

Final Project Report, Part 2

Prepared For:

Dr. Hichem Frigui

CSE 546, Introduction to Machine Learning

Department of Computer Science and Engineering

University of Louisville

Prepared By:

Jacob Sanders (Supervised Learning - MLP)

Cristian Levya (Supervised Learning – Random Forests)

Nicolas Kosanovic (Unsupervised Learning)

Date: 4 December 2024

## Abstract

Optimized machine learning toolboxes such as Python’s *Sci-Kit Learn* enable rapid prototyping and production of quality machine learning models for a variety of tasks. In this work, an image classifier that uses the outputs of a Convolutional Neural Network (CNN) trained on images from the FLIR Thermal Image Dataset is developed. By using classical estimators on CNN outputs, explainable insights are gained from otherwise incomprehensible neural network features. The Support Vector Machine, Naive-Bayes, Random Forest, and Multi-Layer Perceptron models were used in tandem with normalization and Principal Component Analysis to build a single optimal model. Furthermore, techniques like feature selection, grid-search hyperparameter tuning, and ensemble methods enabled the best developed pipeline (a 50 node, single-layer neural network using 16 principal components) to attain an average score (accuracy, AUC-ROC, F1-measure) on validation-set of 92.97% when classifying images from the FLIR Dataset. The individual metrics from this pipeline and model are accuracy of 89.93%, Area under the Receiver-Operating Characteristic Curve (AUC-ROC) of 98.72%and an F1-measure of 90.25%.

## Introduction

Object detection from images represents an increasingly important and tangible application of machine learning in the contemporary world. Convolutional Neural Networks (CNNs) have shown aptitude in this domain, as their ability to break down images into a plurality of patterns (edges, shapes, colors, etc.) enables them to grasp macroscopic features instead of individual patterns within pixels. Despite these capabilities, CNNs can behave like a *black box,* giving little insight as to which features are being used and for which purpose. By using a separate model pipeline, the individual features of the CNN are given distinct importance, potentially streamlining future classification efforts. Furthermore, techniques can be used to discard unimportant CNN features, hastening runtimes.

This pipeline, of taking CNN outputs and running a custom Machine Learning (ML) model on them, is used on the FLIR Thermal Image Dataset. The dataset contains 5000+ images of bikes, busses, people, cars, and signs taken with IR thermal cameras. The images are greyscaled and vary in height and width. In this work, multiple models are created using prevalent algorithms such as Support Vector Machines, Naive-Bayes, Random Forests, and Multi-Layer Perceptrons to classify CNN feature outputs as one of the five previously mentioned classes.

## Methodology

*Modularity and Consistency*

To ensure consistency between experiments and to avoid repetition of code, a *common* module was created to facilitate this. Within this module, a framework for conducting the experiments was defined. It contains functions for loading and displaying images in a standard way, similar to provided code. To ensure consistency in the initial data between experiments, functions were added to load and consolidate data to be learned from. To ensure consistency in the fitting, scoring, and cross-validation process, a function, *validation\_scores(),* was added to handle the execution of an experiment provided the components: pipeline, param grid(s), X and y training data. The function executes the experiments three times using three different scoring methods: accuracy, roc\_auc, and f1\_score, returning the complete results for further analysis. At every point, random states were set explicitly to, again, provide consistency throughout the experimental process and between researchers.

*Unsupervised Learning*

Prior to generating models that will map inputs to outputs, it is often useful to gain a deeper understanding of the dataset. Regrettably, visualizing raw data is impossible when a dataset contains more than three features. To this end, two techniques, namely Principal Component Analysis (PCA) and Feature Selection can be used to *compress* the dataset (either for visualization or accelerated training). The former analyzes covariance among the features to formulate a special transformation which retains significant amounts of variability among samples, while the latter seeks to capture an optimal variance to number-of-features ratio in the original dataset without explicit transformation. PCA is particularly useful for transforming the high-dimensional dataset to two or three principal components that can be visualized. In contrast, Feature Selection is accomplished via a Variance Thresholding with a threshold value of 0.2 or using SelectKBest to identify the 8 features with the most variance.

Outside of dataset compression, Unsupervised Learning is also used for *clustering* to identify critical characteristics of the dataset, including patterns similar (or differing) from the rest of the data. With such knowledge, one can forecast which samples may be challenging for models, especially for those with probabilistic outputs. Thus, two clustering algorithms are employed: K-Means Clustering and TSNE. The former is commonly used but has the drawbacks of struggling with non-convex data distributions and requiring a user-specified number of clusters. In contrast, TSNE is fully autonomous, requiring no hyperparameter specification.

*Supervised Learning*

After setting up common code that we would use in all our models, we got to work setting up our pipelines for each model. We started with SVM and decision trees for this report. We decided upon these because of their benefits, namely SVM’s plentitude of regularization techniques that would improve our accuracy as well as working well with larger datasets. Decision trees were a good choice too for their ensemble methods of Random Forests and Gradient Boosting, as well as how quickly they can be trained. Decision trees are also very clear to interpret.

Starting with decision trees, we set up two pipelines for now, both with Random Forest ensemble, but one with Standard Scaling and the other with MinMax Scaling. They would both perform a grid search on the parameter of max depth for the trees, ranging from 1 to 5.

*Supervised Learning: Support Vector Machine Classifier (SVM)*

In contrast,

|  |  |  |
| --- | --- | --- |
| **Step** | **Option 1** | **Option 2** |
| 1. **Scaler** | StandardScaler() | MinMaxScaler() |
| 1. **PCA** | PCA(n\_components=8) | PCA(n\_components=16) |
| 1. **Feature Selection** | SelectKBest(k=8) | VarianceThreshold(threshold=0.2) |

Table 1: Preprocessing step options explored in grid searches for MLP Classifier

|  |  |  |
| --- | --- | --- |
| **SVM Parameter** | **Option 1** | **Option 2** |
| **Regularization C** | 10.0 | (50,50) |
| **Kernel Type** | RBF | tanh |
| **Kernel Gamma** | 0.001 | sgd |

*Supervised Learning: Naïve Bayes (NB)*

*Supervised Learning: Multilayer Perceptron (MLP)*

The next experiment considered the Multilayer Perceptron (MLP) classifier, attempting to identify suitable preprocessing configurations and model parameters. First, the pipeline was defined to ensure a consistent and reasonable order of operations for each of grid-based fits to follow. The following order was chosen for preprocessing steps, which was then followed with model definition for fitting:

1. Data Normalization
2. Dimensionality Reduction
3. Feature Selection

Once the pipeline was defined, grid search was constructed with reasonable initial preprocessing methods and parameters, as well as model parameters. At this stage the values were not expected to be optimal. The choices were primarily chosen to provide information to narrow the search area for optimal parameters later and to be explorable in a reasonable amount of time.

|  |  |  |
| --- | --- | --- |
| **Step** | **Option 1** | **Option 2** |
| 1. **Scaler** | StandardScaler() | MinMaxScaler() |
| 1. **PCA** | PCA(n\_components=8) | PCA(n\_components=16) |
| 1. **Feature Selection** | SelectKBest(k=8) | VarianceThreshold(threshold=0.2) |

Table 3: Preprocessing step options explored in grid searches for MLP Classifier

|  |  |  |
| --- | --- | --- |
| **MLP Parameter** | **Option 1** | **Option 2** |
| **MLP Hidden Layer Sizes** | (50,) | (50,50) |
| **MLP Activation** | relu | tanh |
| **MLP Solver** | adam | sgd |
| **MLP Alpha** | 0.0001 | 0.01 |

Table 4: MLP Classifier parameter options explored in grid searches for MLP Classifier

Experiments using combinations of the options detailed above in Table 1 and Table 2 were executed using the *validation\_scores()* method from the common module*.* This yielded 385 results for each combination of parameters and scorer method. These results were combined by averaging the scores and standard deviation of scores for each three tests for each combination. This resulted in 128 results which for brevity, only the top 10 are shown below in Table 3.

## Results

*Unsupervised Learning*

Data compression and initial visualization is done via PCA. Figure 1 depicts a plot showing how explained variance changes as the number of components increases. Little improvement for additional components is observed beyond analyzing the first 5 principal components. Moreover, Figure 1 also shows minute differences in explained variance among the three different dataset transformations applied, namely Unscaled Data (US), Standard Scaled Data (SS), and MinMax Scaled Data (MM).

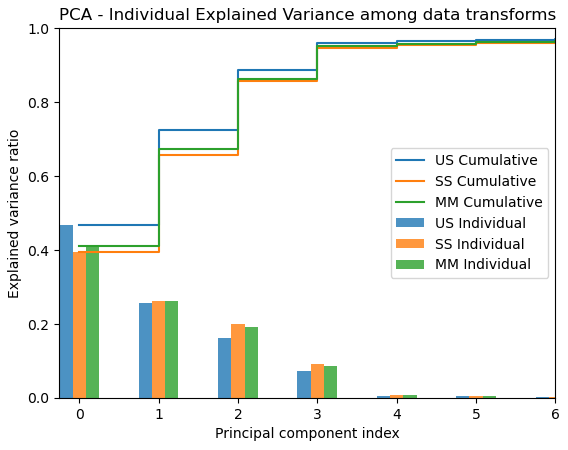


Figure 2: Principal Component Analysis chart showing differences in individual and cumulative explained variances for all three scaled models using between 1 and 7 principal components.

Regardless of the selected number of components, Figure 1 indicates that over 60% of the explained variance exists within the first two principal components. Thus, visualization of the effects of such dimensionality reduction is shown in Figure 2, where the plot on the left represents all training samples from the US, the center is SS, and the rightmost plot is MM. Colors are determined by the corresponding samples’ output class.

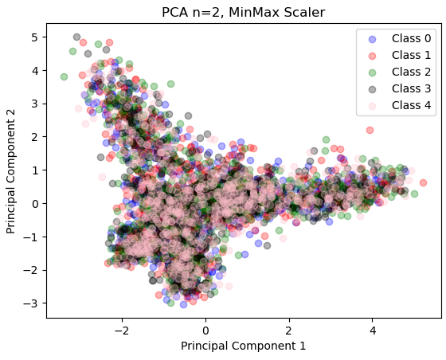
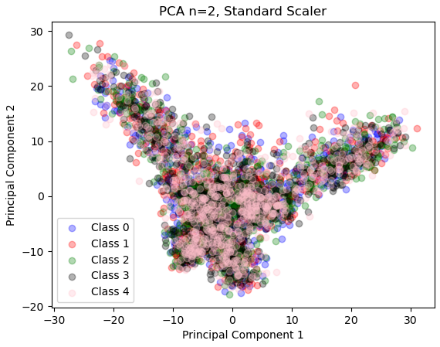
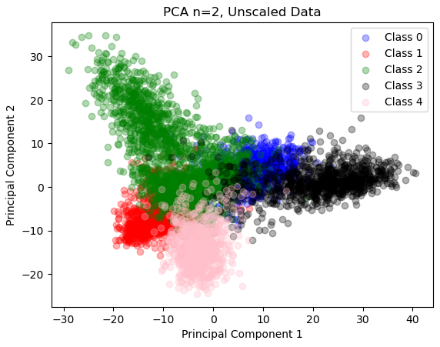


Figure 3: Visualization of PCA with dataset labels for all three scaled datasets.

TSNE was also used for visualization of potential clusters within the transformed datasets, as PCA showed little useful information in scaled configurations.

A diagram of a dataset

Description automatically generated with medium confidenceA diagram of a number of datasets

Description automatically generatedA diagram of a number of data set

Description automatically generated with medium confidence

Figure 4: TSNE visualization of all scaled datasets.

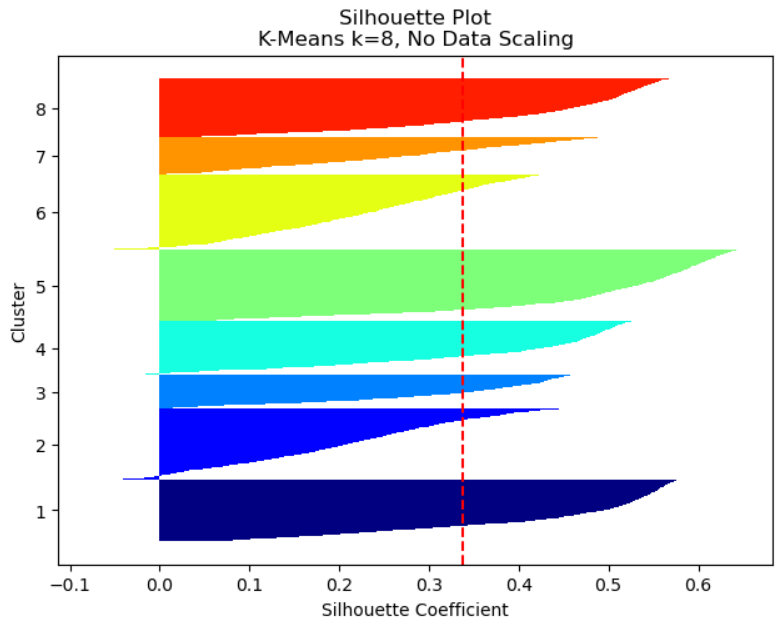
A graph with a line

Description automatically generated

A graph with a red line

Description automatically generated

Outside of pure exploratory visualization, Silhouette Plots were used to identify silhouette coefficients corresponding with the datasets to evaluate clustering capabilities.

A chart of different colors

Description automatically generatedA chart of different colored lines

Description automatically generated

Figure 5: Silhouette plots of all scaled datasets for KNN with 8 neighbors each.

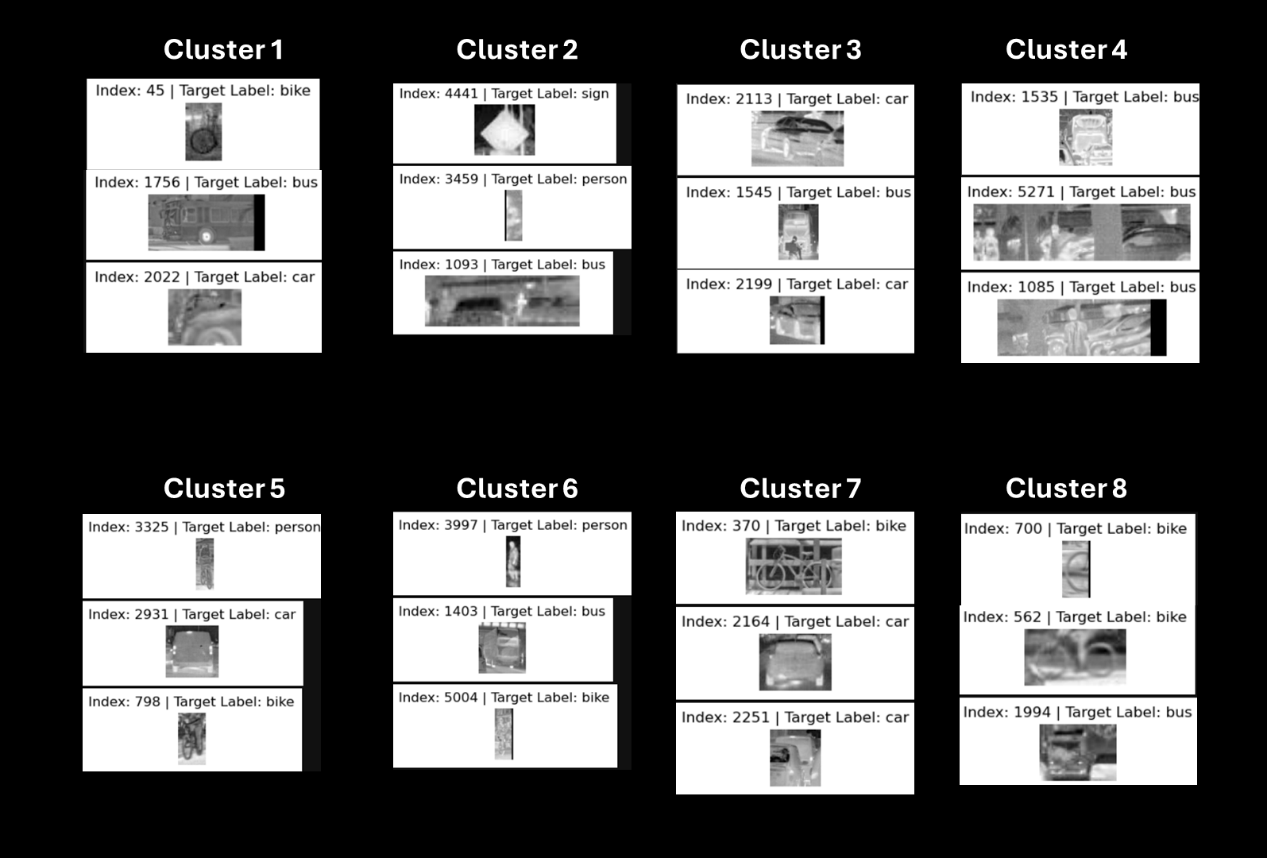


Figure 6: Silhouette plot cluster core samples. These samples agree most with the KMeans clusters.

*Supervised Learning*

Sample results from select Decision Tree pipeline experiments are shown in Figures 7 and 8. This highlights differences in accuracy, mean fit time, AUC-ROC, and the f1 score.

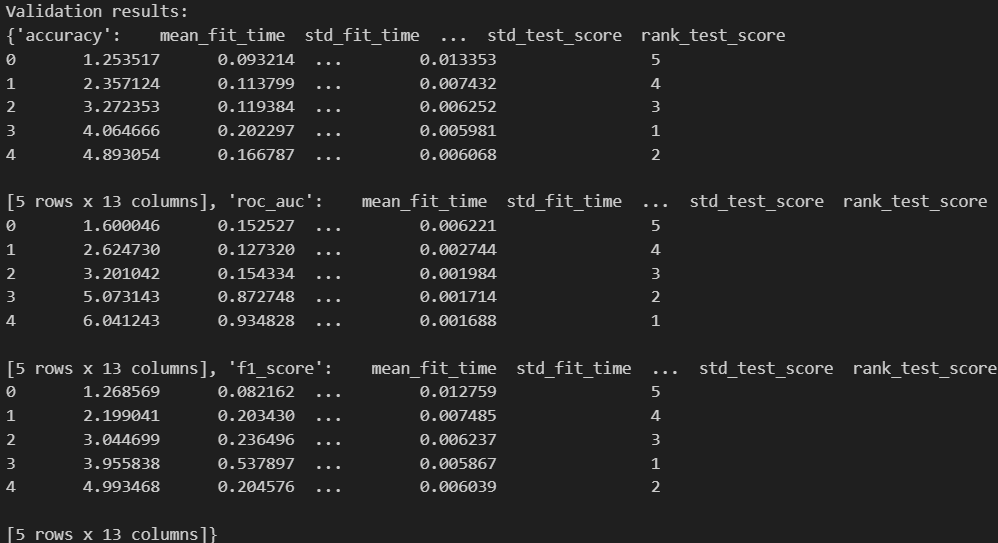


Figure 7: Standard Scaling, Random Forest Ensemble Results

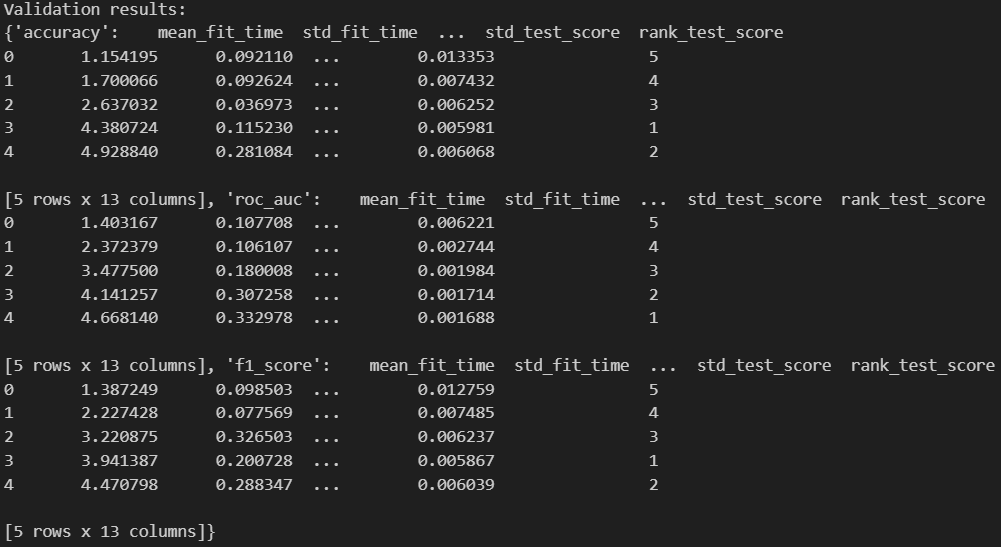


Figure 8: MinMax Scaling, Random Forest Ensemble Results

*Supervised Learning: MLP Classifier*

Overall, MLP performed fairly well for most all combinations of parameters, the lowest average score of all three scoring methods was 88.50%. Because of the volume of data (128 combinations), only the top 10 average scoring models are shown in the Table 3 below.



Table 3: MLP Classifier – Top 10 Performing Model Options, Avg. Mean Scores, and Avg. Std. Of Scores

## Analysis and Discussion

*Unsupervised Learning*

As shown in Figure 1, a significant portion of the dataset is described by the first half dozen principal components, meaning significant savings in computational time are available via dimensionality reduction. Initial tests with much higher numbers of components (n=8 and 16, respectively) displayed tolerable results in terms of computational speed. That said, additional training and inference time savings can be achieved using even fewer components (down to about n=6). Pushing it further can become exceedingly problematic, as evidenced by the chaotic nature of the scaled data shown in Figure 4. Moreover, Figure 5 reinforces this concept by displaying the TSNE plots of all datasets. The messy and incomprehensible overlap of all the classes indicates that all of the datasets require at least 3 dimensions to begin being intelligently clusterable.

K-Means clustering is visualized in Figure 6 via silhouette plots. The prevalence of a mean silhouette coefficient value of >0.3 indicates that with 8 clusters, the clustering model was rather confident in its distinctions.

*Supervised Learning*

For our decision tree models, they were able to run on the entire dataset in not too much time, with both taking around three and a half minutes. All three scoring methods (accuracy, F1, and AUC of ROC) follow the same trend of scoring better with more depth but leveling off. We can also see as max depth increases that so does the mean fit time. The standard deviation of test scores also lowered as the max depth increased, as expected because deeper trees capture more data and patterns at the cost of time. Both F1 and accuracy scores were highest at max depth of 4 and for AUC of ROC, a max depth of 4 was second only to 5, thus this seemed to be our best depth for sure.

*Supervised Learning: MLP Classifier*

The top results from the grouped averages of scores highlight the optimal parameter combinations that yield the highest performance across different scoring metrics. For example, the parameter set using *SelectKBest(k=4)*, *relu* activation, *adam* solver, *PCA(n\_components=8)*, and *MinMaxScaler()* consistently achieves high average test scores across all splits, with an average mean test score exceeding 0.92. This suggests that this combination balances feature selection, dimensionality reduction, and scaling effectively to enhance the neural network's performance. The low standard deviation in test scores for these top results also indicates consistency and robustness across different data splits.

*Supervised Learning: SVM*

A close-up of a bicycle

Description automatically generatedA blurry image of a person's head

Description automatically generatedA person riding a bicycle

Description automatically generated

## Conclusion & Future Work

Following these preliminary results, the path is paved ahead for what needs to be done in the future. For instance, further evaluation of decision trees can be done with Gradient-Boosted Ensemble Methods. Early testing of such models was terminated due to their long training times (1+ hour). Additionally, the number of estimators within such ensemble methods requires tuning.

Continued refinement of MLP classifier will be conducted to home in on the optimal range somewhat revealed during these experiments. Outside of what was shown here, not much experimentation was done with the Naïve Bayes model. Given the relative closeness of some forms of the transformed dataset (e.g. Figure 4, unscaled data), Naïve Bayes could prove fruitful for its simplicity and speed.

Finally, the true ensemble methods of Bagging and AdaBoost will be used to hybridize the models to attain optimal model performance once individual models are performing well.