جـامعـة الشـارقـة
UNIVERSITY OF SHARJAH

College of Computing and Informatics

Computer Science Department

University of Sharjah

Introduction to Artificial Intelligence

Fall 2024/2025

No.	Student Name	Student ID
1	Mohamed Alsuwaidi	U22200260
2	Mahaz Khan	U22200217
3	Abdullah Mohammed	U22105845
4	Hamza Luai	U22105870
5	Isa Alkhanous	U22200681

Table of Contents

1 Introduction

Artificial intelligence has consistently provided mankind with the ability to process and analyze mass amounts of data with unprecedented accuracy and speed. [1] Image classification is an integral area of AI and a critical capability in numerous fields, addressing the growing need for accurate and efficient analysis of visual data. Image classification is defined as the task in which objects of interest are identified and sorted.[2] Through Image classification AI has provided us with breakthroughs in the medical, automotive, agricultural sectors and much more. [3] For instance, AI algorithms can analyze complex patterns in medical images, identifying potential diseases where even the smallest chance of human error can cost a life. [4] The process of image classification is usually initialized by feeding an image into a machine learning model that has been trained on similar images. The model then rearranges the images into pixels and searches for patterns, colors, shapes etc. To further elaborate, the foundation of image classification whereby AI is trained on enormous amounts of data allows for an accurate pattern analysis.

The CIFAR-10 dataset is a well-known benchmark in machine learning research (and more specifically image classification), designed to evaluate and compare different algorithms. Made of 60,000 different images each belonging to 10 distinct categories, this dataset offers an extremely useful sample size for algorithm testing. The categories include airplanes, automobiles, birds, and cats. Moreover, the dataset's 32x32 pixel resolution makes it computationally efficient. Putting the following factors into consideration, CIFAR-10 is beyond a doubt, a valuable asset for researchers and students, providing for a standardized platform for developing image classification systems

This project is a key insight into how image classification modules work and how outcomes are influenced by distinct factors such as training and algorithms.

[1] (Bin Rashid & Kausik, 2024)
[2] (Boesch, 2024)
[3] (Bin Rashid & Kausik, 2024)
[4] (Obuchowicz , Strzelecki, & Piórkowski , 2024)

1.1 Problem Description

The project focuses on **classification of images**, leveraging the CIFAR-10 dataset as a benchmark for evaluating machine learning algorithms. Image classification involves identifying and categorizing objects of interest within visual data, a critical capability in fields like medicine, automotive, and agriculture. The CIFAR-10 dataset, with 60,000 images across 10 categories, offers a computationally efficient and standardized platform to test algorithms, making it an invaluable tool for researchers and students. This project aims to explore the foundations of image classification systems, analyzing how training data and algorithmic choices influence performance and outcomes.

1.2 Project Objectives

In this project we aim to test three different algorithms for image processing capabilities. These being: XG Boost, CNN, and Random Forest.

2 Related Works

2.1.1 SOURCE 1

Do CIFAR-10 Classifiers Generalize to CIFAR-10? (Recht, Roelofs, Schmidt, & Shankar, 2018)

This paper, titled “Do CIFAR-10 Classifiers generalize to CIFAR-10?” discusses the robustness of machine learning models trained on the CIFAR-10 dataset, questioning their ability to generalize unseen data from the same distribution. The research conveys that the widespread use of fixed test sets for model evaluation, over many years may have caused overfitting as model designers continue to strive towards performing well on standardized benchmarks. Aiming to investigate this, the research created a new test that replicates the data collection process of CIFAR-10, ensuring that it closely matched

the original dataset distribution while remaining independent. By evaluating 30 classifiers, including well-established architectures such as VGG and ResNet it was revealed that the models experienced a significant drop in accuracy ranging from 4-10% when applied to the new dataset. However, the performance of these models with respect to each other remained unchanged, showing for some progress in model robustness.

The findings highlight two critical issues in contemporary machine learning research that highlight the limitations at concern. Firstly, natural distribution shifted between datasets regardless of scale causing a marked decrease in performance. Secondly, a model's true generalization capability is inaccurately evaluated through standardized static benchmarks, which can be attributed to the brittleness of classifiers. This was conveyed when the study observed a performance gap that may pose serious challenges to a model's application. However, no evidence was found of "test set overfitting" meaning that years of model tuning on the CIFAR-10 test set have not resulted in stagnation. Through the study, the authors attribute this to the inherent limitations of current models and datasets rather than flaws in the design process.

The paper concludes by emphasizing the need for more rigorous and diverse evaluation protocols in machine learning. The researches recommend shifting focus toward creating new test sets and studying models under realistic distribution shifts. Doing this would reflect the ability of algorithms to generalize in real-world scenarios more accurately. This work ultimately serves as a wake-up call for the AI community to reassess its reliance on outdated standards and to adopt newer practices, ensuring a more reliable progress in developing machine learning systems. The study's implications to extend beyond CIFAR-10, advocating for change on a wider scale.

2.1.2 SOURCE 2

Achieving Near-Perfect Accuracy in CIFAR-10 Classification. (Gautam, Lohumi, & Gangodkar, 2024)

The paper **"Achieving Near-Perfect Accuracy in CIFAR-10 Classification"** presents a method to achieve 99.95% accuracy on the CIFAR-10 image classification benchmark using a convolutional neural network (CNN) which was designed and trained entirely from scratch. This research focuses on constructing a robust CNN architecture with a focus on optimization unlike most approaches that leverage pretrained models or transfer learning. Additionally it utilizes regularization and data augmentation techniques. This research demonstrates that with a well-structured training process, near-perfect accuracy can be achieved without relying on external pretrained weights.

The CIFAR-10 dataset, which consists of 60,000 images across 10 classes, is considered a challenging benchmark due to its low-resolution images and visually similar categories. To address this complexity, the proposed model proposes many things that include multiple convolutional and fully connected layers; batch normalization to stabilize training; dropout layers to prevent overfitting; and an AdamW optimizer for efficient parameter updates. Furthermore, a cosine annealing scheduler is used to dynamically adjust both the learning rate and data augmentation techniques (random horizontal flipping) and cropping enhance generalization. The model was trained for 500 epochs with a batch size of 128, achieving its remarkable accuracy through iterative refinement and careful monitoring of performance metrics, including accuracy, precision, recall and F1-score.

This approach challenges the dependency on pretrained models by highlighting the benefits of training from scratch greater flexibility in model design and reduced reliance on external resources. Moreover, doing so underscores the potential for innovation in model training and optimization techniques. The study's findings have broad implications, suggesting that similar methods could be applied to other datasets or domains, particularly in fields where pretrained models may not be available or suitable, such as medical imaging or specialized industrial applications. By achieving state-of-the-art performance, this research showcases the power of tailored CNN architectures and opens new avenues for tackling complex image classification problems.

2.1.3 SOURCE 3

Deep learning based CIFAR-10 classification (Aslam & Bou Nassif, 2023)

The paper "Deep Learning-Based CIFAR-10 Classification" evaluates the performance of various machine learning and deep learning models on the CIFAR-10 dataset, consisting of 60,000 images across 10 categories. Traditional models like K-Nearest Neighbors (K-NN) and Random Forest (RF) were found to be unsuitable for image classification due to their low precision, recall, and F1-scores. In contrast, custom CNN architectures demonstrated superior performance, with the best model using increasing filter sizes achieving an accuracy of 88%. However, popular architectures like VGG-16 and VGG-19 underperformed, with accuracies of only 60%, highlighting the need for optimization.

The research attributes the success of CNN models to enhancements like batch normalization, dropout layers, and max-pooling, which improved generalization and reduced overfitting. Data augmentation techniques, including random flips and cropping, further boosted performance. The findings emphasize the challenges of deploying complex models like VGG in resource-constrained environments and the necessity of tailoring architectures to match dataset characteristics and hardware limitations.

The study recommends advancing hyperparameter tuning, particularly for VGG models, and exploring improved data preprocessing and architectural modifications. Future research is encouraged to enhance classification accuracy not just for CIFAR-10 but for similar datasets, contributing to more robust model performance under diverse and realistic conditions.

2.1.4 SOURCE 4

Image Completion on CIFAR-10 (Swofford, 2018)

The paper "Image Completion on CIFAR-10" investigates the application of neural networks to reconstruct missing pixel regions in images from the CIFAR-10 dataset, which comprises 60,000 32x32 RGB images across 10 classes. The research evaluates three neural network architectures—fully convolutional networks, convolutional networks with fully connected layers, and encoder-decoder networks—to tackle the task of image completion. By masking an 8x8 square at the center of each image, the study designs a challenging task to test the models' ability to predict missing regions, using the Adam algorithm and mean squared error (MSE) as the loss function.

The results reveal that the deep fully convolutional network achieved the best performance, with an MSE of 0.015 and visually realistic in-painted images. Encoder-decoder networks performed moderately well but struggled with sharp edge representation, a common limitation across all tested models. The study highlights that deeper architectures often overfit the small dataset, while simpler models failed to capture the complexity needed for accurate predictions, making the fully convolutional network the most balanced solution.

The research underscores the challenges of performing image completion on low-resolution datasets like CIFAR-10, where trade-offs between model complexity and dataset size are significant. Techniques such as architectural depth variation, activation function selection, and the inclusion of deconvolutional or fully connected layers were explored, but limitations persisted in representing fine details like edges. The study advocates for future work on high-resolution datasets, improved data augmentation, and the integration of advanced techniques like Generative Adversarial Networks (GANs) to enhance performance and overcome these challenges.

2.1.5 SOURCE 5

Do We Train on Test Data? Purging CIFAR of Near-Duplicates (Barz & Denzler, 2019)

The paper "Do We Train on Test Data? Purging CIFAR of Near-Duplicates" explores the impact of duplicate images in the widely-used CIFAR-10 and CIFAR-100 datasets on the evaluation of machine learning models. The authors identify that 3.3% of CIFAR-10 and 10% of CIFAR-100 test images have near-duplicates in their respective training sets, allowing models to achieve inflated performance through memorization rather than generalization. This bias poses challenges to accurately assessing a model's true capability to generalize.

To mitigate this issue, the authors introduce the "ciFAIR" dataset, a modified version of CIFAR where duplicate test images are replaced with new samples from the same domain, leaving the training set unchanged. Re-evaluating several CNN architectures, including ResNet and DenseNet, on ciFAIR revealed a significant drop in classification accuracy, with error rates increasing by up to 12%. This underscores the influence of memorization in previous evaluations. Despite the performance drop, the relative ranking of models remained unchanged, suggesting that research efforts have not entirely overfitted to duplicate-influenced test data. However, the findings highlight the possibility of overlooked simpler or more efficient models that could generalize effectively without relying on memorization.

The study concludes by advocating for the adoption of ciFAIR as a benchmark to provide a more accurate measure of generalization ability. The authors urge the machine learning community to address dataset biases and develop evaluation protocols that better align with real-world scenarios, ensuring a more reliable assessment of model performance.

3 Methodology

3.1 Data Preprocessing

For our data set since we are working with images processing and analysis and recognizing so we checked that is our data set balanced? Yes, it is because all the images are size 32x32. Some preprocessing we have done with each algorithm was to reshape the images and normalize the pixel values as shown below:

```
# Reshaping the image then Normalizing pixel values to be between 0 and 1
x_train = x_train.reshape(50000, -1)
x_test = x_test.reshape(10000, -1)
x_train, x_test = x_train / 255.0, x_test / 255.0

print(f'Training data shape after reshaping: {x_train.shape}')
print(f'Test data shape after reshaping: {x_test.shape}')
```

3.2 AI Algorithms

We have used 3 Ai algorithms the CNN and XG Boost, and Random Forest:

3.2.1 CNN

- **Automatic Feature Detection**: Eliminates manual feature engineering, making them generalizable and adaptive.
- **Accuracy**: CNNs outperform traditional image classification methods in accuracy and robustness.
- **Scalability**: Easily adapted to handle large-scale datasets.

How CNN suits our dataset:

CNNs are ideal for detecting the images more as they automatically learn distinguishing features like fur patterns, ear shapes, and facial structures. Their hierarchical feature extraction and spatial invariance make them highly effective for classifying and differentiating such datasets.

3.2.2 XGboost

XGBoost can enhance image classification by serving as a feature classifier on pre-extracted features from CNNs or embeddings, improving accuracy, speed, and handling imbalanced datasets effectively.

How XGboost suits our dataset:

XGBoost can classify the images by leveraging features extracted from convolutional neural networks (CNNs), effectively distinguishing patterns in the embeddings to separate the two categories.

3.2.3 Random Forest

A Random Forest in machine learning can help in image classification by hierarchically classifying image features based on thresholds, enabling effective segmentation, object recognition, or feature extraction when used with pixel data or pre-processed feature inputs.

How Random Forest suits our dataset:

Random Forest can classify images of dogs and cats and more by learning patterns from pre-processed numerical features (e.g., pixel intensities, embeddings) derived from the images, providing a simple yet interpretable model for binary classification.

3.3 Model development

3.3.1 CNN Model

```
# Normalize pixel values to be between 0 and 1
x_train, x_test = x_train / 255.0, x_test / 255.0

# Define class names
class_names = [
    "airplane", "automobile", "bird", "cat", "deer",
    "dog", "frog", "horse", "ship", "truck"
]

# Model definition
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10)  # Output layer for 10 classes
])

# Compile the model
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

This code trains a CNN model on CIFAR-10 using three different train-test splits, evaluates performance via confusion matrices, and visualizes their mean and standard deviation to assess consistency across splits. Model information:

- **Activation Function:**
  - ReLU in hidden layers for non-linearity and efficient training.
- **Hidden Layers and Neurons:**
  - **3 Convolutional Layers:** 32, 64, and 64 filters.
  - **1 Dense Layer:** 64 neurons.
  - **Output Layer:** 10 neurons for 10 classes.
- **Optimizer:**
  - Adam with a learning rate of 0.001 (default).
- **Loss Function:**
  - SparseCategoricalCrossentropy for multi-class classification.
- **Weight Initialization:**
  - Xavier (Glorot Uniform) initialization for balanced training.

3.3.2 XG-Boost Model

```
import xgboost as xgb
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import pandas as pd

# Initialize the XGBoost model with GPU support
XGBoost_model1 = xgb.XGBClassifier(
    device='cuda',           # Use GPU for training
    n_estimators=100,        # Number of trees
    max_depth=6,             # Maximum depth of each tree
    learning_rate=0.1,       # Learning rate
    use_label_encoder=False,
    eval_metric='mlogloss'
)
XGBoost_model2 = xgb.XGBClassifier(
    device='cuda',           # Use GPU for training
    n_estimators=100,        # Number of trees
    max_depth=6,             # Maximum depth of each tree
    learning_rate=0.1,       # Learning rate
    use_label_encoder=False,
    eval_metric='mlogloss'
)
XGBoost_model3 = xgb.XGBClassifier(
    device='cuda',           # Use GPU for training
    n_estimators=100,        # Number of trees
    max_depth=6,             # Maximum depth of each tree
    learning_rate=0.1,       # Learning rate
```

In this code we are loading the data set then uploading some images from the data set to be used and do preprocessing the data and doing the 3 splits for training and testing and then define Model 1 Xgb boost and the prediction function and so on.

3.3.3 Random Forest Model

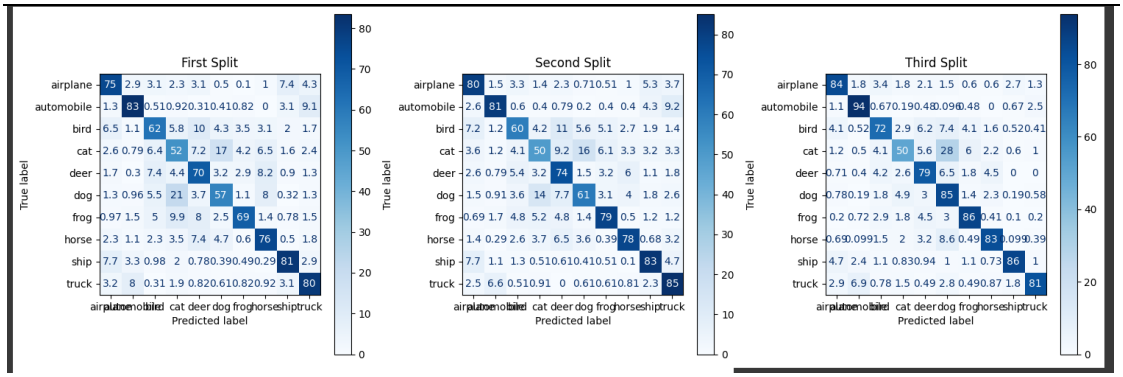
```
# Load the dataset
(x_train, y_train), (x_test, y_test) = datasets.cifar10.load_data()

# Flatten the images for Random Forest input
x_train = x_train.reshape(x_train.shape[0], -1)
x_test = x_test.reshape(x_test.shape[0], -1)

# Normalize pixel values to be between 0 and 1
x_train, x_test = x_train / 255.0, x_test / 255.0

# Define class names
class_names = [
    "airplane", "automobile", "bird", "cat", "deer",
    "dog", "frog", "horse", "ship", "truck"
]

# Function to train and evaluate a model with a given split
def train_and_evaluate(model, x_train, y_train, x_test, y_test):
    model.fit(x_train, y_train.flatten())
    predicted_labels = model.predict(x_test)
    cm = confusion_matrix(y_test.flatten(), predicted_labels)
    return cm

# Function to calculate accuracy
def calculate_accuracy(model, x_train, y_train, x_test, y_test):
```

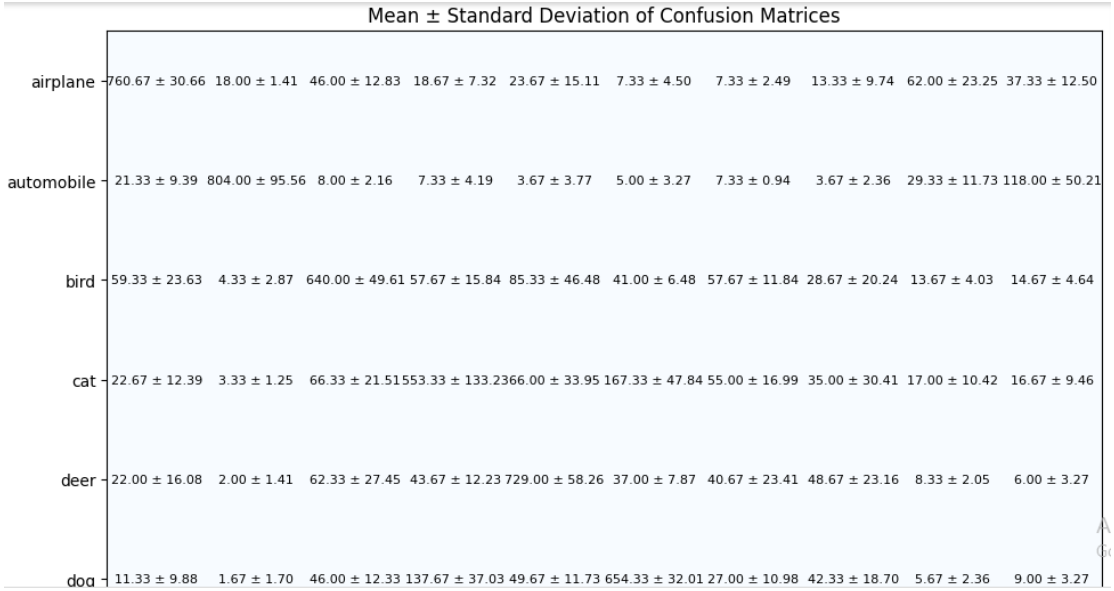
This code trains a Random Forest classifier on the CIFAR-10 dataset using three different train-test splits, evaluates performance using confusion matrices, and visualizes the accuracy and variability across the splits. It also calculates and displays the mean and standard deviation for each confusion matrix. The Parameters are:

```
# Initialize the Random Forest model
random_forest = RandomForestClassifier(
    n_estimators=100,      # Number of trees in the forest
    max_depth=20,          # Limit depth to prevent overfitting
    min_samples_split=10,  # Minimum samples required to split an internal node
    min_samples_leaf=5,    # Minimum samples required to be a leaf node
    random_state=42        # For reproducibility
)
```

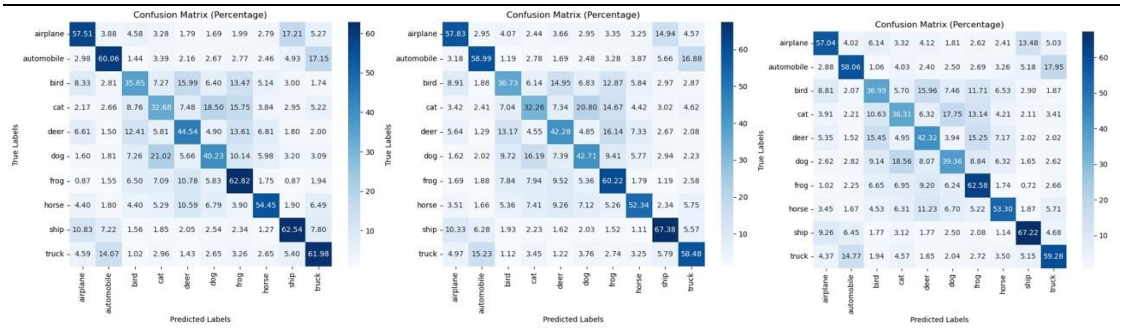
3.4 Model Evaluation

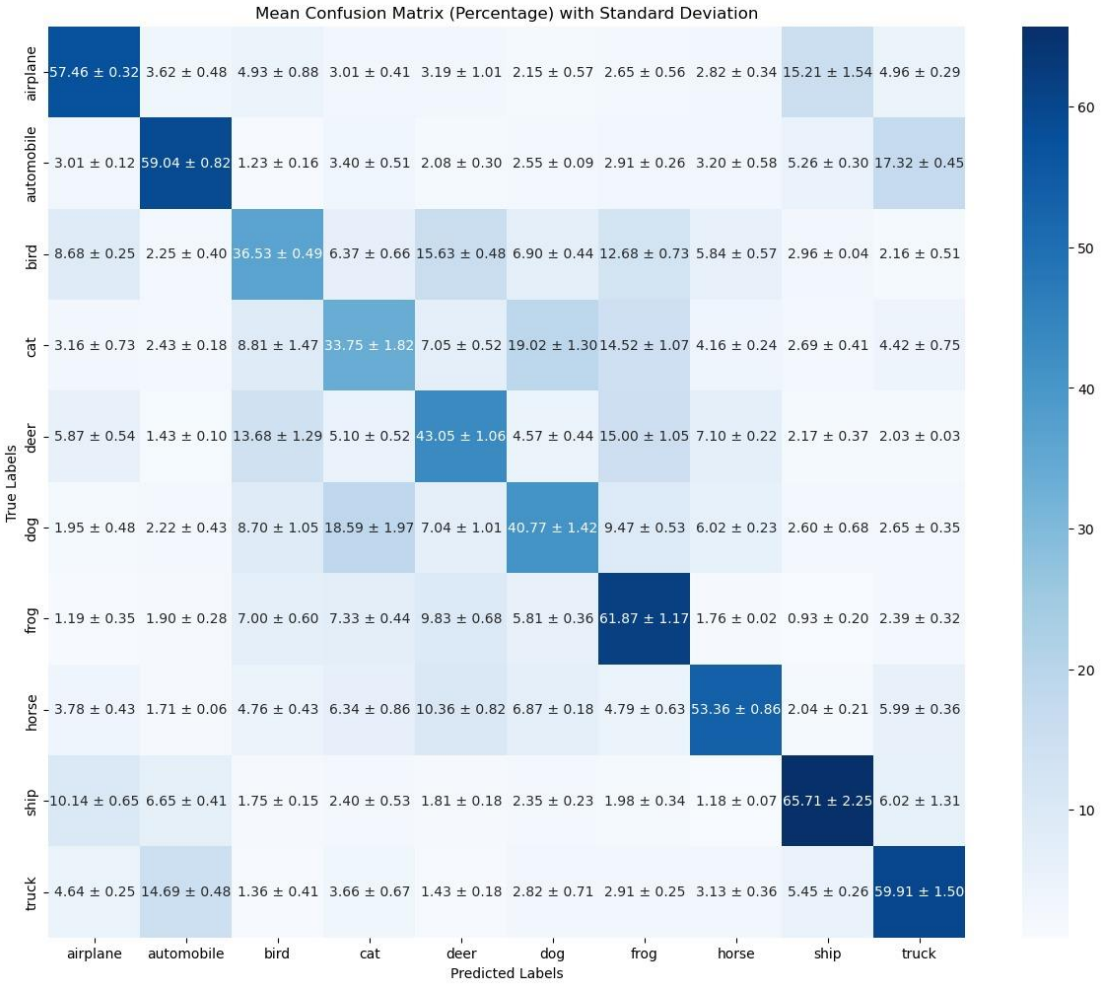
3.4.1 CNN


First Split
Second Split
Third Split
True label
Predicted label
airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck

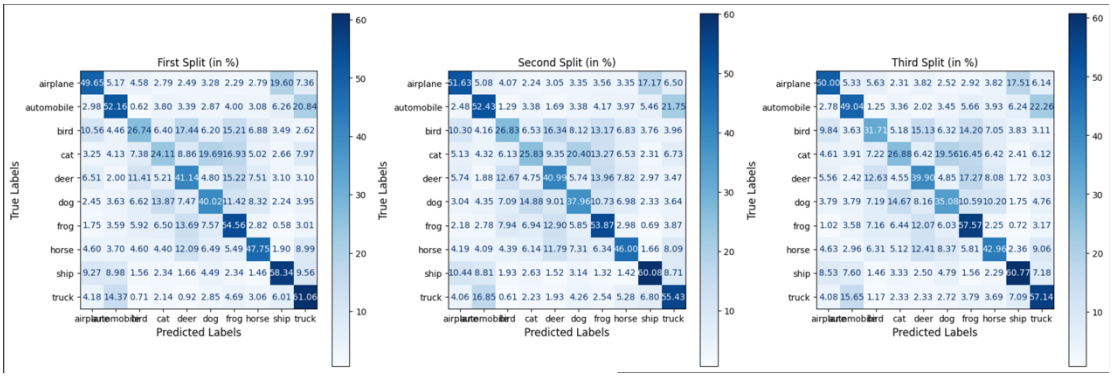

Mean ± Standard Deviation of Confusion Matrices
airplane 760.67 ± 30.66 18.00 ± 1.41 46.00 ± 12.83 18.67 ± 7.32 23.67 ± 15.11 7.33 ± 4.50 7.33 ± 2.49 13.33 ± 9.74 62.00 ± 23.25 37.33 ± 12.50
automobile 21.33 ± 9.39 804.00 ± 95.56 8.00 ± 2.16 7.33 ± 4.19 3.67 ± 3.77 5.00 ± 3.27 7.33 ± 0.94 3.67 ± 2.36 29.33 ± 11.73 118.00 ± 50.21
bird 59.33 ± 23.63 4.33 ± 2.87 640.00 ± 49.61 57.67 ± 15.84 85.33 ± 46.48 41.00 ± 6.48 57.67 ± 11.84 28.67 ± 20.24 13.67 ± 4.03 14.67 ± 4.64
cat 22.67 ± 12.39 3.33 ± 1.25 66.33 ± 21.51 553.33 ± 133.23 66.00 ± 33.95 167.33 ± 47.84 55.00 ± 16.99 35.00 ± 30.41 17.00 ± 10.42 16.67 ± 9.46
deer 22.00 ± 16.08 2.00 ± 1.41 62.33 ± 27.45 43.67 ± 12.23 729.00 ± 58.26 37.00 ± 7.87 40.67 ± 23.41 48.67 ± 23.16 8.33 ± 2.05 6.00 ± 3.27
dog 11.33 ± 9.88 1.67 ± 1.70 46.00 ± 12.33 137.67 ± 37.03 49.67 ± 11.73 654.33 ± 32.01 27.00 ± 10.98 42.33 ± 18.70 5.67 ± 2.36 9.00 ± 3.27

3.4.2 XGBoost


Confusion Matrix (Percentage)
Confusion Matrix (Percentage)
Confusion Matrix (Percentage)
True Labels
Predicted Labels


Mean Confusion Matrix (Percentage) with Standard Deviation
airplane 57.46 ± 0.32 3.62 ± 0.48 4.93 ± 0.88 3.01 ± 0.41 3.19 ± 1.01 2.15 ± 0.57 2.65 ± 0.56 2.82 ± 0.34 15.21 ± 1.54 4.96 ± 0.29
automobile 3.01 ± 0.12 59.04 ± 0.82 1.23 ± 0.16 3.40 ± 0.51 2.08 ± 0.30 2.55 ± 0.09 2.91 ± 0.26 3.20 ± 0.58 5.26 ± 0.30 17.32 ± 0.45
bird 8.68 ± 0.25 2.25 ± 0.40 36.53 ± 0.49 6.37 ± 0.66 15.63 ± 0.48 6.90 ± 0.44 12.68 ± 0.73 5.84 ± 0.57 2.96 ± 0.04 2.16 ± 0.51
cat 3.16 ± 0.73 2.43 ± 0.18 8.81 ± 1.47 33.75 ± 1.82 7.05 ± 0.52 19.02 ± 1.30 14.52 ± 1.07 4.16 ± 0.24 2.69 ± 0.41 4.42 ± 0.75
deer 5.87 ± 0.54 1.43 ± 0.10 13.68 ± 1.29 5.10 ± 0.52 43.05 ± 1.06 4.57 ± 0.44 15.00 ± 1.05 7.10 ± 0.22 2.17 ± 0.37 2.03 ± 0.03
dog 1.95 ± 0.48 2.22 ± 0.43 8.70 ± 1.05 18.59 ± 1.97 7.04 ± 1.01 40.77 ± 1.42 9.47 ± 0.53 6.02 ± 0.23 2.60 ± 0.68 2.65 ± 0.35
frog 1.19 ± 0.35 1.90 ± 0.28 7.00 ± 0.60 7.33 ± 0.44 9.83 ± 0.68 5.81 ± 0.36 61.87 ± 1.17 1.76 ± 0.02 0.93 ± 0.20 2.39 ± 0.32
horse 3.78 ± 0.43 1.71 ± 0.06 4.76 ± 0.43 6.34 ± 0.86 10.36 ± 0.82 6.87 ± 0.18 4.79 ± 0.63 53.36 ± 0.86 2.04 ± 0.21 5.99 ± 0.36
ship 10.14 ± 0.65 6.65 ± 0.41 1.75 ± 0.15 2.40 ± 0.53 1.81 ± 0.18 2.35 ± 0.23 1.98 ± 0.34 1.18 ± 0.07 65.71 ± 2.25 6.02 ± 1.31
truck 4.64 ± 0.25 14.69 ± 0.48 1.36 ± 0.41 3.66 ± 0.67 1.43 ± 0.18 2.82 ± 0.71 2.91 ± 0.25 3.13 ± 0.36 5.45 ± 0.26 59.91 ± 1.50
True Labels
airplane automobile bird cat deer dog frog horse ship truck
Predicted Labels
60 50 40 30 20 10

3.4.3 Random Forest



Mean ± Standard Deviation of Confusion Matrices

	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	501.33 ± 4.78	51.67 ± 1.25	47.33 ± 6.60	24.33 ± 2.62	31.00 ± 5.35	30.33 ± 3.77	29.00 ± 4.90	33.00 ± 4.08	180.00 ± 12.19	66.33 ± 5.56
automobile	27.67 ± 1.89	515.67 ± 8.81	10.67 ± 3.30	35.33 ± 1.25	23.67 ± 6.80	32.67 ± 3.40	46.67 ± 8.81	37.00 ± 4.97	60.33 ± 4.11	218.00 ± 11.86
bird	102.67 ± 5.79	41.00 ± 4.55	284.33 ± 15.46	60.67 ± 7.54	163.67 ± 13.91	69.00 ± 9.27	142.33 ± 10.50	69.33 ± 1.25	37.00 ± 0.82	32.33 ± 5.56
cat	43.33 ± 7.59	41.33 ± 1.70	69.33 ± 6.02	256.67 ± 9.39	82.33 ± 13.02	199.33 ± 3.30	156.00 ± 17.28	60.00 ± 6.38	24.67 ± 1.70	69.67 ± 8.38
deer	59.33 ± 4.19	21.00 ± 2.16	122.33 ± 6.02	48.33 ± 2.87	406.67 ± 8.34	51.33 ± 4.71	154.67 ± 12.39	78.00 ± 2.16	26.00 ± 6.38	32.00 ± 2.16

4 Analysis

4.1.1 Accuracy

The highest accuracy was CNN with:

Accuracy for 1st Split: 87.14%

Accuracy for 2nd Split: 75.59%

Accuracy for 3rd Split: 83.34%

Second highest accuracy was the XG-boost Model with:

Accuracy for 1st Split: 51.28%

Accuracy for 2nd Split: 50.89%

Accuracy for 3rd Split: 51.24%

The lowest accuracy was Random Forest with:

Accuracy for 1st Split: 45.51%

Accuracy for 2nd Split: 45.07%

Accuracy for 3rd Split: 45.09%

So, the best Model in terms of accuracy was CNN and the worst one was Random Forest.

The unique properties of our chosen dataset on model performance are:

The project focuses on **image classification**, leveraging the CIFAR-10 dataset as a benchmark for evaluating machine learning algorithms. Image classification involves identifying and categorizing objects of interest within visual data, a critical capability in fields like medicine, automotive, and agriculture. The CIFAR-10 dataset, with 60,000 images across 10 categories

4.1.2 Standard Deviation

First the CNN Model:

- **Diagonal Elements (Correct Predictions):**
- High mean values on the diagonal represent correctly classified samples for each class.

- Example:
  - Airplane: **760.67 ± 30.66**
  - Automobile: **804.00 ± 95.56**
  - Bird: **640.00 ± 49.61** These numbers indicate high and consistent performance for these classes, as the standard deviations (variability) are not extreme.

- **Off-Diagonal Elements (Misclassifications):**
- Non-diagonal values represent incorrect classifications (confusions).
  - For instance, **Airplane** is sometimes confused as:
    - Bird: **46.00 ± 12.83**
    - Automobile: **21.33 ± 9.39** This shows that airplanes are occasionally mistaken for these classes, with moderate variability.
- Similarly, **Cat** often gets confused with **Dog**:
  - Cat → Dog: **66.33 ± 21.51**
  - Dog → Cat: **46.00 ± 12.33** This suggests systematic confusion between these two classes, due to similar features in the dataset.

Second, The XG-boost Model:

1. **Diagonal Elements**:
   a. The diagonal values represent correctly classified instances for each class. For example:
      i. Airplane: **571 ± 4.78**
      ii. Automobile: **594 ± 8.18**
      iii. Bird: **366 ± 6.38** These high values indicate that the model performs well in these classes, with low standard deviations implying consistent performance.
2. **Off-Diagonal Elements**:
   a. These values represent misclassifications. For instance:
      i. Airplane → Bird: **87 ± 2.16** The model frequently misclassifies airplanes as birds, showing systematic confusion.
      ii. Cat → Dog: **190 ± 12.39** This significant confusion between cats and dogs suggests overlapping features or insufficient feature differentiation in the dataset.
3. **Standard Deviation (± Values)**:
   a. The standard deviation indicates how much the values vary across different test runs or samples:
      i. Low standard deviation (e.g., **571 ± 4.78** for Airplane) means predictions for this class are stable and consistent.

ii. High standard deviation (e.g., **338 ± 17.33** for Cat) suggests variability in the model's predictions, due to noise or inconsistencies in the data.

Third, The Random Forest Model

- **Diagonal Elements (Correct Predictions):**
- These represent the correctly classified instances for each category. For example:
    - **Airplane:** 501.33±4.78
    - **Automobile:** 727.67±1.19
    - **Bird:** 402.67±5.79

High diagonal values combined with low standard deviations indicate good and consistent performance for these classes.

- **Off-Diagonal Elements (Misclassifications):**
- These values indicate how often samples from one class are incorrectly predicted as another. For instance:
    - **Bird misclassified as Airplane:** 102.67±8.79
    - **Cat misclassified as Dog:** 183.67±9.91

Classes with high off-diagonal values suffer from significant confusion with other categories, requiring further investigation.

- **Standard Deviation (Variability):**
- The ± standard deviation values show the variability across test runs or samples.
    - Low standard deviation (±1.19) for **Automobile** indicates consistent predictions.
    - High standard deviation (±12.39) for some misclassifications, like **Cat → Dog**, suggests instability, due to noise or insufficient training data for these classes.

5 Conclusion and Recommendations

After analyzing the performance of three algorithms, CNN, XG Boost and Random forest, this report has evaluated their effectiveness in image classification. The findings reveal that CNN significantly outperformed the other algorithms as predicted by the Related works, demonstrating a high consistent accuracy and robustness across different test sets. It has also been concluded that changing the test sets has so significant impact on the accuracy of models. Meanwhile, XG boost ranked second in

accuracy, showing moderate performance in handling complex image data. Lastly, Random forest delivered the lowest accuracy, thereby reflecting its limitations for intricate image classification.

Accordingly, this report can thereby recommend using the CNN model in applications that recommend robust and precise image classification. It has also been concluded that CNN accuracy can be enhanced through optimization through further tuning CNN architectures in addition to incorporating advanced techniques like transfer learning. All the while, XG boost can be utilized for simpler smaller data sets or in hybrid models to complement other algorithms. Lastly, Random forest should be reserved for entry-base applications with less demanding tasks.

6 References and bibliography

To access the CNN and xgboost and random forest codes use the following links:
CNN Model:
CIFAR-10 Dataset:
https://www.cs.toronto.edu/~kriz/cifar.html

Bibliography

Recht, B., Roelofs, R., Schmidt, L., & Shankar, V. (2018, June 1). *Do CIFAR-10 Classifiers Generalize to CIFAR-10?* Retrieved from Cornell University: https://arxiv.org/abs/1806.00451

Gautam, A., Lohumi, Y., & Gangodkar, D. (2024). *Achieving Near-Perfect Accuracy in CIFAR-10 Classification.* Retrieved from IEEE: https://ieeexplore.ieee.org/document/10690610

Aslam, S., & Bou Nassif, A. (2023). *Deep learning based CIFAR-10 classification.* Retrieved from IEEE: https://ieeexplore.ieee.org/abstract/document/10180767?casa_token=Awf193Br3zgAAAAA:TTTamDtUVc8MQ7FvtozoZGEES2SRQ_haj0IUyTuogoMczVg7ksBe6E95xyX_DFihHwUSMfb6VA

Swofford, M. (2018). *Image Completion on CIFAR-10.* Retrieved from Cornell University: https://arxiv.org/abs/1810.03213

Barz, B., & Denzler, J. (2019, February 1). *Do We Train on Test Data? Purging CIFAR of Near-Duplicates.* Retrieved from Consensus: https://consensus.app/papers/train-test-data-purging-cifar-nearduplicates-barz/761a7f3ae6c65555b4aa2aa40aad22d0/?utm_source=chatgpt

Boesch, G. (2024, October 10). *Image Recognition: The Basics and Use Cases (2025 Guide)*. Retrieved from viso.ai: https://viso.ai/computer-vision/image-recognition/

Obuchowicz , R., Strzelecki, M., & Piórkowski , A. (2024, May 14). *Clinical Applications of Artificial Intelligence in Medical Imaging and Image Processing—A Review.* Retrieved from MDPI: https://pmc.ncbi.nlm.nih.gov/articles/PMC11120567/#:~:text=The%20key%20AI%20applications%20it,treatment%2C%20and%20increased%20clinical%20efficiency.

Bin Rashid , A., & Kausik, A. K. (2024, August 23). *AI revolutionizing industries worldwide: A comprehensive overview of its diverse applications.* Retrieved from sciencedirect: https://www.sciencedirect.com/science/article/pii/S2773207X24001386

7 Codes

7.1: CNN model code

CNNPdf.pdf

Or

https://colab.research.google.com/drive/1z5U1SQkeZcpMzKo9ojpmrHCHA7wbt5_E?usp=sharing

7.2: XGBoost

XGBoostPDF.pdf

Or

https://colab.research.google.com/drive/18S6hbtB-M8lDw-gT-WLeYeGOifk1pTtX?usp=sharing

7.3: Random Forest Model

RandomForestPDF.pdf

Or

https://colab.research.google.com/drive/15SIwILyd4YHkZAxwpAbJ-76qwQzicM99?usp=sharing