HSKL: A MACHINE LEARNING FRAMEWORK FOR HYPERSPECTRAL IMAGE ANALYSIS

Qian Cao[1], Deependra Mishra[1], John Wang[1], Steven Wang[1], Helena Hurbon[1], Mikhail Y. Berezin[1]*

Department of Radiology, Washington University School of Medicine, 4515 McKinley Avenue, St. Louis, MO 63110, USA
Corresponding author: berezinm@wustl.edu

ABSTRACT

A new framework for advanced machine learning-based analysis of hyperspectral datasets HSKL was built using the well-known package *scikit-learn.* In this paper, we describe HSKL's structure and basic usage. We also showcase the diversity of models supported by the package by applying 17 classification algorithms and measure their baseline performance in segmenting objects with highly similar spectral properties.

Index Terms— Hyperspectral Imaging, Machine Learning, Image classification

1. INTRODUCTION

High-quality classification and regression models are fundamental to the most workflows in hyperspectral image (HSI) analysis. The typical tasks range from classification of skin lesions in medical diagnostics [1], to detecting oil spills in remote sensing [2]. There is a need for a greater selection of data-driven machine learning algorithms for the rapidly increasing type and volume of hyperspectral data. Numerous commonly used statistics-based algorithms for denoising, dimensional reduction, clustering, and classification have been adapted for HSI analysis. However, their capabilities are not sufficient to distinguish subtle differences in the spectral - spatial domains. While deep learning approaches have been developed in HSI [3] they are mostly limited by availability of large standardized and labeled datasets for training and testing. Moreover, the typical researcher does not have the access to data at the scale where deep learning models can be efficiently trained. An intermediate approach that provides fast satisfactory results from relatively small HSI datasets is highly appreciated.

In this work, we present a software package, hyperspectral-scikit-learn (hereon referred to as HSKL) for relatively quick and convenient identification of objects in hyperspectral datasets. HSKL aims to provide researchers and engineers in hyperspectral imaging with an easy-to-use interface to a family of machine learning algorithms in the popular python package scikit-learn. Scikit-learn is a well-known versatile library of established and validated algorithms with a consistent interface and a rapidly growing number of classifications, regression, and clustering models [4]. In comparison to previous rather segmented approaches, algorithms in scikit-learn provide baseline performance that enables researchers to carry out classification or regression tasks with limited data and minimal labeling. Importantly for hyperspectral data analysis, different estimators and transformations can be combined using the straightforward pipeline application programming interface for training composite or ensemble models.

Despite the advantages of scikit-learn for data analysis, the direct processing of hyperspectral datasets with this package is time-consuming requiring careful transformations of the image, labels, and masks. Even though user can always apply scikit-learn directly to process hyperspectral data, the workflow of transforming the images to a scikit-learn compatible format can be highly error prone because of the many manual steps involved. HSKL streamlines this process through i) handling direct transformation of hyperspectral image input to scikit-learn-compatible matrices, and ii) providing availability of common and emerging utilities to pre-process hyperspectral images. This approach establishes a flexible yet standardized workflow to data analysis and reduces boilerplate code required to perform machine learning studies on hyperspectral data. With a few lines of code, HSKL users can train and use models, which take hyperspectral images as input and obtain high quality pixel-wise or image-level labels, thus achieving the goal of many hyperspectral data analysis.

Herein, we present an introduction to the HSKL software package, apply a collection of classification algorithms to an example hyperspectral dataset acquired on a bench-top hyperspectral system, and compare baseline model performance.

2. HSKL DESCRIPTION

The basic workflow for HSKL is shown in **Figure 1**. The workflow included the following elements. *Preprocessing:* Though algorithms in scikit-learn do not explicitly require preprocessing, normalization, dimensional reduction and other standard steps, they are often necessary to achieve a satisfactory performance. For example, dimensional

reduction techniques such as principal component analysis (PCA) and endmember decomposition might serve to remove noisy channels and decrease the amount of data to the most essential components. *Load images and labels:* With the exception of unsupervised clustering, most algorithms in HSKL requires some labeling of the training data. The package allows for convenient input of pixel-level labeling and masking, accelerating model development workflow. *Estimator fit*: This function implements model training for most estimators in scikit-learn using training hyperspectral images and labels. This includes but are not limited to Classification, Regression, and Clustering algorithms. The outcome of this module is the trained model that can be applied for inference. *Estimator predict*: The trained model from the fit stage is used for predicting and assigning labels for previously unseen data.

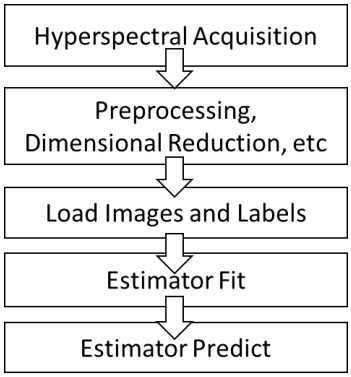

Figure 1. Overview of features in HSKL and relation to hyperspectral image analysis workflow.

HSI-Learn is designed to work with NumPy arrays, with a syntax fully compatible with that of scikit-learn. The following listing shows the minimal code required to train a classifier model. The desired method name can be passed to initialize the model, with the default option being Random Forest. As consistent with scikit-learn, the classifier features fit() and predict() methods to train the model and apply inference, respectively.

```
import hskl.classification as classification

# Load training, testing, and label images
train, test, label = ...

# Train a classifier and predict test image
cl = classification.HyperspectralClassifier(
        method_name="RandomForest")
cl.fit(train, label)
prediction = cl.predict(test)
```

Listing 1. Basic usage for `HSKL`.

Figure 2 shows the structure of the codebase. The *BaseEstimator*, *ClassifierMixin*, and *RegressorMixin* classes are all belong to scikit-learn. The *BaseEstimator* class implements basic setter and getter methods for the model parameters; *the ClassifierMixin* and *RegressorMixin* classes implements accuracy scoring functions. Unlike scikit-learn estimators, the `fit` and `predict` methods are implemented in the parent class *HyperspectralEstimator*, which handles transformations and masking of hyperspectral images. Additionally, a utilities module (`hskl.utils`) is provided to generate overlay visualization of labeled images, spectral normalization, and dimensional reduction using PCA.

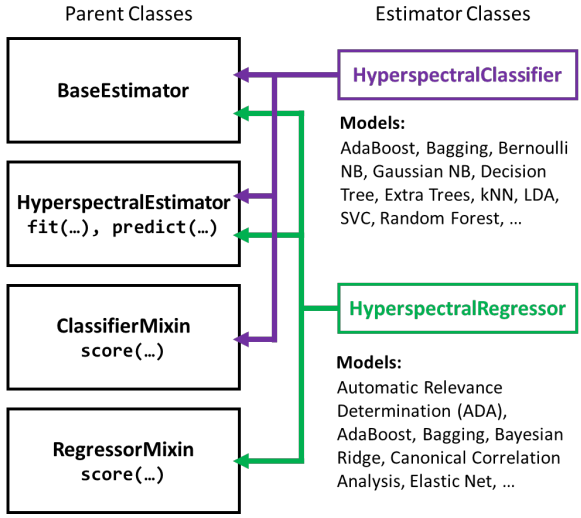

Figure 2. Visualization of the HSKL codebase as a class inheritance diagram. Shown also is a non-exhaustive list of models supported by `HyperspectralClassifier` and `HyperspectralRegressor`.

3. APPLICATION OF HSKL

The images were collected using and HSI imaging system in shortwave-infrared (SWIR). The imaging HSI pushbroom data system featured an SWIR sensitive 2D InGaAs thermoelectrically cooled CCD camera (Ninox, Raptor), a 25 mm focal length SWIR lens (StingRay Optics), an imaging spectrograph Imspector N17E (Specim), and a linear, PC-controlled movable stage (Middleton Inc.). These components were integrated by Middleton Inc. into a stand-alone image acquisition system. The system provided negligible chromatic aberration in the range of 600-1600 nm. Two conventional incandescent 2x50 W halogen lamps with broad output from 400 to 2500 nm were used as the light sources.

The data were processed for spectral analysis, PCA and endmembers using a hyperspectral imaging software IDCube (HSpeQ LLC). The number of the endmembers was performed by the noise-whitened Harsanyi–Farrand–Chang (NWHFC) method. After the extraction procedures, all endmembers were presented through the corresponding abundance maps using N-finder [5]. A map for each

endmember was visually evaluated and the best combinations of two endmembers corresponding to the objects were used. Four types of objects: polymeric round dish, centrifuge tubes, a coin, and a plastic wrench (**Figure 3)** were scanned with HSI-SWIR as shown in **Figure 4A**. The dataset is considered challenging to segment because the material composition of the centrifuge tube and the plastic wrench are similar, although colored with a different pigment. These colors are visible to human eyes and conventional visible cameras with Si-based sensors (blue vs. green) (**Figure 3**), but not visible in SWIR because the pigments are not optically active (do not absorb photons) beyond 1,000 nm. Because of the lack of spectral difference (**Figure 4B)**, conventional methods, such as PCA (**Figure 4C)** or endmember analysis (**Figure 4D)** were unable to distinguish between the two.

Figure 3 Objects in a dataset with the visible camera.

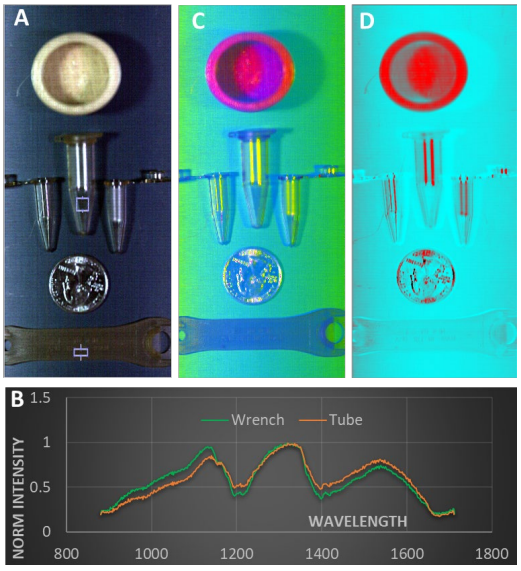

Figure 4 Example image set. A: Pseudo RGB image using image of training data made in SWIR 1094 nm (red), 1301 (green), 1475 nm (blue). B: Normalized spectra of the wrench and the tube from the selected region of interests. Spectral correlation is 0.93; **C**: PCA in pseudo RGB: component 1 (red), component 2 (green), component 3 (blue). **D:** Endmembers in pseudo RGB (see text).

Two HSI datasets were acquired, one for model training and one for testing. Both datasets were segmented manually using MATLAB's Image Labeler. For preprocessing, power normalization in the spectral domain was applied to both images according to:

$$I_n(x,y,\lambda) = \frac{I(x,y,\lambda)}{\int_{\lambda_0}^{\lambda_1}[I(x,y,\lambda)]^2 ds}$$

where I is the acquired image; I_n is the normalized image; x, y and λ are the spatial and spectral dimensions, respectively. The support of captured spectra is from λ_0 to λ_1.

After normalization, PCA was applied to the training image, reducing the spectral dimension from 510 channels to 19 channels, which contain 80% of the data variance. The eigenvectors were then used to transform the testing image into the same space, also with 19 channels.

Model training and testing were carried out using `hsk1`. Seventeen models were trained using different classification methods. These are: AdaBoost [6], Bagging [7], Bernoulli and Gaussian Naïve Bayes (NB) [8] , Decision Tree [9], Extra Trees [10], Random Forest [11], Gradient Boosting [12], Linear and Quadratic Discriminant Analysis (LDA and QDA) [13], Support Vector Classifier (SVC) [14], Logistic Regression, Multi-layer Perception (MLP) [15], Ridge Classifier [16], and Stochastic Gradient Descent [17]. Since we were evaluating the baseline performance, all models were trained with default parameters as defined in scikit-learn, with no further hyperparameter tuning.

4. RESULTS

The results summarized in **Figure 5** shows the model-predicted labels for pixels in the test set. As expected, in most cases, error from misclassification of the centrifuge tubes was most severe. However, the majority of the classifier models were able to separate the two objects based on spectral characteristics.

The predicted labels were compared with ground truth labels in the manual segmentation. The precision, recall, and F-score were used to assess model testing performance. For each object, the metrics were computed in a binary one-vs-rest fashion (e.g., wrench vs. non-wrench, coin vs. non-coin). Precision, recall and F-scores for all objects and classification methods are shown in **Figure 6**. Almost all models resulted in good predictions for the plastic wrench, coin, and rubber cap, with precision, recall, and F-scores of ≥0.8. The sole exception is the AdaBoost classification of the coin, in which some pixels are misclassified as background and vice versa.

Pixels belonging to the centrifuge tubes are more difficult to classify. This is mostly due to its similarity with the plastic wrench in terms of material composition, both appeared to be made from polyethylene terephthalate. Spectral correlation

between the wrench and the tubes was found to be 0.93 indicating high similarity between the two subjects. For comparison, the spectral correlation between the wrench and the dish was much lower 0.86 facilitating their segmentation. Another factor is that both materials tubes are semi-transparent. Therefore, some spectral mixing with the anodized aluminum tabletop background is present. The anodized aluminum has low reflection in the visible spectral range but reflects strongly in SWIR. In addition, the surface of the centrifuge tube is highly reflective, leading to some misclassification with the coin. Despite these challenges, the Extra Trees, Gradient Boosting, Random Forest, SVC, and somewhat kNN algorithms were still able to classify most of the pixels belonging to the centrifuge tube, achieving an F-score > 0.75.

5. CONCLUSIONS

In this work, we showcase a software package HSKL for the application of general-purpose machine learning algorithms to hyperspectral images. We tested the 17 classification algorithms on an example hyperspectral dataset acquired from a bench scanner. Even without parameter tuning, the baseline performance of most algorithms achieved satisfactory pixel-wise classification of the objects in the test image. The package available on GitHub and installable via the Python Package Index (PyPi). HSKL supports a suite of image classification and regression algorithms, with more features are planned, such as support for clustering and robust hyperparameter tuning.

6. ACKNOWLEDGEMENT

The team acknowledges funding from NSF 1827656 (MB), NSF 1355406 (MB), NIH R01 CA208623 and Mallinckrodt Institute of Radiology at Washington University (HH, QC). We also thank Optical Spectroscopy Core Facility at Washington University funded through NIH award 1S10RR03162.

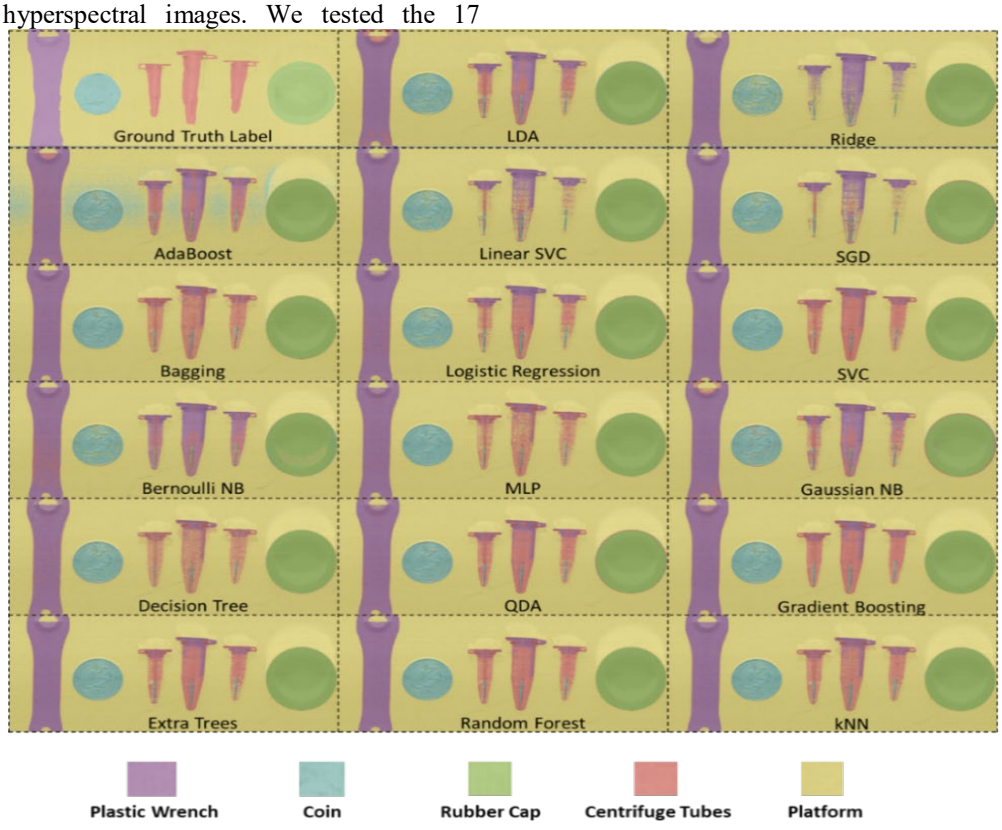

Figure 5 Ground truth label and model-predicted labels based on pixel spectra. The color encodings are purple-plastic wrench, blue-coin, red-centrifuge tube, and green-rubber cap. Extra trees, Gradient Boosting, Random Forest, SVC support vector, and kNN methods demonstrate the best visual differentiation between the plastic wrench and the tubes.

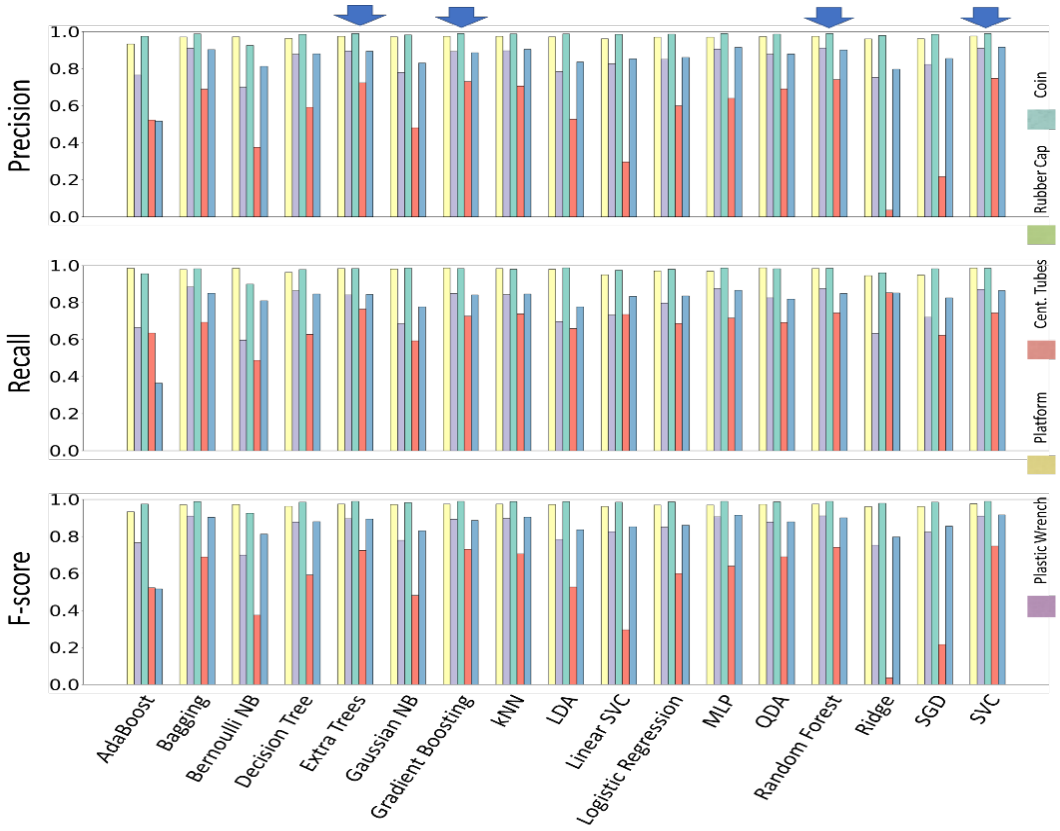

Figure 6 Precision, recall and F-score for all methods and objects, using default classifier parameters. Each color encodes a pixel type. Arrows show the classification algorithms with the highest scores for all objects.

6. REFERENCES

[1] Du, T., Mishra, D.K., Shmuylovich, L., Yu, A., Hurbon, H., Wang, S.T., and Berezin, M.Y.: 'Hyperspectral imaging and characterization of allergic contact dermatitis in the short-wave infrared', J Biophotonics, 2020, 13, (9), pp. e202000040

[2] Yang, J., Wan, J., Ma, Y., Zhang, J., and Hu, Y.: 'Characterization analysis and identification of common marine oil spill types using hyperspectral remote sensing', International Journal of Remote Sensing, 2020, 41, (18), pp. 7163-7185

[3] Paoletti, M.E., Haut, J.M., Plaza, J., and Plaza, A.: 'Deep learning classifiers for hyperspectral imaging: A review', ISPRS Journal of Photogrammetry and Remote Sensing, 2019, 158, pp. 279-317

[4] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., and Dubourg, V.: 'Scikit-learn: Machine learning in Python', the Journal of machine Learning research, 2011, 12, pp. 2825-2830

[5] Winter, M.E.: 'N-FINDR: An algorithm for fast autonomous spectral end-member determination in hyperspectral data', in Editor (Ed.)^(Eds.): 'Book N-FINDR: An algorithm for fast autonomous spectral end-member determination in hyperspectral data' (International Society for Optics and Photonics, 1999, edn.), pp. 266-275

[6] Freund, Y., and Schapire, R.E.: 'A decision-theoretic generalization of on-line learning and an application to boosting', Journal of computer and system sciences, 1997, 55, (1), pp. 119-139

[7] Breiman, L.: 'Bagging predictors', Machine learning, 1996, 24, (2), pp. 123-140

[8] Harry, Z.: 'The optimality of naive bayes', in Editor (Ed.)^(Eds.): 'Book The optimality of naive bayes' (2004, edn.), pp.

[9] Breiman, L., Friedman, J., Stone, C.J., and Olshen, R.A.: 'Classification and regression trees' (CRC press, 1984. 1984)

[10] Geurts, P., Ernst, D., and Wehenkel, L.: 'Extremely randomized trees', Machine learning, 2006, 63, (1), pp. 3-42

[11] Breiman, L.: 'Random forests', Machine learning, 2001, 45, (1), pp. 5-32

[12] Friedman, J.H.: 'Greedy function approximation: a gradient boosting machine', Annals of statistics, 2001, pp. 1189-1232

[13] Hastie, T., Tibshirani, R., and Friedman, J.: 'The elements of statistical learning: data mining, inference, and prediction' (Springer Science & Business Media, 2009. 2009)

[14] Chang, C.-C., and Lin, C.-J.: 'LIBSVM: a library for support vector machines', ACM transactions on intelligent systems and technology (TIST), 2011, 2, (3), pp. 1-27

[15] Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J.: 'LIBLINEAR: A library for large linear classification', the Journal of machine Learning research, 2008, 9, pp. 1871-1874

[16] Rifkin, R.M., and Lippert, R.A.: 'Notes on regularized least squares', 2007

[17] Bottou, L.: 'Stochastic gradient descent tricks': 'Neural networks: Tricks of the trade' (Springer, 2012), pp. 421-436