance of the Random Forest model at different classification thresholds to determine the optimal threshold that maximizes the model's performance (see Figure 7).

## Results Table

| Models | Accuracy |
|---|---|
| Logistic Regression | 35.09% |
| Random Forest | 42.86% |
| CNN | 81.84% |

## Interpretation of the Results

The results of the three models show that the CNN model performed the best, achieving an accuracy of 81.84% (see Figure 10). The logistic regression model achieved an accuracy of 35.09% (see Figure 11) and the random forest model achieved an accuracy of 42.86% (see Figure 3).

The CNN model's superior performance can be attributed to its ability to capture spatial dependencies in images. It has multiple convolutional and pooling layers, which can detect and extract various features in the images, making it more effective in Image Classification tasks.

On the other hand, the logistic regression model is a simpler model that only uses a single dense layer. It is not as effective as the CNN model in capturing the spatial dependencies and extracting features from images. Therefore, its accuracy is lower than that of the CNN model.

The random forest model performed better than the logistic regression model but not as good as the CNN model. The random forest model is an ensemble of decision trees that can capture complex relationships between features. However, it still falls short compared to the CNN model's ability to capture spatial dependencies and extract features from the images. As we examined the classification report (see Figure 3), we saw that the precision levels were moderate ranging from 0.34-0.52 meaning that it may be incorrectly identifying many cases as positive, when they are not.

From the ROC-AUC curve (see Figure 7) and the confusion matrix heatmap (see Figure 8), we were able to see that classes 2 and 3 that are Bird and Cat were misclassified the most as they have the lowest value of all. Also, classes 1 and 8, which are Automobile and Ship are classified most accurately in comparison to the other classes.

One potential source of error in this analysis is overfitting. The models could have overfitted to the training data, causing them to perform poorly on the test data. This was avoided by using the dropout layer, setting the dropout normalization to 20% in our CNN model, we were able to see difference in our test accuracy and decrease in the entropy loss (see Figure 10).

## Sensitivity Analysis

To test the robustness of our model, we conducted a sensitivity analysis to gain a better understanding of the robustness and limitations of each model and identify potential areas for improvement. For our CNN model, we tried to train our model with 100 epochs. Doing so we got a test accuracy of approximately 87%. But the time taken to train the model was 1 hour.

Whereas, training the model with 24 epochs gave us 82% accuracy in 20 mins of training. Hence, giving an immense time difference of almost one fifth.

## Conclusion

After conducting the analysis, we found that the CNN performed the best, achieving the highest accuracy than compared to logistic regression and random forest. This is not surprising, as CNNs are designed specifically for image classification tasks and are able to learn more complex features than logistic regression and random forest models.

We also conducted a sensitivity analysis, varying the parameters and features used in the models. We found that changing the number of epochs and the number of neurons in the dense layers of the CNN had a significant impact on its performance. We could have achieved even higher accuracy by running it for higher epochs but choosing it to be 24 gave us a high accuracy in less time.

The results of this analysis have important implications for the use of machine learning models in image classification tasks. While logistic regression and random forest models can be useful for simpler classification tasks, more complex tasks such as image classification are better suited for CNN. However, it is important to note that our models were trained on a relatively small dataset and may not generalize well to other datasets or real-world applications. Future research could involve training these models on larger datasets like CIFAR-100, which consists of 100 classes of images with higher number of images in each class as well. We would also like to fine-tune the CNN architecture to further improve its performance. Additionally, other deep learning models such as recurrent neural networks could be explored for image classification tasks.

# List of Figures

Figure 1: CIFAR-10 Dataset

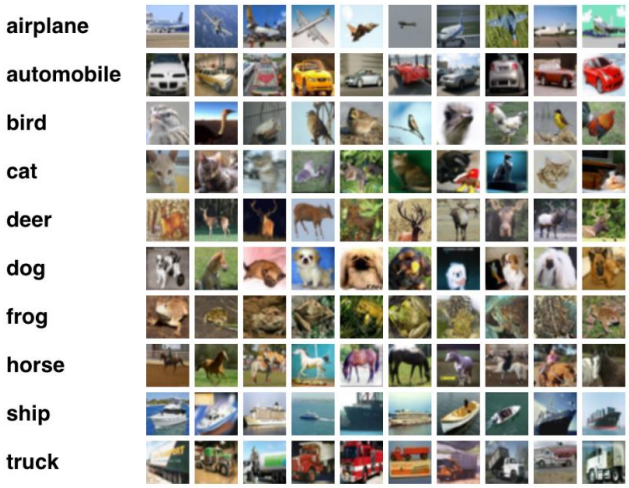



Figure 2: Random Forest Best Parameters

```
'criterion': 'gini',
'max_depth': 10,
'max_features': 'auto',
'min_samples_leaf': 10
```

Figure 3: Random Forest Classification Report

Classification Report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.51 | 0.52 | 0.51 | 1000 |
| 1 | 0.50 | 0.52 | 0.51 | 1000 |
| 2 | 0.38 | 0.18 | 0.24 | 1000 |
| 3 | 0.34 | 0.17 | 0.23 | 1000 |
| 4 | 0.34 | 0.42 | 0.37 | 1000 |
| 5 | 0.38 | 0.36 | 0.37 | 1000 |
| 6 | 0.39 | 0.56 | 0.46 | 1000 |
| 7 | 0.45 | 0.42 | 0.43 | 1000 |
| 8 | 0.52 | 0.58 | 0.55 | 1000 |
| 9 | 0.44 | 0.55 | 0.49 | 1000 |
| accuracy | | | 0.43 | 10000 |
| macro avg | 0.42 | 0.43 | 0.42 | 10000 |
| weighted avg | 0.42 | 0.43 | 0.42 | 10000 |

Figure 4: Logistic Regression training

```
def logistic_regression_model():
    # make a sequential model
    model = models.Sequential()
    # adding dense layer
    model.add(layers.Dense(units=10, activation='softmax', input_shape=(3072,)))
    # compiling the model
    model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])
    return model
```

Figure 5: CNN optimized model

```
def cnn_model():
    # make a sequential model
    model = models.Sequential()
    # add the convolution layer, and then add the pooling
    model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
    model.add(layers.MaxPooling2D((2, 2)))
    # dropout normalization rate is 20%
    model.add(layers.Dropout(0.2))
    # repeat adding convolution, pooling, and dropout 2 more times
    model.add(layers.Conv2D(64, (3, 3), activation='relu'))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Dropout(0.2))
    model.add(layers.Conv2D(64, (3, 3), activation='relu'))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Dropout(0.2))
    # flatten
    model.add(layers.Flatten())
    # add 2 more dense layers on top
    model.add(layers.Dense(64, activation='relu'))
    model.add(layers.Dense(10))
    # print the summary of the model
    model.summary()
    return model
```

Figure 6: Random Forest grid search parameters

```
def random_forest_model():
    # building a random forest model
    rf = RandomForestClassifier(random_state=42)
    # providing tuning parameters
    param_grid = {
        'n_estimators': [75, 100, 150, 200],
        'criterion': ['gini', 'entropy'],
        'max_depth': [5, 10, 100, 150, None],
        'min_samples_leaf': [1, 2, 5, 10],
        'bootstrap': [True, False]
        }

    # using grid search to find the best tuning paramets
    GridSearchCV(estimator=rf, param_grid=param_grid, cv=5)
```

Figure 7: Random Forest ROC Curve

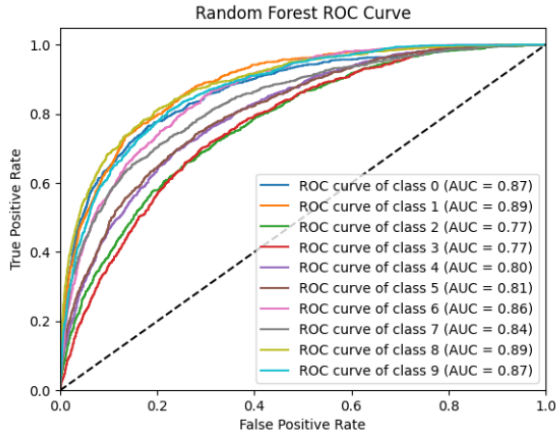



Figure 8: Confusion Matrix Heatmap

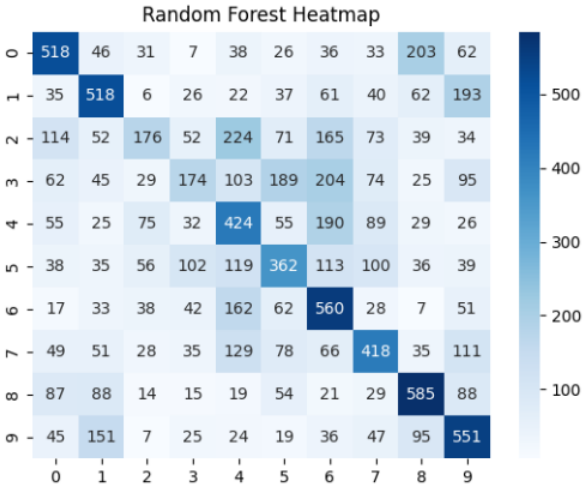



Figure 9: CNN predictions without Dropout

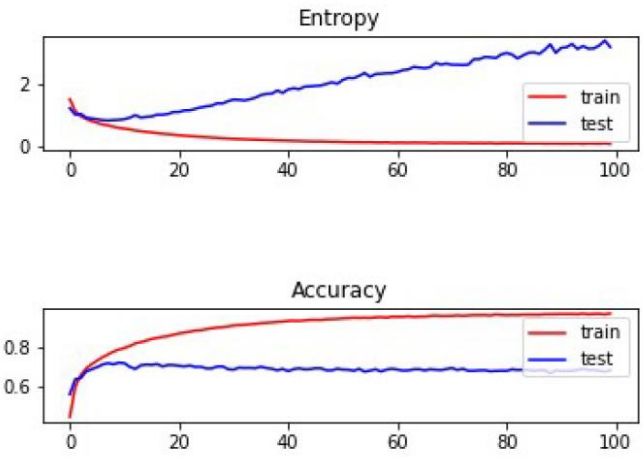


Figure 10: CNN predictions with Dropout

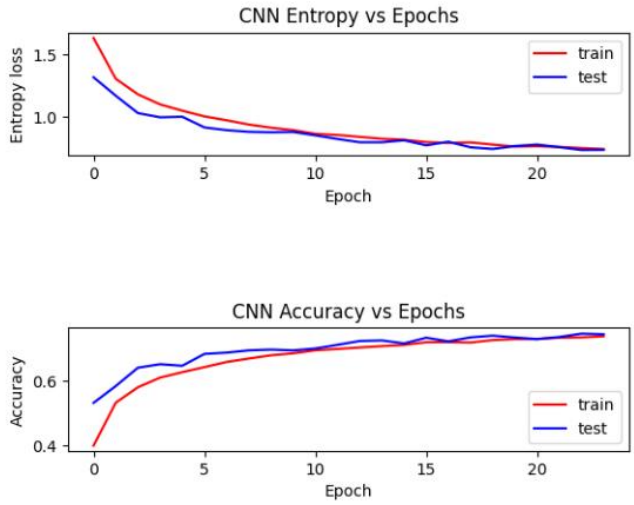
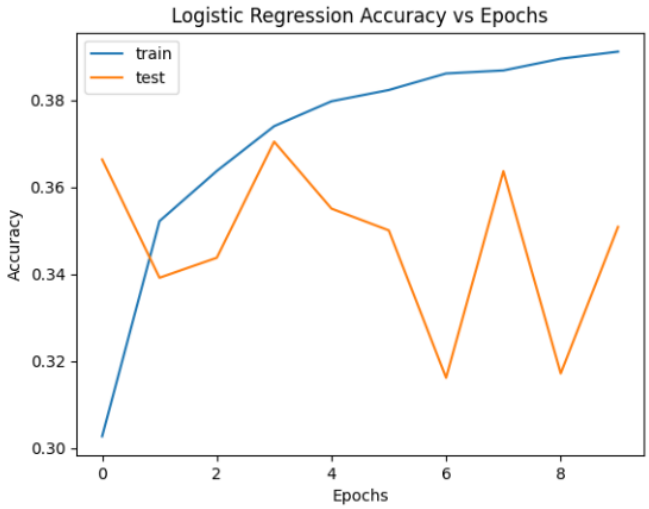



Figure 11: Logistic Regression accuracy plot



# References

[1] Boesch, G. (2023, February 24). *A complete guide to Image Classification in 2023*. viso.ai., from https://viso.ai/computer-vision/image-classification/

[2] CIFAR-10 and CIFAR-100 datasets. (n.d.)., from https://www.cs.toronto.edu/~kriz/cifar.html

[3] Danielsen, N. (2019, November 26). *Simple Image Classification with resnet 50*. Medium., from https://medium.com/@nina95dan/simple-image-classification-with-resnet-50-334366e7311a

[4] *VGG Image Classification (Vic) engine*. Visual Geometry Group - University of Oxford. (n.d.)., from https://www.robots.ox.ac.uk/~vgg/software/vic/