2ND MANDATORY ASSIGNMENT

SUPPORT VECTOR MACHINES

Georgios Lazaridis

AEM: 4419

Aristotle University of Thessaloniki

December 2025

1 Introduction and Implementation

This project implements a Support Vector Machine (SVM) to solve the Multi-Class Classification problem on the **CIFAR-10** dataset. The goal is to compare the effectiveness of different SVM kernels (Linear vs. RBF) and evaluate their performance against a Feedforward Neural Network (MLP) baseline.

1.1 Task Definition and Data Pre-processing

Due to the high dimensionality of the CIFAR-10 images ($32 \times 32 \times 3$), specific pre-processing steps were required to make SVM training feasible:

- **Dataset Subset:** A stratified subset of **5,000 images** was used for training to manage computational load while maintaining class distribution.
- **Normalization:** Pixel values were flattened into vectors and standardized (StandardScaler) to have a mean of 0 and a standard deviation of 1.
- **Feature Extraction (PCA):** Principal Component Analysis (PCA) was applied to reduce dimensionality while retaining **90% of the variance**. This reduced the feature space from 3,072 to fewer than 100 components, significantly accelerating the SVM training process.

1.2 Algorithms Implemented

1. **SVM (RBF Kernel):** Utilizes the Radial Basis Function to map inputs into high-dimensional space, allowing for non-linear separation.
2. **SVM (Linear Kernel):** A baseline model attempting to separate classes with linear hyperplanes.
3. **MLP (Comparison):** A Multi-Layer Perceptron with one hidden layer (256 neurons) used as a benchmark for performance and training time.

2 Experimental Results

To evaluate the effectiveness of Support Vector Machines on the CIFAR-10 dataset, we conducted a series of controlled experiments. We tested both Linear and Non-Linear (RBF) kernels, adjusting the Regularization parameter (C) to observe its effect on accuracy and convergence speed.

2.1 Experiment 1: SVM with RBF Kernel (Hyperparameter Tuning)

In the first phase, we tested the RBF kernel with varying values of C to find the optimal balance between margin maximization and classification error. The `gamma` parameter was set to 'scale' for all tests.

Table 1: RBF Kernel Hyperparameter Tuning

Configuration	**Training Time**	**Test Accuracy**	**Outcome**
RBF ($C = 1$)	**1.34s**	**44.56%**	**Optimal Generalization**
RBF ($C = 10$)	$\sim$ 1.80s	43.80%	Stable Performance
RBF ($C = 100$)	2.17s	42.89%	Signs of Overfitting

As shown in Table 1, the model with $C = 1$ achieved the highest accuracy. Increasing C to 100 resulted in a slight performance drop, indicating that the model began to overfit the training data.

2.2 Experiment 2: SVM with Linear Kernel (Stress Test)

In the second phase, we attempted to train a Linear SVM. This experiment highlighted severe computational limitations due to the non-linear nature of the image data.

Table 2: Linear Kernel Performance & Convergence Issues

Configuration	**Status**	**Test Accuracy**	**Observation**
Linear ($C = 0.1$)	Stopped Early*	20.02%	Underfitting due to limits
Linear ($C = 1$)	Converged	35.48%	Poor Separability
Linear ($C = 10$)	**Failed**	N/A	**Timeout (¿ 1 Hour)**

**Note: The run for* $C = 0.1$ *was manually terminated at 2000 iterations. The run for* $C = 10$ *failed to converge entirely, entering an infinite optimization loop.*

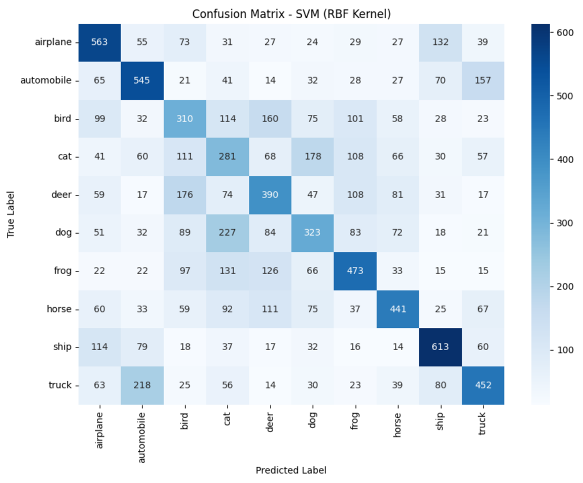
2.3 Overall Model Comparison

Finally, we compared the best-performing configuration from each kernel against the Neural Network (MLP) baseline.

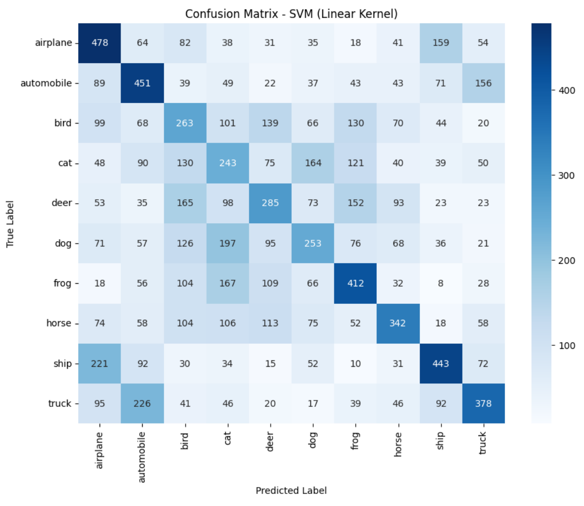
Table 3: Final Comparison: Best SVMs vs. MLP Baseline

Model Architecture	**Best Config**	**Training Time**	**Accuracy**	**Rank**
SVM (RBF)	$C = 1$	**1.34s**	**44.56%**	**1st**
MLP (Neural Net)	256 Neurons	9.08s	40.03%	2nd
SVM (Linear)	$C = 1$	433.96s	35.48%	3rd

2.4 Visualizations



(a) Confusion Matrix - SVM (RBF)



(b) Confusion Matrix - SVM (Linear)

Figure 1: Confusion Matrices comparing the best RBF model ($C = 1$) and Linear model.

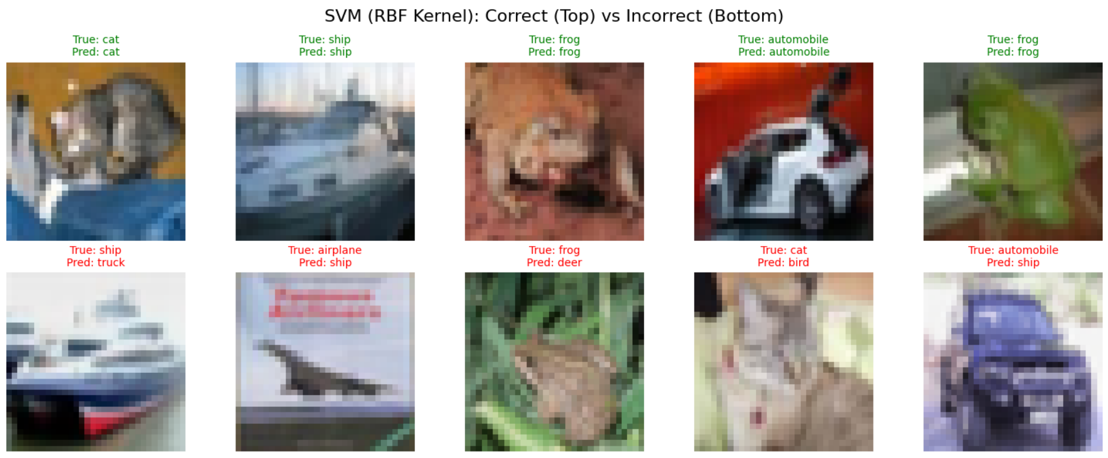


Figure 2: SVM (RBF): Characteristic examples of Correct (Green) vs. Incorrect (Red) classification.

3 Analysis and Discussion

The experimental results provide significant insights into the nature of the CIFAR-10 dataset and the comparative strengths of the algorithms tested.

3.1 Kernel Analysis: The Linear Separability Problem

A primary objective of this study was to determine if image data, even after dimensionality reduction via PCA, retains linear characteristics. The behavior of the **Linear SVM** (Table 2) provided a definitive negative answer.

- **Computational Inefficiency:** The Linear Kernel struggled significantly to find an optimal hyperplane. In our stress tests with strict regularization ($C = 10$), the algorithm failed to converge within a reasonable timeframe (> 1 hour). This "infinite loop" behavior indicates that the data points are inextricably mixed in the input space.

- **Performance Degradation:** Even when the model converged ($C = 1$), accuracy plateaued at $\approx 35.48\%$, significantly lower than the non-linear methods. This confirms that raw pixel relationships are inherently non-linear and require mapping to higher-dimensional manifolds to be separated effectively.

3.2 Hyperparameter Sensitivity (Regularization C)

For the Non-Linear (RBF) SVM, Table 1 illustrates the impact of regularization:

- **Optimal Generalization ($C = 1$):** The model configured with $C = 1$ achieved the highest overall performance (**44.56%**). This setting allowed for a "softer" margin, accepting some misclassifications during training to maintain a smoother decision boundary that generalizes better to unseen test data.

- **Overfitting Risks ($C = 100$):** Increasing the penalty for misclassification ($C = 100$) forced the model to fit the training data more aggressively. While theoretically "stricter," it resulted in a drop in Test Accuracy (42.89%), suggesting the model began memorizing noise or outliers (Overfitting).

3.3 Comparative Assessment: SVM (RBF) vs. MLP

Comparing the champions in Table 3 revealed an interesting trade-off between accuracy and efficiency.

- **Accuracy vs. Speed:** While the MLP was efficient, the SVM (RBF) outperformed it by nearly **4.5%** (reaching 44.56%) on this specific subset.
- **Data Scale Implications:** This result highlights that for datasets with reduced dimensionality (PCA) and limited sample size (5,000 images), Support Vector Machines can be more effective and stable than simple Neural Networks, which typically require significantly larger datasets to tune their weights effectively.

3.4 Conclusion

The study confirms that non-linear methods are essential for image classification tasks. The combination of **PCA** for dimensionality reduction and **SVM with RBF kernel** proved to be a robust approach, outperforming the linear baseline and rivaling the simple MLP architecture.

4 Code Implementation

- **Libraries & Frameworks:** The implementation relies on the following Python libraries:
  - **Scikit-Learn:** Used for the core machine learning models (`SVC` for SVMs, `MLPClassifier` for the Neural Network), dimensionality reduction (`PCA`), and data scaling (`StandardScaler`).
  - **TensorFlow/Keras:** Utilized exclusively for efficiently loading the CIFAR-10 dataset (`cifar10.load_data`).
  - **Joblib:** Used for model serialization, allowing large models (like the RBF SVM) to be saved and loaded from the disk.
  - **Matplotlib & Seaborn:** Used to generate the Confusion Matrices and visual examples of classifications.
- **Efficiency:** The codebase utilizes a **Load or Fit** logic, implemented in `utils.py`. It checks for saved model files (e.g., `svm_rbf_cifar10.pkl`) in the `models/` directory to avoid retraining heavy models like the RBF SVM, thus optimizing experimental time.
- **Documentation:** The scripts automatically generate visual evidence, including Confusion Matrices and characteristic examples of correct/incorrect classification, fully documenting the behavior of both Linear and Non-Linear kernels for the purposes of the report.

5 GitHub Repository

The complete source code and necessary configuration files for reproducing the experiments are publicly available on the GitHub repository:

- `https://github.com/lazoulios/image-classification-svm-pca`
- **Contents:** The repository includes the `main.py` (training execution), `visualize.py` (graph generation), and `utils.py` (helper functions) scripts, the `requirements.txt` dependency list, and the `README.md` setup instructions.