

**DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING
BACHELORS IN COMPUTER SYSTEMS ENGINEERING**

Course Code: CS-324

Course Title: Machine Learning

Complex Engineering Problem

TE Batch 2022, Spring Semester 2025

Grading Rubric

TERM PROJECT

Group Members:

Student No.	Name	Roll No.
S1	Aqiba Abdul Qadir	CS22003
S2	Tooba Aftab	CS22020
S3	Bareera Ahsan	CS22030

CRITERIA AND SCALES					Marks Obtained		
					S1	S2	S3
Criterion 1: Does the application meet the desired specifications and produce the desired outputs? (CPA-1, CPA-2, CPA-3) [8 marks]							
1	2	3	4				
The application does not meet the desired specifications and is producing incorrect outputs.	The application partially meets the desired specifications and is producing incorrect or partially correct outputs.	The application meets the desired specifications but is producing incorrect or partially correct outputs.	The application meets all the desired specifications and is producing correct outputs.				
Criterion 2: How well is the code organization? [2 marks]							
1	2	3	4				
The code is poorly organized and very difficult to read.	The code is readable only to someone who knows what it is supposed to be doing.	Some part of the code is well organized, while some part is difficult to follow.	The code is well organized and very easy to follow.				
Criterion 3: Does the report adhere to the given format and requirements? [6 marks]							
1	2	3	4				
The report does not contain the required information and is formatted poorly.	The report contains the required information only partially but is formatted well.	The report contains all the required information but is formatted poorly.	The report contains all the required information and completely adheres to the given format.				
Criterion 4: How does the student performed individually and as a team member? (CPA-1, CPA-2, CPA-3) [4 marks]							
1	2	3	4				
The student did not work on the assigned task.	The student worked on the assigned task, and accomplished goals partially.	The student worked on the assigned task, and accomplished goals satisfactorily.	The student worked on the assigned task, and accomplished goals beyond expectations.				

Final Score = (Criteria_1_score) + (Criteria_2_score) + (Criteria_3_score) + (Criteria_4_score)

= _____

Teacher's Signature

CIFAR-10 IMAGE CLASSIFICATION MODELS

PREPROCESSING OF IMAGES:

1. Converting Multiclass to Binary Labels

- **Purpose:** Convert the multiclass (10-class) problem to a binary classification.
 - **Binary Labeling Criteria:**
 - Labels **0, 1, 8, 9** → **Vehicle** → **1**
 - All others → **Animal** → **0**
-

2. Grayscale Conversion

- **Function Used:** `rgb2gray()`
 - **Formula:** $\text{gray} = 0.2989 * R + 0.5870 * G + 0.1140 * B$
 - **Purpose:** Reduce dimensionality by converting 3-channel (RGB) images to single-channel grayscale images.
 - **Resulting Shape:**
 - Train: (50000, 32, 32)
 - Test: (10000, 32, 32)
-

2. Flattening

- **Purpose:** Convert 3D images into 1D vectors for ML models.
 - **Original RGB Flattened Shapes:**
 - Train: (50000, 3072) $\leftarrow 32 \times 32 \times 3$
 - Test: (10000, 3072)
 - **Grayscale Flattened Shapes:**
 - Train: (50000, 1024) $\leftarrow 32 \times 32$
 - Test: (10000, 1024)
-

3. Scaling (Normalization)

- **Method:** Divide all pixel values by 255 to bring them in range [0, 1].
 - **Applied To:**
 - RGB Flattened images
 - Grayscale Flattened images
-

4. PCA (Principal Component Analysis)

- **Goal:** Dimensionality Reduction (retain 95% variance)

- **Grayscale Flattened Data:**
 - Reduced dimensions: **160**
 - **RGB Flattened Data:**
 - Also reduced to: **160**
 - **Effect:**
 - **Train Reduced Shape:** (50000, 160)
 - **Train Recovered Shape:** (50000, 1024)
 - **Test Reduced Shape:** (10000, 160)
 - **Test Recovered Shape:** (10000, 1024)
 - **Observation:** Visual inspection of recovered images showed considerable feature loss, typical after high-dimensional compression.
-

5. Dataset Re-splitting

- **Original Split:** Train: 50,000 | Test: 10,000
- **New Split:**
 - Train: **40,000**
 - Test: **20,000** (10,000 original test + 10,000 from tail of training set)
- **Shapes After Resplitting and Scaling:**
 - New Train: (40000, 3072)
 - New Test: (20000, 3072)

MODEL TRAINING:

ARTIFICIAL NEURAL NETWORK:

Model 1a: ANN WITH TWO HIDDEN LAYERS & RMSPROP OPTIMIZER:

Training Overview

Layer	Description
Flatten	Converts input from (32, 32, 3) to 1D vector (3072,)
Dense(300)	First hidden layer with 300 neurons and ReLU activation
BatchNormalization()	Normalizes activations to speed up training and reduce internal covariate shift
Dense(100)	Second hidden layer with 100 neurons and ReLU activation
Dense(1, activation='sigmoid')	Output layer for binary classification (returns probability)

- **Model Summary Command:** model1a.summary()
- **Total Parameters:** 953,301 (3.64 MB)
 - **Trainable Parameters:** 952,701
 - **Non-trainable Parameters:** 1,200
- **Optimizer Used:** RMSprop
- **Loss Function:** Binary Crossentropy
- **Metrics Tracked:** F1-Score, Accuracy, Recall, Precision

Training Configuration

- **Training Set:** 50,000 samples scaled
- **Validation Set:** 10,000 samples scaled
- **Epochs:** 20
- **Batch Size:** 32
- **Callback:** ModelCheckpoint – saves best model based on validation accuracy

Training Performance Highlights

Metric	Training	Validation
Accuracy	90.76%	81.48%
Loss	0.2308	0.8881
Precision	89.31%	70.81%
Recall	87.15%	91.37%

Observation: The model shows good training performance but somewhat lower generalization on the validation set, with increasing validation loss after several epochs – suggesting **mild overfitting**.

Model 1a– Testing Performance Review

Metric	Value	Interpretation
Accuracy	85.76%	The model correctly classified about 86 out of 100 test images. This is a solid overall performance, indicating good generalization to unseen data.
F1 Score	83.01%	This harmonic mean of precision and recall shows balanced performance . A value above 80% reflects a strong ability to handle both false positives and false negatives.

Model 1b: ANN WITH FIVE HIDDEN LAYERS & ADAM OPTIMIZER:

Model Architecture

Layer	Description
Flatten	Converts input from (32, 32, 3) to a 1D vector (3072,)
Dense(1024)	First hidden layer with 1024 neurons and ReLU activation
BatchNormalization()	Normalizes activations to stabilize and accelerate training
Dropout(0.5)	A drop out layer with 0.5 drop out probability
Dense(512)	Second hidden layer with 512 neurons and ReLU activation
BatchNormalization()	Normalizes activations to stabilize and accelerate training
Dropout(0.4)	A drop out layer with 0.4 drop out probability
Dense(256)	Third hidden layer with 256 neurons and ReLU activation
BatchNormalization()	Normalizes activations to stabilize and accelerate training
Dropout(0.3)	A drop out layer with 0.3 drop out probability
Dense(128)	Fourth hidden layer with 128 neurons and ReLU activation
BatchNormalization()	Normalizes activations to stabilize and accelerate training
Dropout(0.2)	A drop out layer with 0.2 drop out probability
Dense(64)	Fifth hidden layer with 64 neurons and ReLU activation
BatchNormalization()	Normalizes activations to stabilize and accelerate training
Dropout(0.1)	A drop out layer with 0.1 drop out probability
Dense(1, activation='sigmoid')	Output layer for binary classification (returns probability between 0 and 1)

- **Model Summary Command:** model1b.summary()
 - **Total Parameters:** 3,852,033 (14.69 MB)
 - **Trainable Parameters:** 3,848,065
 - **Non-trainable Parameters:** 3,968
 - **Optimizer Used:** Adam (learning rate = 0.008)
 - **Loss Function:** Binary Crossentropy
 - **Metrics Tracked:** F1-Score, Accuracy, Recall, Precision
-

Training Configuration

- **Training Set:** 50,000 samples (unscaled)
- **Validation Set:** 10,000 samples (unscaled)
- **Epochs:** 50
- **Batch Size:** 128
- **Callback Used:** ModelCheckpoint – saves the best model based on **validation loss**

Training Performance Highlights (Final Epoch)

<i>Metric</i>	<i>Training</i>	<i>Validation</i>
<i>Accuracy</i>	88.87%	87.44%
<i>Loss</i>	0.3276	0.3460
<i>Precision</i>	86.7%	82.45%
<i>Recall</i>	85.33%	87.15%

Observation

The model demonstrates **strong training performance**, with:

- High training accuracy and recall
- **Validation Precsion plateauing around 81–84%**, despite longer training (50 epochs)

Model 1b – Testing Performance Review

Metric	Value	Interpretation
Accuracy	88.01%	The model correctly predicted about 88 out of 100 test samples. While slightly lower than Model 1a, this still indicates strong general performance .
F1 Score	84.94%	The model achieves good balance between precision and recall . This is important in a binary classification where both false positives and false negatives matter.

Model 1c: With Different Neurons In Hidden Layer & RMSProp Optimizer:

Model Architecture

Layer	Description
Flatten	Converts input from (32, 32, 3) to a 1D vector (3072,)
Dense(400)	First hidden layer with 400 neurons and ReLU activation
BatchNormalization()	Normalizes activations to stabilize and accelerate training
Dense(200)	Second hidden layer with 200 neurons and ReLU activation
BatchNormalization()	Normalizes second hidden layer
Dense(1, activation='sigmoid')	Output layer for binary classification (returns probability between 0 and 1)

- **Model Summary Command:** model1c.summary()
- **Total Parameters:** 1,312,001 (≈5.00 MB)
 - **Trainable Parameters:** 1,310,801
 - **Non-trainable Parameters:** 1,200
- **Optimizer Used:** RMSprop
- **Loss Function:** Binary Crossentropy
- **Metrics Tracked:** Accuracy, Precision, Recall

Training Configuration

Setting	Details
Training Set	50,000 samples(scaled)
Validation Set	10,000 samples(scaled)
Epochs	30
Batch Size	32
Callback Used	ModelCheckpoint (monitors validation accuracy and saves the best model)

Training Performance Highlights (Final Epoch)

Metric	Training	Validation
Accuracy	96.67%	79.11%
Loss	0.0863	4.2024
Precision	96.03%	69.17%
Recall	95.56%	86.20%

Observation

Model 1c exhibits **strong training performance** but also shows signs of **overfitting**, with the following insights:

- **Training accuracy and precision** are exceptionally high, suggesting the model fits the training data very well.
- **Validation loss is significantly higher** than training loss (4.2024 vs. 0.0863), highlighting potential overfitting despite batch normalization.
- **Validation accuracy** plateaus around **79–80%**, which is lower than in Model 1b and previous variant reports.

Model 1c – Testing Performance Review

Metric	Value	Interpretation
Accuracy	87.71%	Despite overfitting signs, test accuracy is strong, suggesting the model generalizes reasonably well on unseen data.
F1 Score	84.04%	Balanced performance between precision and recall. Slightly below Model 1b’s F1 score but competitive overall.

Model 1d: With Three Hidden Layers & Adam Optimizer

Model Architecture

Layer	Description
Flatten	Converts input from (32, 32, 3) to a 1D vector (3072,)
Dense(400)	First hidden layer with 400 neurons and ReLU activation
BatchNormalization()	Normalizes activations of the first hidden layer to stabilize training
Dense(200)	Second hidden layer with 200 neurons and ReLU activation
BatchNormalization()	Normalizes activations of the second hidden layer
Dense(100)	Third hidden layer with 100 neurons and ReLU activation
BatchNormalization()	Normalizes activations of the third hidden layer
Dense(1, activation='sigmoid')	Output layer for binary classification (returns probability between 0 and 1)

- **Model Summary Command:** model1d.summary()
- **Total Parameters:** 1,332,401 (≈5.08 MB)
 - **Trainable Parameters:** 1,331,001
 - **Non-trainable Parameters:** 1,400
- **Optimizer Used:** Adam
- **Loss Function:** Binary Crossentropy
- **Metrics Tracked:** Accuracy, Precision, Recall

Configuration

Setting	Details
Training Set	40,000 samples
Validation Set	20,000 samples
Epochs	40
Batch Size	64
Callback Used	ModelCheckpoint (monitors validation loss and saves best model)

Training Performance Highlights (Final Epoch)

Metric	Training	Validation
Accuracy	99.33%	86.26%
Loss	0.0188	3.3286
Precision	99.17%	85.87%
Recall	99.13%	78.57%

Model 1d – Testing Performance Review

Metric	Value	Interpretation
Accuracy	86.07%	High and consistent with validation, showing solid generalization
F1 Score	82.51%	Balanced performance between precision and recall, higher than Model 1c's F1 score

Observations

- **Training accuracy (99.33%) is significantly higher** than both validation (86.26%) and test accuracy (86.07%), indicating the model has likely **overfit** the training data.
- Despite using **Batch Normalization and the Adam optimizer**, the model still **memorizes the training data**, as seen in the extremely low training loss (0.0188).
- **Validation loss (3.3286)** remains much higher than training loss, reinforcing the **overfitting behavior** after extended training (40 epochs).

Model 1e: Regularized Version of Model 1d with L2 Penalty & RMSProp Optimizer

Model Architecture

Layer	Description
Flatten	Converts input from (32, 32, 3) to a 1D vector (3072,)
Dense(400)	First hidden layer with 400 neurons and ReLU activation
BatchNormalization()	Normalizes activations of the first hidden layer
Dense(200, L2=0.05)	Second hidden layer with L2 regularization and ReLU activation
BatchNormalization()	Normalizes activations of the second hidden layer
Dense(100, L2=0.02)	Third hidden layer with L2 regularization and ReLU activation
BatchNormalization()	Normalizes activations of the third hidden layer
Dense(1, activation='sigmoid')	Output layer for binary classification

- **Model Summary Command:** model1e.summary()
- **Total Parameters:** 1,332,401 (≈5.08 MB)
 - **Trainable Parameters:** 1,331,001
 - **Non-trainable Parameters:** 1,400
- **Optimizer Used:** RMSprop
- **Loss Function:** Binary Crossentropy
- **Metrics Tracked:** Accuracy, Precision, Recall

Training Configuration

Setting	Details
Training Set	50,000 samples
Validation Set	10,000 samples
Epochs	40
Batch Size	64
Callback Used	ModelCheckpoint (monitors validation accuracy and saves best model)

Training Performance Highlights (Final Epoch)

Metric	Training	Validation
Accuracy	89.91%	83.74%
Loss	0.2623	0.3948
Precision	87.53%	77.58%
Reall	86.97%	83.47%

Model 1e – Testing Performance Review

Metric	Value	Interpretation
Accuracy	86.51%	Slightly better than Model 1d; shows improved generalization
F1 Score	82.47%	Comparable to Model 1d (82.51%) — well-balanced precision and recall

Observations

- **Model 1e exhibits improved generalization** compared to earlier variants by effectively applying **L2 regularization** on Dense layers.
- The **training accuracy (89.91%)** is now much closer to **validation (83.74%)** and **testing (86.51%)** accuracy, indicating **reduced overfitting**.

Model 1f: Three Hidden layers, HE Kernel Initialiser and RMSProp Optimizer:

Model Architecture

Layer	Description
Flatten	Converts input from (32, 32, 3) to a 1D vector (3072,)
Dense(1024)	First hidden layer with 1024 neurons, ReLU activation, He normal initialization
BatchNormalization()	Normalizes activations of the first hidden layer
Dropout(0.5)	Drops 50% of units to reduce overfitting
Dense(512, L2=0.001)	Second hidden layer with 512 neurons, ReLU activation, L2 regularization
BatchNormalization()	Normalizes activations of the second hidden layer
Dropout(0.4)	Drops 40% of units
Dense(256, L2=0.002)	Third hidden layer with 256 neurons, ReLU activation, L2 regularization
BatchNormalization()	Normalizes activations of the third hidden layer
Dropout(0.3)	Drops 30% of units
Dense(1, sigmoid)	Output layer for binary classification

Model Summary

- **Model Summary Command:** `model1f.summary()`
 - **Total Parameters: 3,810,305** (≈14.54 MB)
 - Trainable Parameters: 3,806,721
 - Non-trainable Parameters: 3,584
-

Training Configuration:

Setting	Details
Training Set	50,000 samples(scaled)
Validation Set	10,000 samples(scaled)
Epochs	100
Batch Size	64
Loss Function	Binary Crossentropy
Optimizer	RMSPROP

Training Performance Highlights (Final Epoch)

Metric	Training	Validation
Accuracy	84.98%	85.44%
Loss	0.3692	0.3607
Precision	82.13%	89.16%
Recall	79.45%	72.40%

Model 1f – Testing Performance Review

Metric	Value	Interpretation
Accuracy	87.13%	Higher than validation, indicating decent generalization
F1 Score	83.44%	Significant improvement over validation, balanced performance

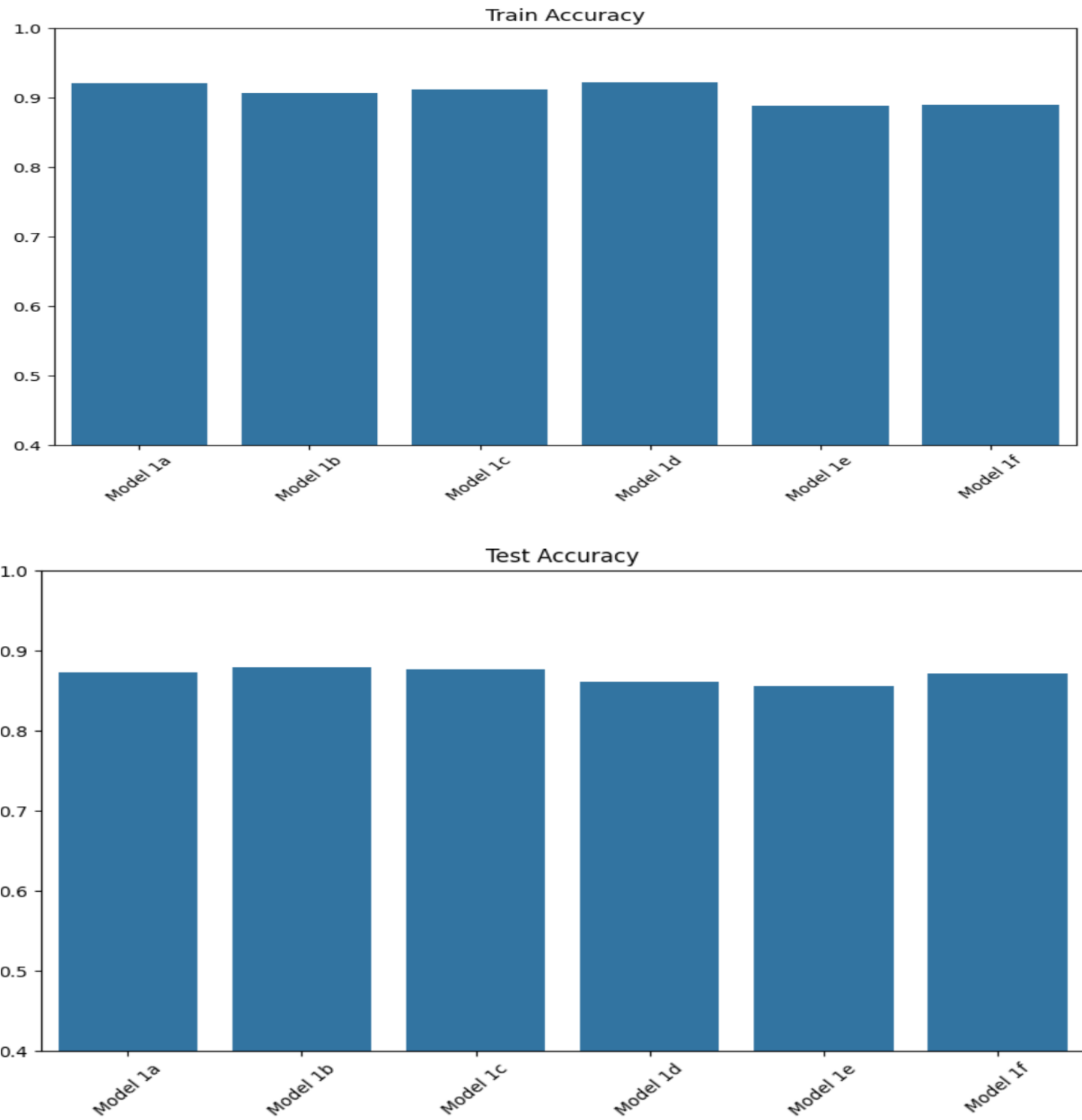
Observations

- Model 1f incorporates deeper layers and stronger regularization via both Dropout and L2 penalties.
- Achieves better test accuracy and F1 score than Model 1e, suggesting enhanced generalization.

COMPARING ALL THE MODELS OF ANN:

Model Name	Train Accuracy (%)	Train F1-Score (%)	Test Accuracy (%)	Test F1-Score (%)
Model 1a	92.07	89.82	87.25	83.47
Model 1b	90.72	88.36	88.01	84.94
Model 1c	91.14	88.55	87.71	84.04
Model 1d	92.18	90.17	86.07	82.51
Model 1e	88.83	85.36	85.64	81.09
Model 1f	89.04	85.92	87.13	83.44

GRAPHICAL REPRESENTATION:



FINAL COMMENT ON ANN:

Model 1b offers the best trade-off between learning capacity and generalization, making it the most effective model overall. If model simplicity and robustness are priorities, Model 1a and Model 1f are also strong contenders due to their controlled overfitting and stable test performance.

Model 2a: Shallow Parametric Model – Logistic Regression (Gray-scale Dataset)

Model Architecture

Component	Details
Model Type	Logistic Regression (Binary Classification)
Input Data	Gray-scale image data (flattened and scaled)
Regularization	Default (L2), solver-specific
Training Command	LogisticRegression(max_iter=1000)
Persistence	Saved using pickle as model2a.pkl
Prediction	model.predict() used on scaled test set

Testing Performance Review

Metric	False Class	True Class	Macro Avg	Weighted Avg
Precision	0.74	0.69	0.71	0.72
Recall	0.83	0.55	0.69	0.72
F1 Score	0.78	0.61	0.70	0.71
Support	6000	4000	—	10000

- **Overall Accuracy:** 72.09%
- **Test F1 Score:** 61.23% (specific to the *True* class)

Observations

- **Model 2a is a shallow linear model**, and its performance is **limited by its simplicity**, especially compared to deeper ANN models.
- The **overall accuracy is moderate (72.09%)**, but **class imbalance in recall** shows the model struggles with detecting the **positive (True) class**.
- The **recall for the True class is only 55%**, indicating many **false negatives**, while precision (69%) is acceptable.
- This model **performs well for the False class** but is clearly **biased toward the majority class** in terms of detection.
- The **F1 Score for the positive class (61.23%)** reflects poor balance and overall predictive power, particularly for tasks where both classes are equally important.

Summary & Recommendation

Model 2a (Logistic Regression) serves as a **baseline shallow model** for binary classification on gray-scale data. While it offers **fast training** and **interpretability**, its performance is **significantly lower** than the ANN models in both **accuracy and F1 score**. It is not suitable for high-dimensional data like images, even when flattened. For better generalization, especially in image tasks, a **ANNs** are strongly recommended.

Model 2b: Shallow Parametric Model – Logistic Regression (RGB Scaled Dataset)

Model Architecture

Component	Details
Model Type	Logistic Regression (Binary Classification)
Input Data	RGB image data (flattened and min-max scaled)
Training Command	LogisticRegression(max_iter=1000)
Persistence	Model saved using pickle as model2b.pkl
Prediction	model.predict() used on scaled RGB test data

Testing Performance Review

Metric	False Class	True Class	Macro Avg	Weighted Avg
Precision	0.83	0.79	0.81	0.82
Recall	0.87	0.74	0.80	0.82
F1 Score	0.85	0.76	0.81	0.82
Support	6000	4000	—	10000

Overall Accuracy: 81.67%

- Test F1 Score (for True class): 76.24%

Observations

- Model 2b significantly outperforms Model 2a, demonstrating that RGB data preserves more discriminatory features than gray-scale when used with a shallow model like logistic regression.
- The test accuracy (81.67%) is quite respectable for a linear model and not far behind more complex ANN models like Model 1a and 1b.

Model 2c: Shallow Parametric Model – Logistic Regression (PCA-Applied Dataset)

Model Architecture

Component	Details
Model Type	Logistic Regression (Binary Classification)
Input Data	PCA-reduced features (dimensionality reduced and then recovered)
Training Command	LogisticRegression(max_iter=1000)
Persistence	Model saved using pickle as model2c.pkl
Prediction	Performed on PCA-transformed and inverse-transformed test data (X_test_recovered)

Testing Performance Review

Metric	False Class	True Class	Macro Avg	Weighted Avg
Precision	0.74	0.70	0.72	0.72
Recall	0.84	0.55	0.70	0.73
F1 Score	0.79	0.62	0.70	0.72
Support	6000	4000	—	10000

- **Overall Accuracy:** 72.58%
- **Test F1 Score (for True class):** 61.76%

Observations

- **Model 2c’s performance is similar to Model 2a**, both achieving around **72% accuracy** and **~61–62% F1 score** on the True class.
- Although PCA reduces input dimensionality, the performance suggests **information loss during reconstruction**, especially affecting the model’s ability to identify the positive class (Recall = 55%).
- The model is **biased towards the majority (False) class**, achieving stronger performance in that category.

Summary & Recommendation

- Model 2c, using Logistic Regression on PCA-reconstructed input, shows **moderate performance** with **72.58% accuracy** and a **True-class F1 score of 61.76%**. It demonstrates that while PCA can reduce data size and training time, it may also degrade model performance when critical spatial or color information is lost.

Model 2d: Logistic Regression (Liblinear Solver) – RGB Scaled Dataset

Model Architecture

Component	Details
Model Type	Logistic Regression (Binary Classification)
Solver Used	liblinear (suited for small datasets and L1/L2 regularization)
Input Data	RGB image data (flattened and min-max scaled)
Training Command	LogisticRegression(solver='liblinear', random_state=0)
Prediction	model.predict() used on scaled RGB test data

Testing Performance Review

Metric	False Class	True Class	Macro Avg	Weighted Avg
Precision	0.83	0.79	0.81	0.82
Recall	0.87	0.73	0.80	0.82
F1 Score	0.85	0.76	0.81	0.82
Support	6000	4000	—	10000

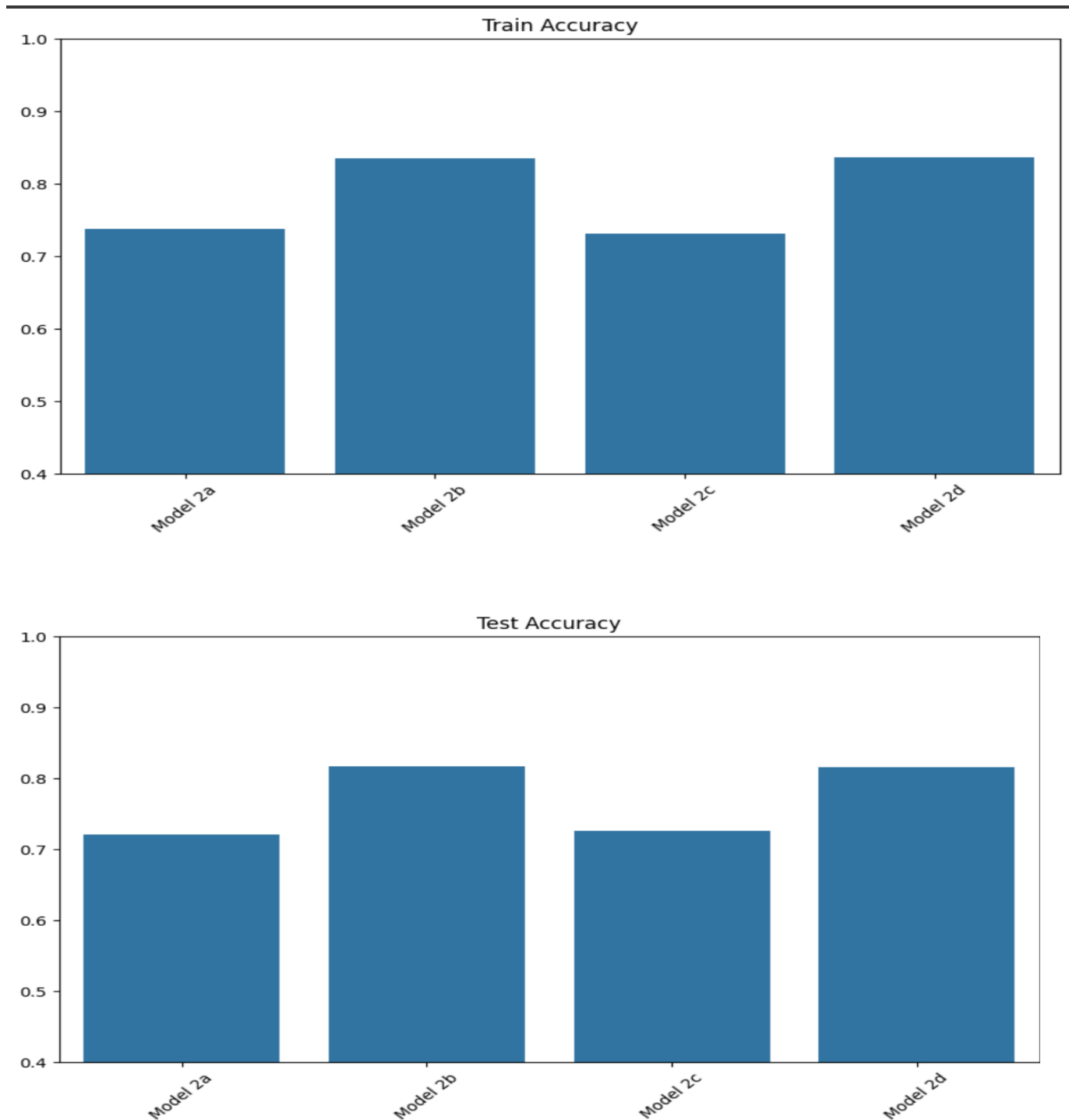
- Overall Accuracy: 81.65%
- Test F1 Score (for True class): 76.22%

Observations

- Model 2d performs identically to Model 2b in almost all key metrics, showing that the liblinear solver is just as effective as the default solver for this task.
- It achieves high test accuracy (81.65%) and balanced F1 scores, especially strong performance on the True class (76.22%).

COMPARING ALL PARAMETRIC MODEL:

Model Name	Train Accuracy (%)	Train F1-Score (%)	Test Accuracy (%)	Test F1-Score (%)
Model 2a	73.76	63.33	72.09	61.23
Model 2b	83.54	78.74	81.67	76.24
Model 2c	73.21	62.33	72.58	61.76
Model 2d	83.64	78.87	81.65	76.22



FINAL COMMENT ON PARAMETRIC ALGORITHM LOGISTIC REGRESSION:

Among the four logistic regression models, Model 2b and Model 2d clearly outperform the rest, both achieving over 81% test accuracy and F1 scores above 76%, thanks to the richer representation provided by RGB input. Model 2d edges out 2b in training metrics, making it a good choice when high training performance and solver flexibility (e.g., for L1 regularization) are desired. On the other hand, Models 2a and 2c underperform significantly, demonstrating that simplifying input via gray-scaling or PCA compression sacrifices important features necessary for accurate classification. Overall, Model 2d is recommended as the best shallow model for this task, offering both strong performance and practical flexibility.

Model 3a: Non-Parametric Model – KNN Classifier (k = 3, Gray-scale Images)

Model Architecture

Component	Details
Model Type	K-Nearest Neighbors (Non-parametric, instance-based)
k-value	3 (i.e., considers 3 nearest neighbors for classification)
Input Data	Flattened gray-scale image data
Distance Metric	Euclidean (default in most implementations)
Training Behavior	Lazy learner – no explicit training; all computation occurs at inference

Testing Performance Review

Metric	False Class	True Class	Macro Avg	Weighted Avg
Precision	0.77	0.69	0.73	0.74
Recall	0.81	0.64	0.72	0.74
F1 Score	0.79	0.66	0.73	0.74
Support	6000	4000	—	10000

- Overall Accuracy: 74.07%
- Test F1 Score (for True class): 66.27%

Observations

- As a **non-parametric, memory-based model**, KNN doesn't train in the traditional sense—its **classification relies on direct comparison to all training examples**.
- **Moderate overall accuracy (74.07%)**, but notably **lower F1 score for the True class (66.27%)**, shows it struggles more on positive cases.

Model 3b: Non-Parametric Model – KNN Classifier (k = 5, RGB Images)

Model Architecture

Component	Details
Model Type	K-Nearest Neighbors (Non-parametric, lazy learning)
k-value	5 (uses majority vote among 5 nearest neighbors)
Input Data	Flattened RGB image data (scaled)
Distance Metric	Euclidean distance (default setting)
Training Behavior	No training phase; predictions are computed during inference

Testing Performance Review

Metric	False Class	True Class	Macro Avg	Weighted Avg
Precision	0.80	0.83	0.81	0.81
Recall	0.91	0.66	0.78	0.81
F1 Score	0.85	0.73	0.79	0.80
Support	6000	4000	—	10000

- Overall Accuracy: 80.79%
- Test F1 Score (for True class): 73.3%

Observations

- Compared to Model 3a (k = 3, gray-scale), **Model 3b achieves much better performance**, showing that **RGB data captures richer features**, and a slightly larger k (5) offers improved stability.
- **Inference remains expensive**, as the model must compute distances across high-dimensional RGB vectors for each prediction.

Summary & Recommendation

Model 3b (KNN with k = 5 on RGB images) delivers **strong non-parametric performance**, achieving **80.79% accuracy** and a **True-class F1 score of 73.3%**. It clearly **outperforms its gray-scale counterpart (Model 3a)** and also beats some logistic regression models (e.g., Model 2a, 2c) thanks to the richer RGB representation. However, while it performs competitively, its **computational inefficiency at inference time** makes it less ideal for real-time or large-scale applications. This model serves well as a **benchmark or interpretable baseline**, especially in smaller datasets or low-stakes environments.

Model 3c: Non-Parametric Model – KNN Classifier (k = 5, PCA Gray-Scaled Features)

Model Architecture

Component	Details
Model Type	K-Nearest Neighbors (Non-parametric, lazy learning)
k-value	5 (uses majority vote among 5 nearest neighbors)
Input Data	PCA-reduced gray-scale image data
Distance Metric	Euclidean (default)
Training Behavior	No training phase; classification is instance-based

Testing Performance Review

Metric	False Class	True Class	Macro Avg	Weighted Avg
Precision	0.71	0.64	0.67	0.68
Recall	0.81	0.50	0.66	0.69
F1 Score	0.76	0.56	0.66	0.68
Support	6000	4000	—	10000

- Overall Accuracy: 68.58%
- Test F1 Score (for True class): 56.06%

Observations

- This model has the **lowest accuracy and F1 score** among all KNN variants and performs even worse than logistic regression on gray-scale (Model 2a).
- The use of **PCA-reduced features seems to have overly simplified the image data**, losing critical details required for effective classification.

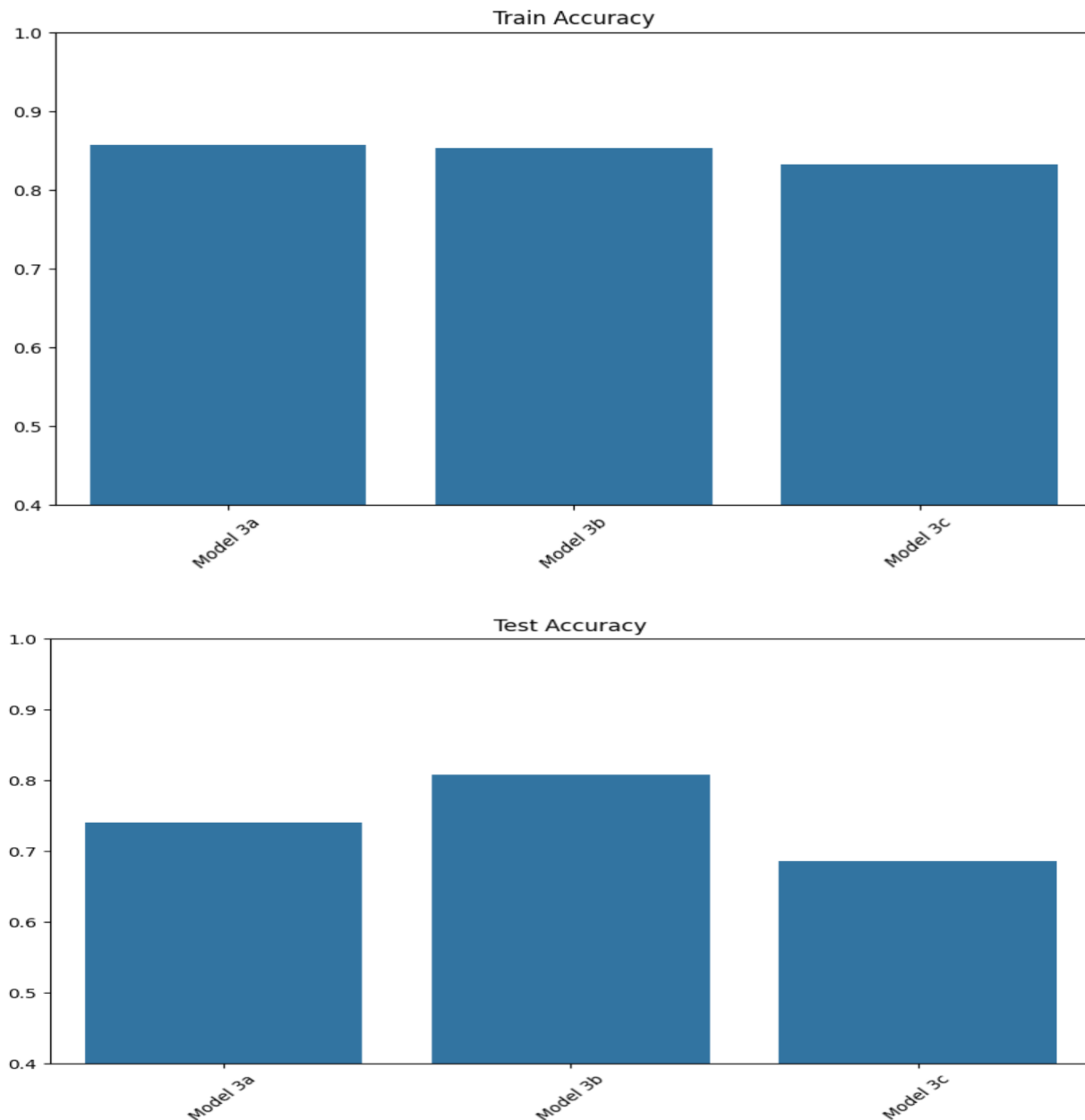
Summary & Recommendation

- Model 3c (KNN with k = 5 on PCA gray-scaled features) yields the **poorest performance** across all tested non-parametric and shallow models, with an **accuracy of just 68.58%** and a **True-class F1 score of 56.06%**. This highlights the drawback of applying **PCA on already limited gray-scale data**, particularly when paired with a memory-intensive method like KNN.

COMPARING ALL NON-PARAMETRIC MODELS:

Model Name	Train Accuracy (%)	Train F1-Score (%)	Test Accuracy (%)	Test F1-Score (%)
Model 3a	85.73	81.48	74.07	66.27
Model 3b	85.40	80.01	80.79	73.30
Model 3c	83.23	77.38	68.58	56.06

GRAPHICAL REPRESENTATION:



FINAL COMMENT ON NON-PARAMETRIC MODELS:

Among the three KNN models, Model 3b ($k=5$, RGB images) clearly outperforms the others, achieving the highest test accuracy (80.79%) and F1 score (73.30%), demonstrating the benefit of using full-color RGB data and a balanced k -value. It strikes a strong balance between learning local patterns and maintaining generalization. Model 3a ($k=3$, gray-scale), while achieving high training scores, suffers from a significant performance drop on the test set, highlighting overfitting and feature limitations due to gray-scaling. Model 3c ($k=5$, PCA gray-scale) performs the worst, with only 68.58% accuracy and a low F1 score of 56.06%, confirming that applying PCA on already simplified gray-scale data leads to a loss of essential discriminative information. Overall, Model 3b is the best KNN variant, offering the most reliable performance, while Model 3c is not suitable for practical use due to its poor generalization.

DISTINGUISHING FEATURES:

INCREMENTAL LEARNING:

An incremental learning interface that allows the user to add further images with labels to train the model and save the newly trained model in a new keras file namely model1a_new.keras.

Incremental Learning Interface

Upload Images (4)



Train Model



Training:



Accuracy:0.75 Loss: 0.3459358215332031



Save Model

Label Images

 Vehicle 

 Animal 


 Vehicle 


 Vehicle 

Another interface allowing the test of new model along in comparison to the old model.


Evaluate New Images


Evaluate New Mod...

Select Model: New Model 1a 



Prediction Results (Old Model 1a):

 Vehicle Probability: 59.67%

 Animal Probability: 40.33%

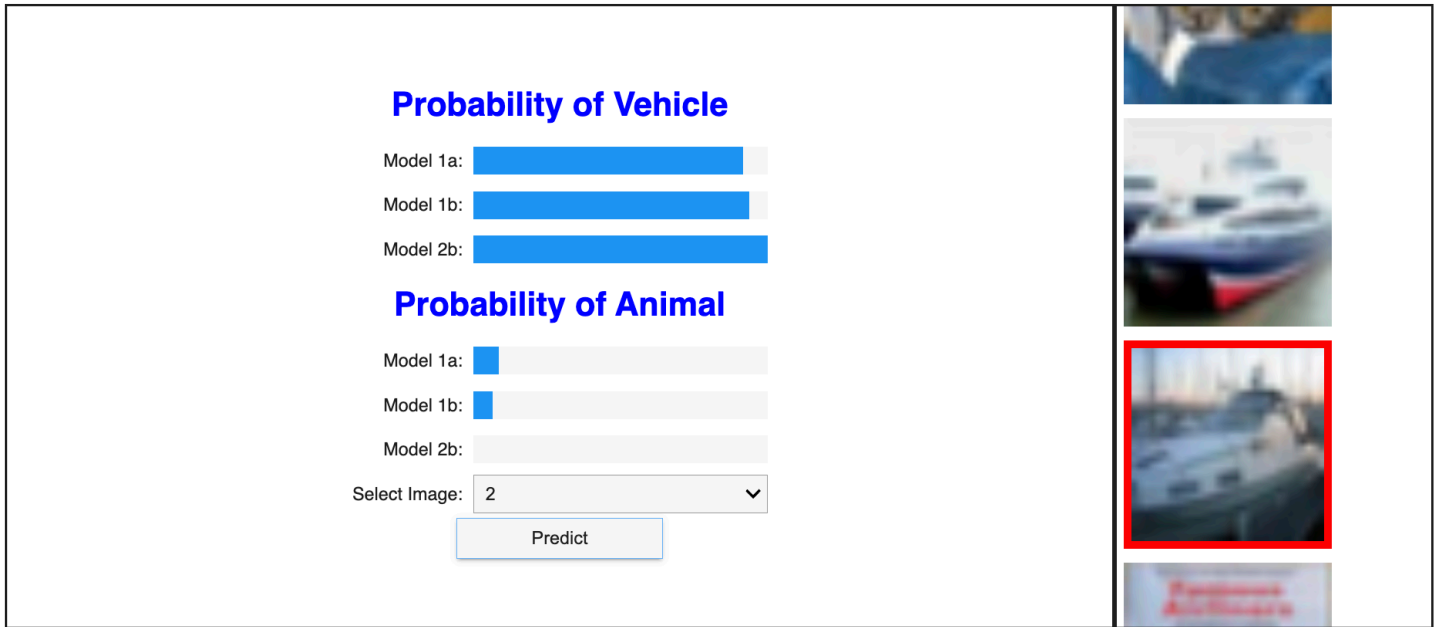
Prediction: VEHICLE

BUILT MORE THAN MINIMUM MODELS:

Trained five ANNs, four Logistic regression models and three KNN models with additional three models trained for the algorithms with different train-test split ratio.

GUI:

Created an interactive GUI to test the three best models through the test set.



EXPERIMENTED WITH PRE-TRAINED RESNET-18 NETWORK:

Tried fine-tuning a pretrained resnet-18 model on CIFAR-10 dataset but could not achieve high accuracy(39%) despite changing input and classification layers.

CONCLUSION:

This report presents a detailed analysis of the implementation and evaluation of different machine learning models for image classification. The objective is to classify images into two categories animals and vehicles. The dataset used for training and testing the models is the CIFAR-10 dataset, consisting of 50,000 training images and 10,000 testing images. The 10 classes of the dataset were converted into binary classes.

This CEP allowed us to practice theoretical knowledge learnt in class by training several ANNs, parametric and non-parametric models. It helped us gauge a better understanding of training metrics and model selection.