Car – Motorcycle
Logistic regression , Logistic regression with neural network and Cnn models – Final project

Yair Turgeman
Amit Rovshitz
Shoval Zohar

Abstract

This project presents a deep learning approach for motorcycle and car image classification using logistic regression, logistic regression with a neural network, and a Convolutional Neural Network (CNN). In simple logistic regression we got an accuracy of 0.759 in logistic regression with a neural network with a learning rate of 0.001 we got 0.848 accuracy, in logistic regression with a neural network with a learning rate of 0.0001 we got accuracy of 0.853, to our final CNN model without dropout, this advanced approach improved Significantly improved the performance of our model, achieving an accuracy of 0.901, and the second one we performed with dropout we got an accuracy of 0.962. We made all the images a uniform size of 224x224 pixels using the image resizing function, and we also normalized the images to maintain uniformity by maintaining consistent dimensions of the input data. Our findings highlight the effectiveness of combining dropout and data augmentation in complex neural network models for image classification, demonstrating significant improvements in model accuracy and robustness. In addition, show the differences between different learning rates.

1 Introduction

This project focuses on the classification of motorcycles and cars, using a dataset of RGB images. Our goal is to develop an advanced deep learning model capable of accurately classify these images, with broader implications for gesture recognition and complex image recognition applications. We started with a basic regression model, progressed to a combined neural network with a logistic regression model where we performed one with a learning rate of 0.001 and the other with a learning rate of 0.0001, and finally implemented a Convolutional Neural Network (CNN). The CNN was further improved with dropout. The strategy significantly improved the performance of our model, as evidenced by a leap in classification accuracy shown in Table 1. Detail This report presents and discusses the methodologies, results and learning key. This contributes to a broader understanding of the application of advanced neural network techniques in image classification, especially to illuminate the effectiveness of data dropout and augmentation in model performence.

Table 1: Model Performance Across Different Parameters

	Epochs	Learning rate	Drop out	Final Accuracy%
logistic regression	100	0.001	X	0.759
logistic regression + neural network	100	0.001	X	0.848
logistic regression + neural network	100	0.0001	X	0.853
Convolutional neural network	100	0.001	X	0.901
Convolutional neural network with drop out	100	0.001	v	0.962

2 Required Background

A basic understanding of neural networks, especially convolutional neural networks (CNNs), is essential to understanding the methodologies of our project. CNNs are known for their effectiveness in analyzing visual images, using convolution operations and layer structures to process input images. In addition, the logistic regression function is an integral part of our model, and enables the classification of images into one of two categories. This function is good for binary classification tasks. This model serves as our baseline for comparison with more complex models.

The logistic regression model uses a logistic function (also known as a sigmoid function) to model the relationship between the independent variables (or attributes) and the probability of belonging to a particular class. The logistic function has an S-shaped curve that maps any real-valued number to a value between 0 and 1, making it suitable for probabilistic modeling and then used to make binary classification predictions like ours. Moreover, a thorough knowledge of deep learning, a subset of machine learning in artificial intelligence, is essential. Deep learning uses layered structures of algorithms, called neural networks, to process data in complex ways, learning to perform tasks by examining examples. This approach is particularly effective in identifying patterns and making smart decisions based on large amounts of data. Our project leverages these deep learning principles, particularly in neural network architectures and algorithms, to advance the capabilities of our image classification model.

3 Project Description

3.1 Model Architecture

Our project uses the Logistic Regression function for binary classification, Logistic Regression with neural network and a Convolutional Neural Network (CNN) for classifying images one created with DropOut and the other without, implemented using TensorFlow. Logistic Regression model [see code in Figure 1] which includes input features, a target variable, a sigmoid function to model the relationship between features and probabilities, a hypothesis function to predict probabilities, a cost function to measure the performance of the model, an optimization algorithm to find optimal coefficients, and alternatively, regulation techniques to prevent overfitting.

A Logistic Regression model with neural network with a learning rate of 0.001[see code in Figure 2] which includes dense layers with ReLu operating functions, optimization with a specific learning rate, a loss function and metrics to evaluate the model's performance during training and testing.A Logistic Regression model with neural network with a learning rate of 0.0001[see code in Figure 3] which includes dense layers with ReLu operating functions, optimization with a specific learning rate, a loss function and metrics to evaluate the model's performance during training and testing.A Convolutional Neural Network (CNN) model without DropOut[see code in Figure 4] which includes five Convolutional Layers (Conv2D) with 8,16,32,64,224 filters, with kernel size (3x3), 'Same' padding is used to maintain the spatial dimensions of the input image , and ReLU activation is applied to the output feature map generated by the convolution operation.

In addition there are 5 more layers of MaxPooling Layers (MaxPooling2D), which includes pool size (2x2).In addition, one layer of Flatten and also 7 layers of Dense Layers (Fully

Connected Layers) with 8,16,32,64,128,200 units with ReLU activation and another layer of 2 units with softmax activation (output layer), this function converts the raw output scores into a probability distribution over the classes. With 2 neurons in the last layer, our model is designed for binary classification tasks, outputting probabilities for two classes. Softmax ensures that the sum of these probabilities is 1, making it ideal for multi-class classification. The predicted class is determined by the neuron with the highest probability
And a Convolutional Neural Network (CNN) model with DropOut[see code in Figure 5] which includes Dropout after specific max-pooling layers to introduce regularization and prevent overfitting. In this model, dropout layers with a dropout rate of 0.1 (meaning 10% of neurons are randomly deactivated) are added after the first, fourth, and fifth max-pooling layers. Dropout is applied after these max-pooling layers to introduce randomness and prevent the model from becoming overly dependent on specific features or neurons during training.

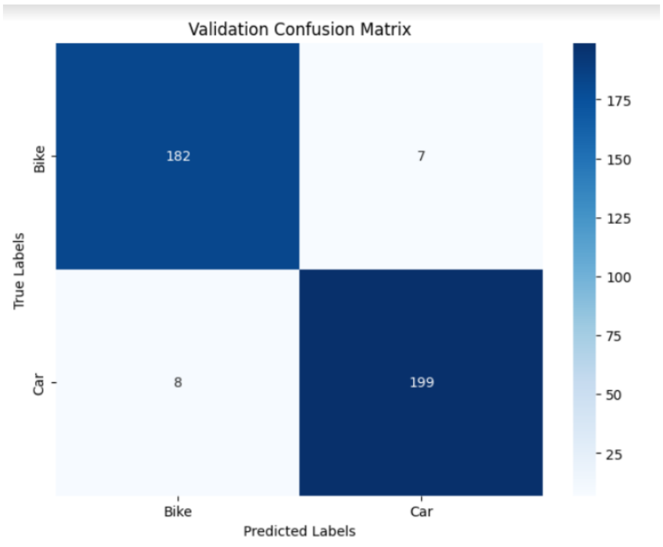
3.2 Preparation of data and training

The dataset consists of RGB images of cars and motorcycles. We processed all images to a common size of 224x224 pixels. We have defined a label that is associated with each image and indicates whether it belongs to the "car" class (label 1) or to the "motorcycles" class.(label 0). Finally, the images were loaded, resized, and normalized to ensure that the pixel values were between 1 and 1, using the normalize function.
The dataset is divided into training (70%), testing (10%) and validation (20%) groups.
The split is stratified based on the class labels to keep the same proportion of classes in each subset. This splitting ensures that the model is trained on a sufficient amount of data, evaluated against unseen data during validation, and finally tested on a separate unseen dataset to evaluate its real-world performance. The training set is used to train the model, the validation set is used to tune hyperparameters and prevent overfitting, and the test set is used to evaluate the performance of the final model. The training uses the crossentropy loss and the Adam Optimizer for 100 epochs.
Real-time performance metrics are available in [Figure 6] and the accuracy of the final model is calculated by running the test set shown in [Figure 7].

4 Results and Observations

The CNN Model with DropOut achieved an accuracy of 0. 962on the test set,
the CNN Model without DropOut achieved an accuracy of 0.901 in the test set, better performance than the previous models:
Logistic Regression - 0.759 accuracy.Logistic Regression with neural network and learning rate of 0.001 - 0.848 accuracy. Logistic Regression with neural network and learning rate of 0.0001 - 0.853, the accuracy level results of all the Logistic Regression model in [Figure 8].
In summary, the optimal performance was observed with 0.1 dropout rate. When the learning rate is higher, the level of accuracy also improves. These results illustrate the balance between model complexity and feature extraction in image classification.

5 Experiments/Simulation Results



5.1 Optimization of Dropout and Data Augmentation

In our experiments, we investigated the effect of dropout [see code in Figures 4,5] which is a technique used to prevent overfitting by randomly dropping units (neurons) during training, which helps to improve the generalization performance of the CNN model. We did experiments with a dropout rate of 0.1. This optimal dropout rate effectively prevented overfitting while maintaining the generalizability of the model.In addition, in the last layer of the model we used the softmax operating function.It is commonly used in classification tasks, where the network needs to output probabilities for multiple classes. The softmax function converts the raw output scores (logits) of the last layer to a probability distribution over the classes. In our code, the last Dense layer has 2 neurons (layers.Dense (2)), which suggests a binary classification task where the network outputs probabilities for two classes. The softmax operating function ensures that the sum of the output probabilities in all classes is equal to 1, making it suitable for multi-class classification problems. The network predicts the class corresponding to the neuron with the output The highest probability by the softmax layer.

5.2 Network Complexity and Training Parameters

We will examine the network complexity and training parameters for our CNN models, focusing first on the model without dropout regularization.Model Architecture A dropout CNN model consists of multiple convolutional layers followed by max-pooling and densely connected layers. The architecture is defined as follows: input shape:
(224, 224, 3)Convolutional Layer 1: 224 filters, kernel size (3x3), ReLU activation, padding='same'.MaxPooling Layer 1: Pool size (2x2).Convolutional Layer 2: 64 filters, kernel size (3x3), ReLU activation, padding='same'.MaxPooling Layer 2: Pool size (2x2).Convolutional Layer 3: 32 filters, kernel size (3x3), ReLU activation,

padding='same'.MaxPooling Layer 3: Pool size (2x2).Convolutional Layer 4: 16 filters, kernel size (3x3), ReLU activation, padding='same'.MaxPooling Layer 4: Pool size (2x2).Convolutional Layer 5: 8 filters, kernel size (3x3), ReLU activation, padding='same'.MaxPooling Layer 5: Pool size (2x2).Flatten Layer: Flatten the output of the previous layer.Dense Layers: Multiple layers are densely connected with ReLU enabled.Output layer: 2 neurons with softmax activation for binary classification. We used the Adam optimization which is an optimization with default parameters, in the loss function: sparse categorical crossentropy, the model is trained for 100 epochs with a batch size of 32 using the training set. Validation data is used to monitor the performance of the model during training. Evaluation: The performance of the model is evaluated on the validation and testing datasets using measures such as accuracy, precision, recall and F1 score.[Figure 4].
The CNN model with dropout regularization follows a similar architecture to that without dropout, but dropout layers are added after certain maximum connection layers to avoid overfitting.

5.3 Testing the Model

We predicted on the model all the images in the test and compared this together with the corresponding labels of the images. In addition, we sampled 9 random images from the test to visually show predictions versus actual labels for each image[Figure 6]

6 previous attempts

6.1 Recap of Assignment 2

In Task 2, the initial phase of the project focused on classifying car and motorcycle images using simpler models. The dataset consisted of color images, taking into account computational efficiency and dimensionality reduction. We used a basic neural network architecture, (`logreg_model`) trained using the flat input data (`X_train_flat`) and the corresponding target labels (`y_train`). The reshaped data is used as the input layer for the logistic regression model, where each feature of the input data corresponds to a neuron in the input layer. Training over 100 periods. We then used a logistic regression architecture with one neural network with a learning rate of 0.001 and the other with a learning rate of 0.0001. The logistic regression function involving a neural network with a learning rate of 0.001 resulted in an accuracy of 0.848.
In contrast, the logistic regression function including a neural network with a learning rate of 0.0001 resulted in an accuracy of 0.853.
This was an improvement over a simpler logistic regression model that achieved an accuracy of 0.759.

6.2 Development of the project

The progress from Task 2 to our final project was a great journey towards refining the accuracy of our model in classifying motorcycles and cars. Initially, we moved from simpler models to a Convolutional Neural Network (CNN), a strategic move driven by our need to handle image data more efficiently. Known for its skill in feature extraction and learning from spatial hierarchies, the CNN architecture initially boosted our accuracy to a 0.901 rating. However, we didn't stop there. Recognizing the potential for further improvement, we incorporated shedding techniques. This addition was critical in our effort to combat overfitting and improve the generalizability of the model from the training data. The combination of dropout, led us to an exceptional accuracy of 0. 962. This phase of our project was not only about moving to a more sophisticated model but also about carefully tuning it with advanced techniques to achieve optimal performance.

7 Conclusions

7.1 Summary of Findings

In this project, we learned a lot about how to improve a computer model for identification between a car and a motorcycle. Our best model, CNN, achieved an accuracy of 0. 962in 100 training rounds. It was much better than our first simple models, which received only 0.759, 0.848 and 0.853 accuracy. The big improvement came from use CNN and adding special techniques like dropout to make our model better at handling different types of images.

7.2 Future Work

There is more to explore in the future. We can try using our CNN model on more powerful computers to see if it performs even better. Another goal is to reach an accuracy of over 93%. To do this, we want to experiment with more ways to change the data, and see the differences. These changes can help our model to be even more accurate and reliable.

7.3 Final thoughts

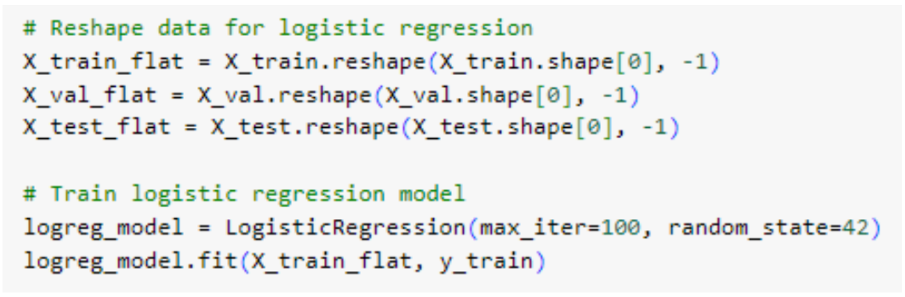
In this project we saw the difference between simple deep learning networks and more advanced CNNs. It was interesting to see how switching to CNN significantly increased the performance of our model. In addition, incorporating techniques such as dropout and changing the learning rate have proven to be game changers. These strategies not only improved the accuracy of our model but also provided important insights into the importance of adapting and refining our approach. It is clear that in the field of computer science, especially in areas such as image recognition, continuous learning and experimentation are the keys to success.

7.2 References

[1] Dataset on Kaggle: fromhttps://www.kaggle.com/datasets/utkarshsaxenadn/car-vs-bike-classification-dataset

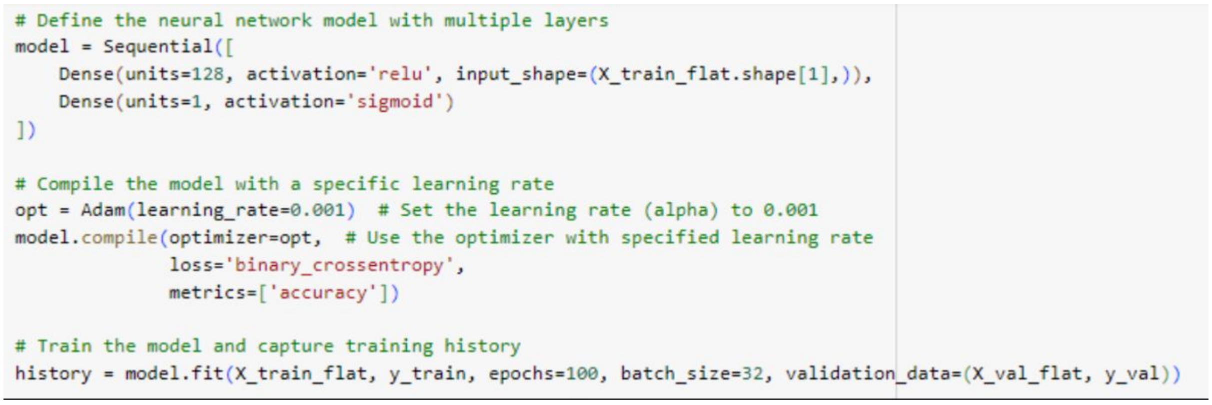
8 Code & Figure Appendix

Figure 1 : Logistic Regression code



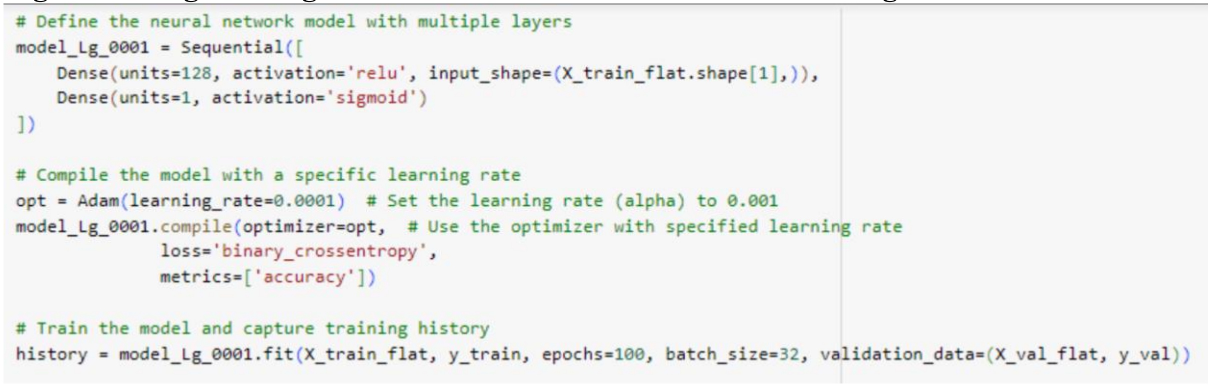
```
# Reshape data for logistic regression
X_train_flat = X_train.reshape(X_train.shape[0], -1)
X_val_flat = X_val.reshape(X_val.shape[0], -1)
X_test_flat = X_test.reshape(X_test.shape[0], -1)

# Train logistic regression model
logreg_model = LogisticRegression(max_iter=100, random_state=42)
logreg_model.fit(X_train_flat, y_train)
```

Figure 2 : Logistic Regression with neural networks and learning rate 0.001



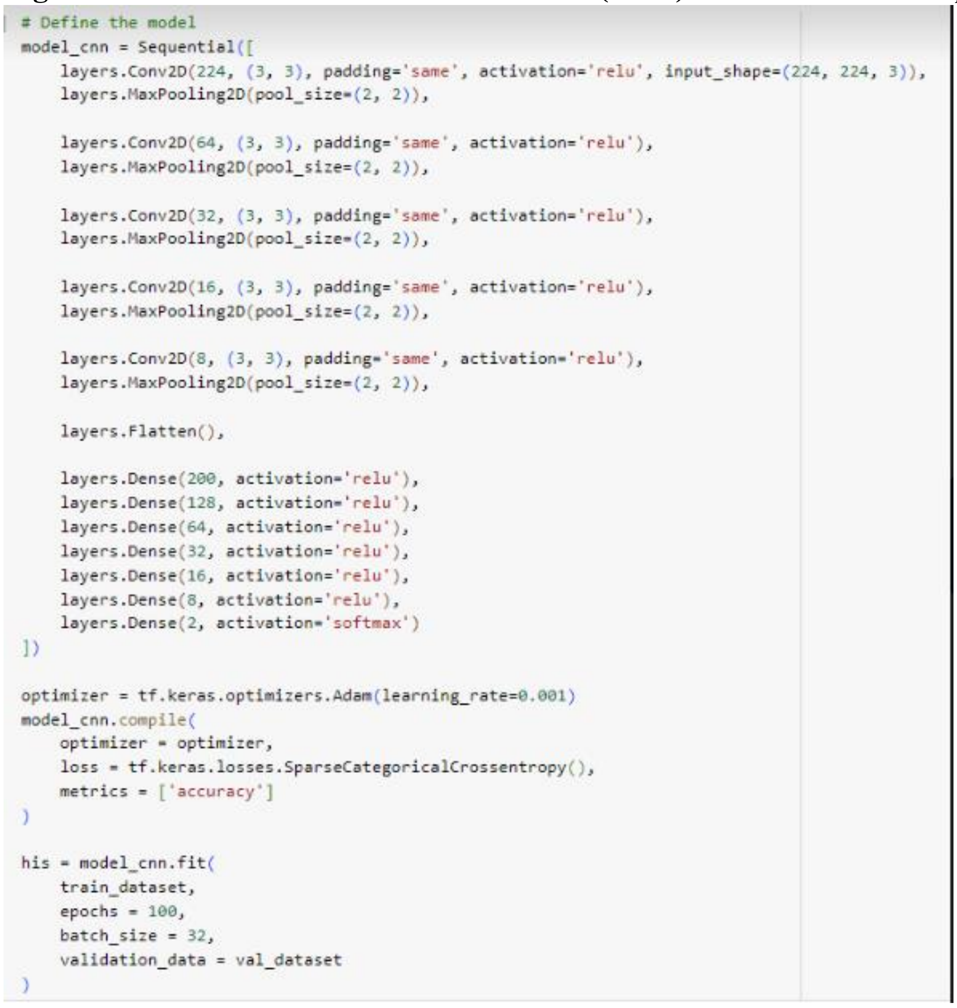
```
# Define the neural network model with multiple layers
model = Sequential([
    Dense(units=128, activation='relu', input_shape=(X_train_flat.shape[1],)),
    Dense(units=1, activation='sigmoid')
])

# Compile the model with a specific learning rate
opt = Adam(learning_rate=0.001)  # Set the learning rate (alpha) to 0.001
model.compile(optimizer=opt,  # Use the optimizer with specified learning rate
              loss='binary_crossentropy',
              metrics=['accuracy'])

# Train the model and capture training history
history = model.fit(X_train_flat, y_train, epochs=100, batch_size=32, validation_data=(X_val_flat, y_val))
```

Figure 3 : Logistic Regression with neural networks and learning rate 0.0001



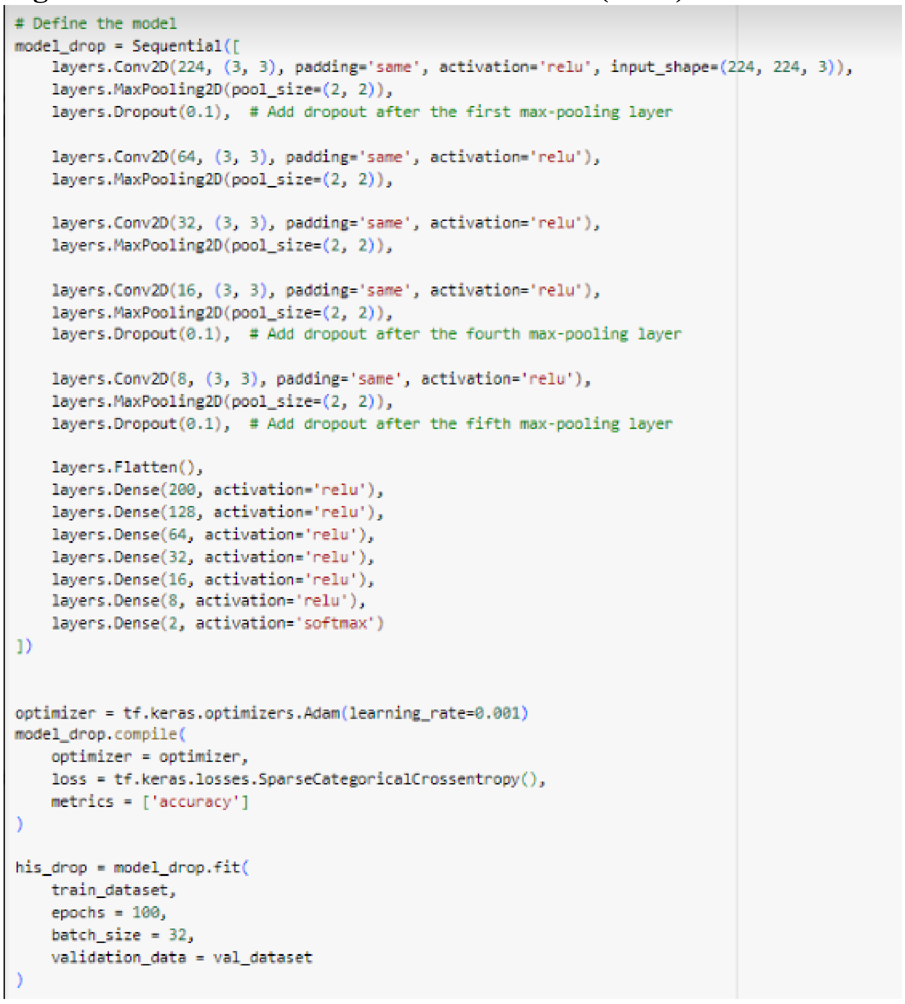
```
# Define the neural network model with multiple layers
model_Lg_0001 = Sequential([
    Dense(units=128, activation='relu', input_shape=(X_train_flat.shape[1],)),
    Dense(units=1, activation='sigmoid')
])

# Compile the model with a specific learning rate
opt = Adam(learning_rate=0.0001)  # Set the learning rate (alpha) to 0.001
model_Lg_0001.compile(optimizer=opt,  # Use the optimizer with specified learning rate
              loss='binary_crossentropy',
              metrics=['accuracy'])

# Train the model and capture training history
history = model_Lg_0001.fit(X_train_flat, y_train, epochs=100, batch_size=32, validation_data=(X_val_flat, y_val))
```

Figure 4: A Convolutional Neural Network (CNN) model without DropOut



```
# Define the model
model_cnn = Sequential([
    layers.Conv2D(224, (3, 3), padding='same', activation='relu', input_shape=(224, 224, 3)),
    layers.MaxPooling2D(pool_size=(2, 2)),

    layers.Conv2D(64, (3, 3), padding='same', activation='relu'),
    layers.MaxPooling2D(pool_size=(2, 2)),

    layers.Conv2D(32, (3, 3), padding='same', activation='relu'),
    layers.MaxPooling2D(pool_size=(2, 2)),

    layers.Conv2D(16, (3, 3), padding='same', activation='relu'),
    layers.MaxPooling2D(pool_size=(2, 2)),

    layers.Conv2D(8, (3, 3), padding='same', activation='relu'),
    layers.MaxPooling2D(pool_size=(2, 2)),

    layers.Flatten(),

    layers.Dense(200, activation='relu'),
    layers.Dense(128, activation='relu'),
    layers.Dense(64, activation='relu'),
    layers.Dense(32, activation='relu'),
    layers.Dense(16, activation='relu'),
    layers.Dense(8, activation='relu'),
    layers.Dense(2, activation='softmax')
])

optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)
model_cnn.compile(
    optimizer = optimizer,
    loss = tf.keras.losses.SparseCategoricalCrossentropy(),
    metrics = ['accuracy']
)

his = model_cnn.fit(
    train_dataset,
    epochs = 100,
    batch_size = 32,
    validation_data = val_dataset
)
```

Figure 5: A Convolutional Neural Network (CNN) model with DropOut



```
# Define the model
model_drop = Sequential([
    layers.Conv2D(224, (3, 3), padding='same', activation='relu', input_shape=(224, 224, 3)),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Dropout(0.1),  # Add dropout after the first max-pooling layer

    layers.Conv2D(64, (3, 3), padding='same', activation='relu'),
    layers.MaxPooling2D(pool_size=(2, 2)),

    layers.Conv2D(32, (3, 3), padding='same', activation='relu'),
    layers.MaxPooling2D(pool_size=(2, 2)),

    layers.Conv2D(16, (3, 3), padding='same', activation='relu'),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Dropout(0.1),  # Add dropout after the fourth max-pooling layer

    layers.Conv2D(8, (3, 3), padding='same', activation='relu'),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Dropout(0.1),  # Add dropout after the fifth max-pooling layer

    layers.Flatten(),
    layers.Dense(200, activation='relu'),
    layers.Dense(128, activation='relu'),
    layers.Dense(64, activation='relu'),
    layers.Dense(32, activation='relu'),
    layers.Dense(16, activation='relu'),
    layers.Dense(8, activation='relu'),
    layers.Dense(2, activation='softmax')
])


optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)
model_drop.compile(
    optimizer = optimizer,
    loss = tf.keras.losses.SparseCategoricalCrossentropy(),
    metrics = ['accuracy']
)

his_drop = model_drop.fit(
    train_dataset,
    epochs = 100,
    batch_size = 32,
    validation_data = val_dataset
)
```

Figure 6 : Real-time performance metrics

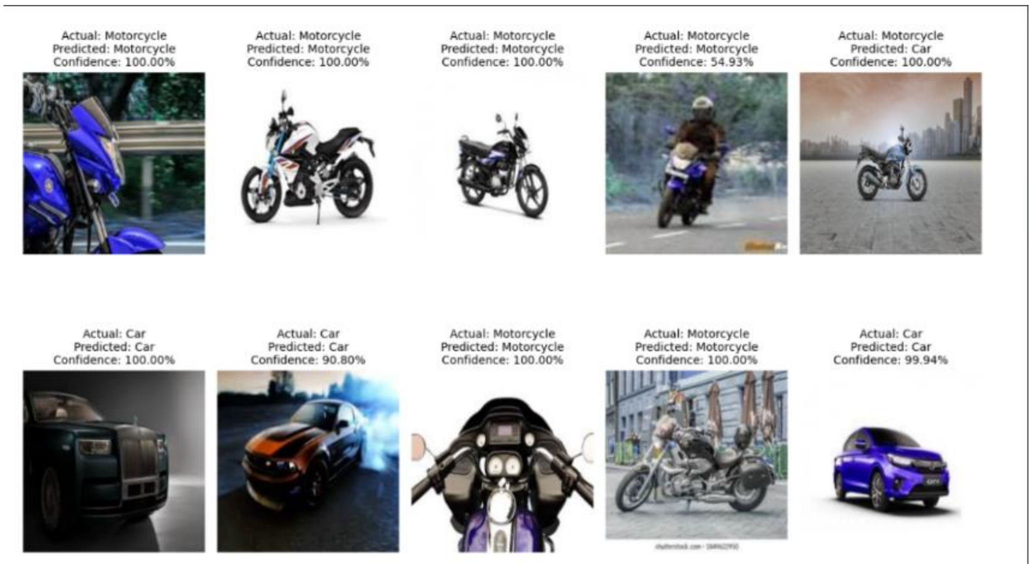


Figure 7 : Loss and accuracy of the model

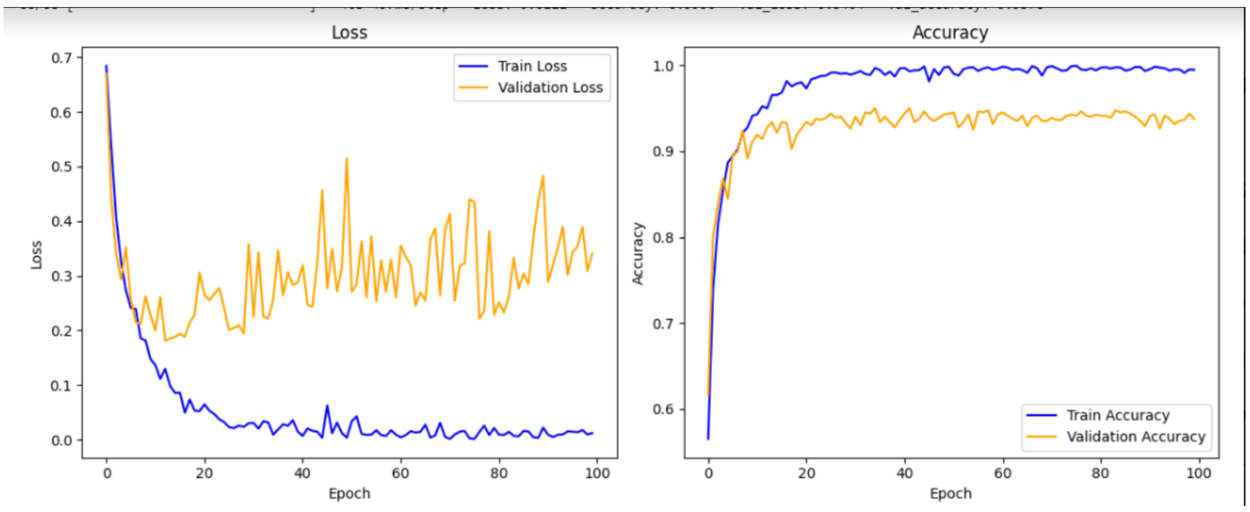
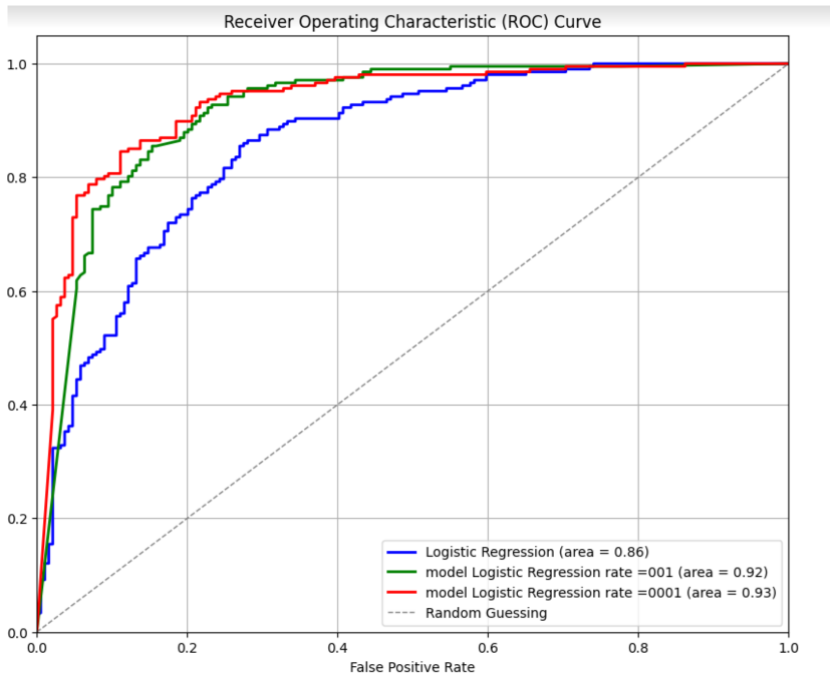


Figure 8 : The accuracy level results of all the Logistic Regression model


Receiver Operating Characteristic (ROC) Curve
Logistic Regression (area = 0.86)
model Logistic Regression rate =001 (area = 0.92)
model Logistic Regression rate =0001 (area = 0.93)
Random Guessing
False Positive Rate