**Reflective Journal**

**L05\_AlishaRush\_ITAI\_1378**

Alisha Rush

Houston Community College

Professor Patricia McManus

ITAI 1378

09/29/2024

**L05\_AlishaRush\_ITAI\_1378**  
**Reflective Journal**

**Introduction**

In this lab, we explored the use of Support Vector Machines (SVM) for image classification using the CIFAR-10 dataset, which contains ten classes of images (airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks). This dataset is well-known for its simplicity and is frequently used in beginner-level image classification tasks due to its moderate computational requirements. The professor provided the subset of the CIFAR-10 data to streamline the processing time, especially for Google Collab, which I used for this lab, as it was the professor's suggestion.

The process helped me understand key steps, including data preparation, model training, and model evaluation, which are essential for working with machine learning models.

**Reflection on Learning**

1. **Support Vector Machine (SVM) in Image Classification**

Before this lab, I had heard of SVM but did not fully grasp its application in image classification. SVM is a supervised learning model used for classification tasks, which works by finding a hyperplane that best separates the data into different classes (Cortes & Vapnik, 1995). In this context, the SVM model uses image features to classify the images into one of the ten categories in the CIFAR-10 dataset. One key aspect I learned is how the SVM's decision boundaries can be adjusted based on the training data, which is why having a well-prepared dataset is crucial for the accuracy of the model.

1. **Data Preparation**

The first part of the process involved installing libraries like NumPy, Matplotlib, and TensorFlow, which are essential for handling the dataset, visualizations, and model development. After installing the libraries, I imported them into my notebook as follows:

# Importing necessary libraries

import numpy as np

import matplotlib.pyplot as plt

from tensorflow.keras.datasets import cifar10

from sklearn.model\_selection import train\_test\_split

from sklearn.svm import SVC

from sklearn.metrics import classification\_report, accuracy\_score

These libraries make it possible to manipulate data arrays, visualize results, and implement machine learning models with fewer lines of code, saving both time and effort.

Next, I loaded the CIFAR-10 dataset with this code:

# Load CIFAR-10 dataset

(X\_train, y\_train), (X\_test, y\_test) = cifar10.load\_data()

# To minimize computational demands lets work with three classes of your choice

# CIFAR-10 classes

class\_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',

'dog', 'frog', 'horse', 'ship', 'truck']

# Choose a subset of classes

chosen\_classes = ['cat', 'dog', 'ship']

class\_indices = [class\_names.index(cls) for cls in chosen\_classes]

Here, the dataset is split into training and testing sets. The training set contains images used to train the model, while the testing set is for evaluating how well the model performs on unseen data. Splitting the data is crucial to prevent overfitting, where a model performs well on the training data but poorly on new, unseen data (Hastie, Tibshirani, & Friedman, 2009).

1. **Flattening, Grayscale Conversion, and Classification**

Before training the SVM, I had to prepare the data further by converting the images to grayscale and flattening them. The grayscale conversion simplifies the images by reducing the color channels from three (RGB) to one, while flattening converts the images into a one-dimensional array, which is necessary for the SVM model, as it requires a flat input.

# Convert images to grayscale

X\_train\_gray = np.dot(X\_train\_subset[...,:3], [0.2989, 0.5870, 0.1140])

X\_test\_gray = np.dot(X\_test\_subset[...,:3], [0.2989, 0.5870, 0.1140])

# Normalize the images

X\_train\_normalized = X\_train\_gray / 255.0

X\_test\_normalized = X\_test\_gray / 255.0

# Flatten the images

X\_train\_flat = X\_train\_normalized.reshape(X\_train\_normalized.shape[0], -1)

X\_test\_flat = X\_test\_normalized.reshape(X\_test\_normalized.shape[0], -1)

Flattening helps reduce the complexity of the data and makes it easier for the SVM model to process each image as a series of pixels rather than a two-dimensional array.

**Model Training**

The core part of this lab was training the SVM classifier. After preparing the data, I used the following code to train the model:

# Train an SVM classifier

model = SVC(kernel='linear')

model.fit(X\_train\_flat, y\_train\_subset.ravel())

The fit function is where the model learns from the data. It was impressive to see how quickly the SVM could classify the grayscale images once the data was properly prepared. As the SVM works by finding a boundary that separates different classes, I could see how the model started making predictions based on these boundaries.

**Model Evaluation**

Once the model was trained, I evaluated its performance using the testing data. The evaluation was done by making predictions and comparing them to the true labels:

# Predict on the test set

y\_pred = model.predict(X\_test\_flat)

# Evaluate the model

print("Accuracy:", accuracy\_score(y\_test\_subset, y\_pred))

print("Classification Report:\n", classification\_report(y\_test\_subset, y\_pred, target\_names=chosen\_classes))

The final accuracy of the model was 0.55. While this accuracy is lower than ideal, it is important to remember that this was a basic implementation without advanced tuning of hyperparameters or feature engineering. The moderate accuracy reflects that the model was able to make correct classifications 55% of the time, which is a decent starting point for such a complex task.

**Challenges and Insights**

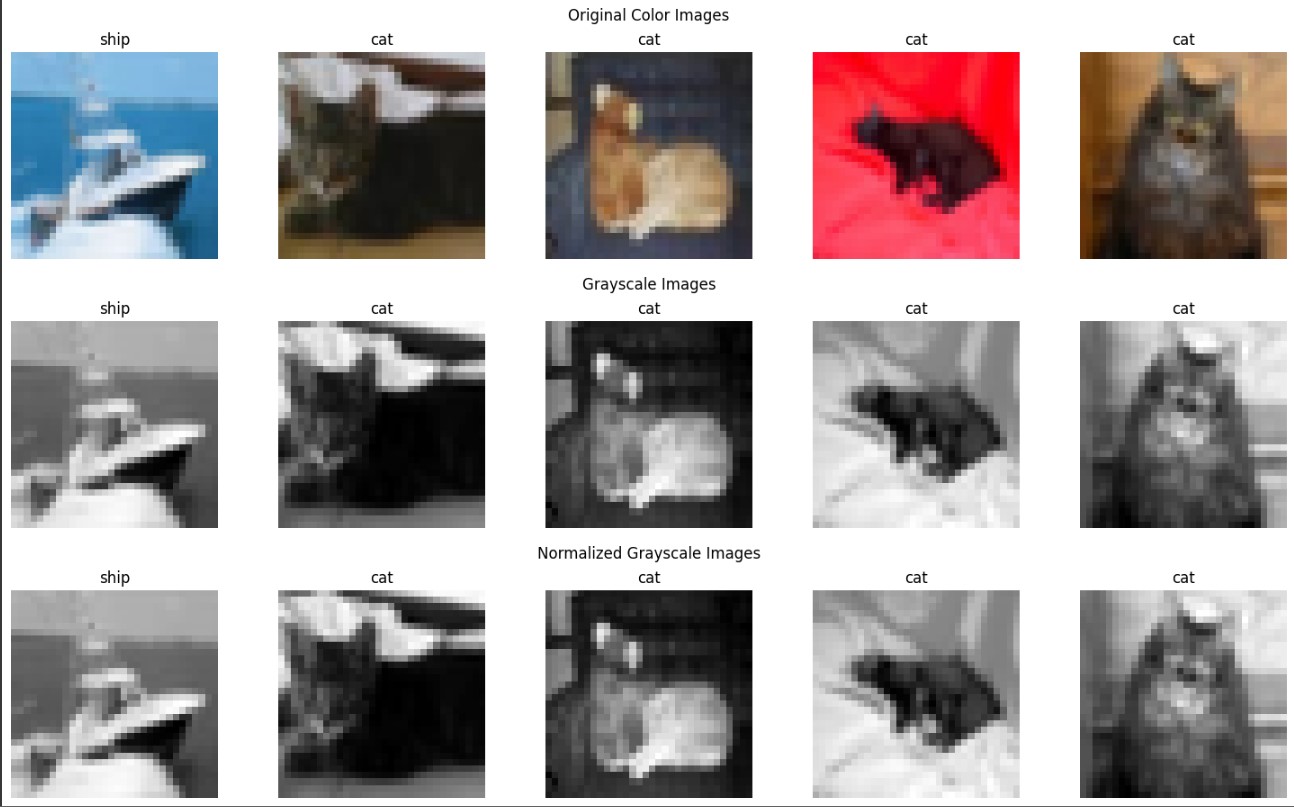
One challenge I encountered was understanding the significance of flattening and converting the images to grayscale. Initially, I thought this might reduce the amount of information the model could use, but after research, I understood that it simplifies the data and speeds up the training process (Russell & Norvig, 2009). Another challenge was waiting for the SVM to process the data, as it is a computationally expensive algorithm. Using Google Collab helped, but I realized that working with larger datasets or more complex models would require more computing power.

**Looking Forward**

Training the SVM model gave me a clearer understanding of how machine learning models work. I anticipate that as I continue learning, I will be responsible for writing more of the code and experimenting with different techniques to improve the model's accuracy. The 0.55 accuracy shows that there is room for improvement, and I’m excited to learn about techniques such as hyperparameter tuning and data augmentation to enhance the model’s performance.

**Conclusion**

This lab on image classification using SVM and the CIFAR-10 dataset was an excellent introduction to machine learning for image data. Through following the instructions provided and reflecting on the process, I gained valuable insights into data preparation, model training, and evaluation. Though I faced some challenges, they ultimately helped me deepen my understanding of the SVM algorithm and how it can be applied to image classification tasks.



**References**

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273- 297.  
Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data*  *mining, inference, and prediction*. Springer Science & Business Media.  
Russell, S., & Norvig, P. (2009). *Artificial Intelligence: A Modern Approach* (3rd ed.). Pearson.