Comparative Analysis of SVM, Decision Trees, and Random Forests

Mohammed Shurrab (40323793), Oleksandr Yasinovskyy (40241188), Huzaifa Mohammed (40242080)

***Abstract*— This work presents multi-class image classification using supervised and semi-supervised models. The focus was on evaluating the performance of traditional classifiers, namely Support Vector Machine (SVM), Random Forest (RF) and Decision Tree (DT). These were trained on handcrafted features extracted from color histograms and Histogram of Oriented Gradients (HOG). Experimentation assessed the impact of key hyperparameters on model accuracy and training time. SVM with an RBF kernel achieved highest classification accuracy, but the RF model showed a trade-off between accuracy and efficiency. The Decision Tree has been a lightweight baseline for comparison.**

***Index Terms*—Machine Learning, Classification, support vector machine (SVM), Decision Tree (DT), Random Forest (RF), semi-supervised Learning.**

I. Introduction

MACHINE Learning (ML) has been widely adopted across numerous fields due to the growing use of large datasets and the significant improvements in computational power over the years. These advancements have created new opportunities and challenges, especially in computer vision, where the task is to accurately classify images based on their semantic content. Supervised machine learning has emerged as a powerful tool for the abundantly available labeled datasets. It has been successfully applied to a range of applications, including object recognition, facial detection, and scene classification. However, generating labeled data at scale remains a costly and time-consuming process. In contrast, unlabeled data are plentiful and often easy to collect, but supervised learning models fail to perform when data are unlabeled.

To address this limitation, semi-supervised learning techniques have gained attention for their ability to leverage both labeled and unlabeled data during model training. These approaches can significantly reduce the reliance on fully annotated datasets while still achieving competitive performance. In this report, the problem of image classification has been explored by investigating different supervised learning models such as support vector machine (SVM) and random forest (RF). Moreover, a semi-supervised model based on a self-training decision tree (DT) has been introduced to explore the impact of having partially labeled data on classification performance. The available dataset consists of RGB images across three distinct classes, namely, *library, museum,* and *shopping mall*. Each class includes 5000 images for training, and 100 images for testing, resulting in a total of 15000 training samples and 300 testing samples. Each image is 256x256 pixels in resolution and stored in a structured directory format that supports automated label extraction.

II. Model Overview

A. Support Vector Machine

Firstly, a support vector machine (SVM) model is used for the given image classification task as a supervised learning model. SVM is suitable for handling high-dimensional data and modeling non-linear decision boundaries with the help of kernel functions. SVM is computationally intensive, but thanks to the feature reduction technique employed, employing SVM has been feasible. A range of hyperparameters have been explored to train the SVM model, where the main components are: 1) the kernel function that includes radial basis function (RBF), linear, and polynomial, and 2) the C value which penalizes the model for misclassifications. Additionally, the degree of the polynomial was also varied to study its effect on model performance, given that it only affects the kernel function 'poly' and it is ignored by the other kernels. After, fine-tuning the hyperparameters of the model, its performance is compared against different supervised and semi-supervised models as will be discussed in the experimental results section.

B. Random Forest

Secondly, the random forest model is used as an ensemble-based learning approach. RF builds multiple decision trees based on different random subsets of the training data available. It then aggregates the output through majority voting, reducing the risk of overfitting that is usually seen with individual trees. The RF classifier was trained during preprocessed feature vectors, resulting in 1800-dimensional feature space. This type of input is adapted for RF, which is efficient in cases of feature redundancy and interactions. RF also provides an adequate balance between accuracy and efficiency. The ability to assess feature importance allows interpretability benefits informing future feature selection. Assessing the performance of RF required configuring a set of hyperparameters, particularly: 1) the number of estimators (trees) in the forest was set to 100 to measure impact on accuracy and overfitting resistance, 2) the maximum depth of the trees was configured to 10 and 3) the splitting criterion was set to entropy to allow the model to choose splits that maximize the information gained. The RF model performance was evaluated through different hyperparameter ranges, exploring their effect on model accuracy, as detailed in the experimental results section.

C. Semi-Supervised Learning – Decision Tree

Sem-supervised learning is ideal when data labels are missing (partially labeled), where the model itself can learn from few labeled data and generalize its knowledge over the entire dataset. This is done by randomly selecting a small portion of the training samples to serve as the initial labeled dataset, whereas the remaining samples are treated as unlabeled. The model is then trained on the small, labeled data and used to predict the labels of the unlabeled instances. A probability value for each prediction is given by the trained model, where it is compared to a probability threshold that evaluates the model's confidence in classifying such instances. Predictions that achieve high confidence are then added to the labeled training dataset. To reduce error propagation, only a limited number of confident samples are added to the labeled set per iteration. This pseudo labeling process is repeated for a fixed number of iterations or until no new confident samples could be added.

Decision Tree (DT) is a machine learning model that works by recursively splitting the input data into subsets based on feature values, thus, forming a tree-like structure. This tree consists of internal nodes, leaf nodes (decision nodes), and edges (branches) connecting the nodes. Each internal node represents a decision on an attribute, each branch corresponds to the outcome of the decision, and each leaf node represents a final class label. DTs are beneficial due to their simplicity and interoperability when compared to other ML models. The hyperparameters essential to train DTs include but not limited to: criterion- a measure of impurities in a node where Gini index and entropy are explored, max depth of the tree, etc. These hyperparameters are fine-tuned to produce the best model. Since DTs are more prone to overfitting, they are particularly useful in low-resource or semi-supervised scenarios.

III. Dataset Preprocessing

The first step for any ML model is to preprocess the dataset to ensure complete and balanced data. Loading the data is straightforward since the dataset is structured into training and testing folders where each contains a folder named by the class label. The ML models explored in this project do not handle 2D inputs, therefore, the images need to be flattened (1D) where each pixel can be considered as a feature. However, this would result in around 200,000 features per image and would require a significant training time for such models to learn. Thus, dimensionality reduction of the images is a crucial stage to improve computational efficiency and ensure feature consistency.

The images were first resized to 64x64 pixels during preprocessing, which resulted in approximately 12,300 features per image. The feature space is still quite large for such models; hence, further preprocessing was performed by extracting features using a hybrid technique combining 1) color histogram and 2) Histogram of Oriented Gradients (HOG). Color histogram is used to capture the distribution of colors in each channel in an image by counting how often each color appears. This is beneficial to capture scene-level cues and distinguish environments by dominant colors. However, color histogram alone is not enough since different classes might have the same color distribution. Therefore, HOG is used to capture the edge and texture structure of the images by computing the orientation of image gradients in localized regions. The final features vector comprises of the concatenation of both color histogram and HOG which results in a further dimensionality reduction to 1800 features, while maintaining the image integrity.

Since the images are already sorted and labeled in organized folders, a dataset load function is introduced to extract the images and save them into their respective variables (training and testing). A label encoder is then used to convert the labels from strings to integers (0: library, 1: museum, 2: shopping mall). Additionally, the reduced feature vector is normalized using StandardScaler to ensure consistency across data samples. Finally, the dataset is shuffled to break correlation between samples. The same dataset with 1800 features per image is used for all models discussed in the subsequent sections.

IV. Model Training and Evaluation

A. SVM

In this work, to assess the efficacy of the proposed approach, some notable computational challenges were faced due to the high dimensionality of the input features, as well as the intensive nature of the kernel-based methods. Managing this restriction requires the evaluation of three different kernels. The use of Radial Basis Function (RBF) kernel, which is well-suited for capturing non-linear relationships throughout the data has shown to have the highest testing accuracy between all the kernels explored (figure 1). Multiple values for the C parameter were tested (C=0.1, 1, 5, 10) to assess trade-offs between margin width and the complexity of the model. Higher values of C led to overfitting and sometimes extremely long wait times that required to cancel the process (more than 5 hours without results). The lower C value (0.1) generalized poorly, as seen in figure 2, where we see the lowest accuracy for the training set. The best compromise was achieved at C=1.0, with a strong testing accuracy (70%) and voided overfitting. Gamma parameter was also tested using fixed values (0.1 and 0.01) and adaptive settings ('scale'). High gamma values (0.1) caused strong overfitting, which resulted in the model predicting a single class. The adaptive 'scale' value for the gamma parameter allowed for the most reliable setting in the balance between accuracy and runtime.

B. RF

The performance of RF is evaluated using metrics such as accuracy, precision, recall, and F1-score. Different hyperparameters were explored where the number of estimators, criterion function (entropy, Gini), maximum tree depth (5-20), minimum samples to split (5-100), and minimum number of leaves (1-50) are varied. The best model with the highest accuracy is the following (n_estimators=100, criterion = 'entropy', max_depth = 10, min_split = 5, min leaf=5). This resulted in a training accuracy of 90% and a testing accuracy of 66%. The comparison between the hyperparameters is shown in Figures 4-6. Figure 4 compares both criterion of entropy and Gini, where the model performance exhibits similar behavior. Figure 5 shows the effect of max tree depth on model performance where as the max tree depth increases, the model is more prone to overfitting. On the other hand, increasing minimum sample to split and minimum number of

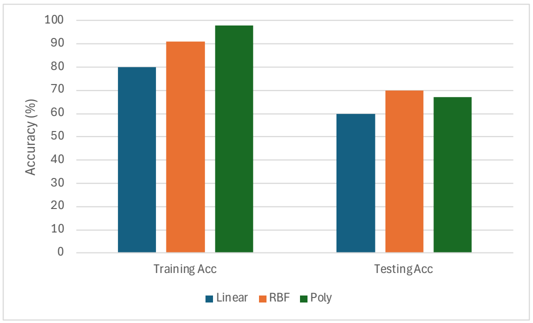


Fig. 1. Accuracy for different kernel function with C=1.

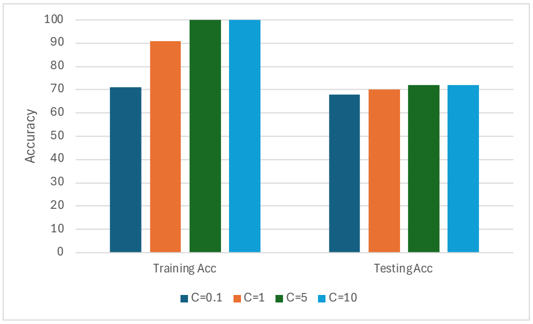
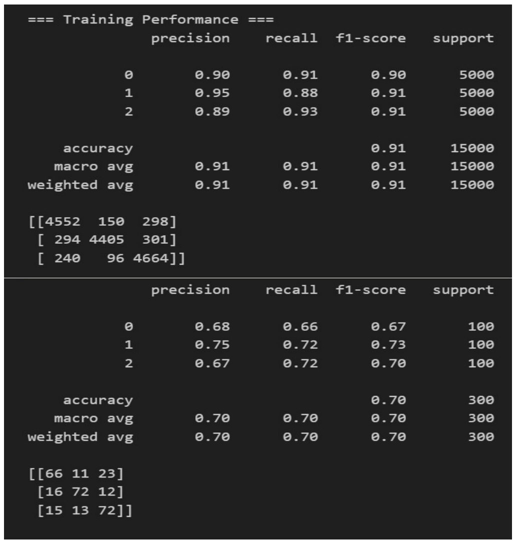


Fig. 2. Accuracy for different kernel function with C=1.



```
=== Training Performance ===
              precision    recall  f1-score   support

           0       0.90      0.91      0.90      5000
           1       0.95      0.88      0.91      5000
           2       0.89      0.93      0.91      5000

    accuracy                           0.91     15000
   macro avg       0.91      0.91      0.91     15000
weighted avg       0.91      0.91      0.91     15000

[[4552  150  298]
 [ 294 4405  301]
 [ 240   96 4664]]
```

```
              precision    recall  f1-score   support

           0       0.68      0.66      0.67       100
           1       0.75      0.72      0.73       100
           2       0.67      0.72      0.70       100

    accuracy                           0.70       300
   macro avg       0.70      0.70      0.70       300
weighted avg       0.70      0.70      0.70       300

[[66 11 23]
 [16 72 12]
 [15 13 72]]
```

Fig. 3. Performance metrics for SVM using RBF, C=1.

leaves, the model training performance deteriorates, resulting in underfitting as shown in Figs. 6 and 7.

C. Semi-supervised DT

The performance of the semi-supervised DT is studied in this subsection. The confidence level to add a new predicted sample is set to 0.9, whereas the number of iterations is set to 25. It is also worth mentioning that only a portion (10% of the total dataset) of the pseudo-labeled data is added to the labeled set. The performance of the semi-supervised model shows signs of underfitting, which may be due to the fact that the model learnt the features (overfitted) of the initial portion of the dataset and failed to generalize on the newer samples that were added at each iteration (misclassified the newly added samples).

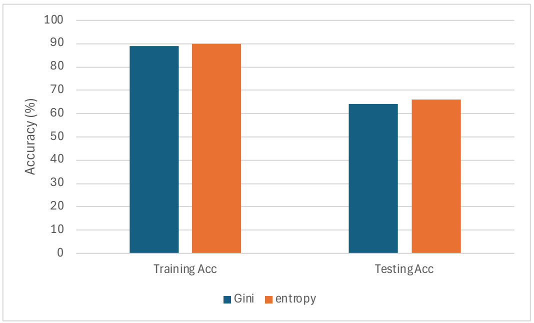


Fig. 4. RF accuracy for different criterions, with 100 estimators, max_depth 10, min_split and min_leaf = 5.

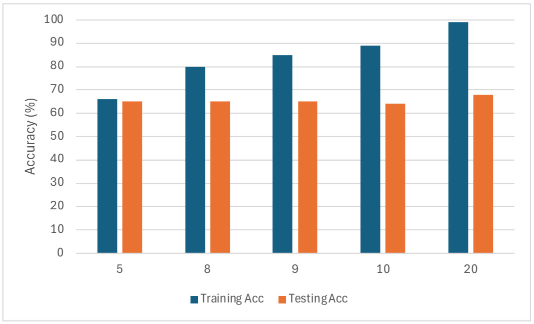


Fig. 5. RF accuracy for different tree depths, with 100 estimators, Gini criterion, min_split and min_leaf = 5.

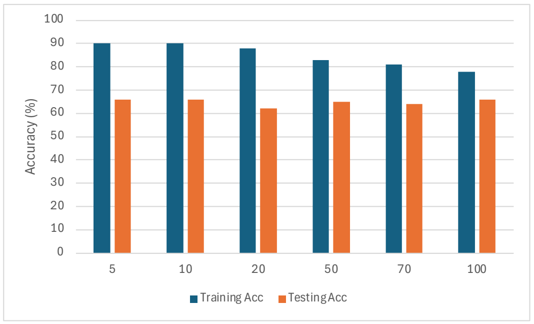


Fig. 6. RF accuracy for different minimum sample splits, with 100 estimators, Gini criterion, max_depth=10 and min_leaf = 5.

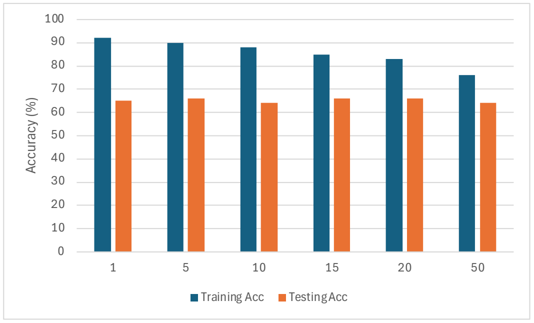
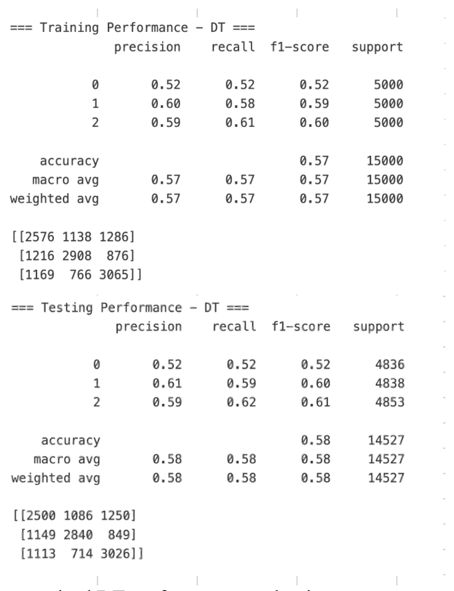


Fig. 7. RF accuracy for different minimum leaf nodes, with 100 estimators, Gini criterion, max_depth=10 min_split= 5.

V. Future Improvements

Future work will focus on implementation of a custom Convolutional Neural Network (CNN). The limitations of traditional classifiers make CNNS a solid choice for improving classification accuracy for visually complex categories. Ultimately, Phase 2 will serve to validate the effectiveness of deep-learning techniques when compared to classical models.



```
=== Training Performance - DT ===
              precision    recall  f1-score   support

           0       0.52      0.52      0.52      5000
           1       0.60      0.58      0.59      5000
           2       0.59      0.61      0.60      5000

    accuracy                           0.57     15000
   macro avg       0.57      0.57      0.57     15000
weighted avg       0.57      0.57      0.57     15000

[[2576 1138 1286]
 [1216 2908  876]
 [1169  766 3065]]

=== Testing Performance - DT ===
              precision    recall  f1-score   support

           0       0.52      0.52      0.52      4836
           1       0.61      0.59      0.60      4838
           2       0.59      0.62      0.61      4853

    accuracy                           0.58     14527
   macro avg       0.58      0.58      0.58     14527
weighted avg       0.58      0.58      0.58     14527

[[2500 1086 1250]
 [1149 2840  849]
 [1113  714 3026]]
```

Fig. 8. Semi-supervised DT performance evaluation.

VI. Conclusion

During this phase, classic supervised and semi-supervised learning approaches were explored for multi-class image classification. Three models were explored, SVM, RF and DT, which were implemented and evaluated through hyperparameter tuning. These experiments provided insights that will guide the development of more powerful, data-driven approach in Phase 2.

References

1- Decision Tree - https://www.ibm.com/think/topics/decision-trees#:~:text=A%20decision%20tree%20is%20a,internal%20nodes%20and%20leaf%20nodes

2- Semi-supervised - https://www.ibm.com/think/topics/semi-supervised-learning#:~:text=Semi%2Dsupervised%20learning%20is%20a,for%20classification%20and%20regression%20tasks

3- Random Forest - https://www.ibm.com/think/topics/random-forest#:~:text=Random%20forest%20is%20a%20commonly,both%20classification%20and%20regression%20problems

4- Python - https://www.python.org/

5- Scikit learn - https://scikit-learn.org/stable/

6- Numpy - https://numpy.org/

7- Skimage - https://scikit-image.org/

8- Matplotlib - https://matplotlib.org/

9- Dataset source: http://places2.csail.mit.edu/download.html