SoftwareX 22 (2023) 101340

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

HSIToolbox: A web-based application for the classification of hyperspectral images

Zeno Dhaene [a], Nina Žižakić [a], Shaoguang Huang [b,*], Xian Li [c], Aleksandra Pižurica [a]

[a] Department of Telecommunications and Information Processing, TELIN-GAIM, Ghent University, Ghent, Belgium
[b] School of Computer Science, China University of Geosciences, Wuhan, China
[c] School of Electronics and Information Engineering, Harbin Institute of Technology, Harbin, China

ARTICLE INFO

Article history:
Received 9 November 2022
Received in revised form 22 January 2023
Accepted 10 February 2023

Keywords:
Hyperspectral images
Hyperspectral image classification
Classification tools
Web applications
Remote sensing

ABSTRACT

Recent deep-learning-based classification models for hyperspectral images (HSIs) yield near-perfect classification accuracy on benchmark data sets. However, applying them in real scenarios often requires programming skills and machine learning expertise, which makes the usage of these algorithms unfriendly for domain experts. In this paper, we provide a web-based application, HSIToolbox, for the classification of HSI with a user-friendly graphical interface, which allows a domain expert to view, label and manage HSIs, and to train out-of-the-box deep learning models on the server. HSIToolbox supports different operating systems and different HSI data formats. With a developed queuing system and web interface, HSIToolbox can be accessed remotely by multiple users at the same time.

Code metadata

Current code version	v1.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-22-00367
Code Ocean compute capsule	–
Legal Code License	Apache License 2.0
Code versioning system used	git
Software code languages, tools, and services used	Docker, Tensorflow, Python, .NET Core, C#, React, MongoDB
Compilation requirements, operating environments & dependencies	.NET Core 3.1, Python 3.7, etc.
If available Link to developer documentation/manual	https://github.com/zenodhaene/HyperspectralClassification/blob/master/docs/README.md
Support email for questions	zeno.dhaene@gmail.com

1. Motivation and significance

Climate change and issues that it entails (natural disasters, rising sea levels, Arctic sea ice decline and food security — to name a few) are among the greatest challenges of the 21st century. The need to observe how our world is changing has become of paramount importance. Remote sensing using hyperspectral imaging provides a way of doing this, by capturing the Earth's surface and classifying its areas depending on the type of surface.

Hyperspectral image (HSI) classification is an active field of research in remote sensing. Over the past decade, a number of classification methods have been proposed [1–3], which are all evaluated and compared with each other on standard benchmark academic datasets such as Indian Pines (imaged in 1995 [4], Pavia University (2001) and Salinas (imaged in 1998). Current state-of-the-art deep learning methods [5–11] have matured to the point of being able to achieve almost perfect accuracy scores when classifying these 20-year-old academic datasets. However, the impact of such methods seems to remain limited to the academic circles. It is unclear how these models would perform on real-world datasets such as those captured by EO1 satellites of NASA [12] and the Hyperion-2 sensors of European Space Agency (ESA) [13]. These datasets do not include labels and it is therefore not possible to evaluate nor to train academical models on them.

Influential deep learning researchers such as Andrew Ng argue that the deep learning community needs to shift from such a *model-centric approach* (where for a fixed dataset we try to find

* Corresponding author.
E-mail address: huangshaoguang@cug.edu.cn (Shaoguang Huang).

https://doi.org/10.1016/j.softx.2023.101340

the best possible model) to a *data-centric approach* (where the focus is also on improving the data) [14]. The argument follows from the fact that in practical applications, starting from a baseline state-of-the-art model applied to a specific application, model-centric approach often gives a couple of per-mille points of improvement, while data-centric approach can improve the performance for even more than 10% for the same application [14]. The field of hyperspectral image classification, given the fact that nearly perfect scores are achieved on academical benchmarks, seems mature for such a shift where we instead try to build solutions that consist of both the deep learning models and practical datasets that will perform well in the real-world problems.

The software presented in this paper aims to bridge the shift to the data-centric approach by allowing both deep learning experts and domain experts to create impact in the real world. It provides a web-based user-friendly graphical interface that allows a domain expert to add, label and manage HSI datasets, and train different out-of-the-box deep learning models on them. Our software can read several dataset formats, so that hyperspectral images from different datasets can be uploaded in a uniform fashion for further labelling and classification. In addition, our software can run on a server with dedicated hardware for deep learning (i.e. GPUs), and be accessed remotely via the web interface allowing users to use the system simultaneously. For the training of the deep learning models, we have implemented a queuing system. Furthermore, deep learning experts can easily plug new neural network architectures and update the existing ones.

Working with HSI datasets calls for special properties that differ from when working with regular image datasets. Firstly, HSI datasets have different formats and the tool that processes them needs to be able to read these different formats. Secondly, when a domain expert is labelling the images, it is often useful for them to be able to see different bands (channels) of an image in order to assign the correct label to an area of an image (for example, looking at the Xnm frequency band, corn fields look the same as wheat fields, but in Ynm frequency band, they look very different).

For these reasons, it is important that the labelling tool is customized for HSI data. There are many tools for data labelling, both commercial ones such as SuperAnnotate [15], Labelbox [16], Supervise.ly [17] and free open-source ones such as LabelImg [18], CVAT [19], labelme [20] and VoTT [21], but none seem to have these special functionalities for hyperspectral images. Concerning all-in-one tools for deep learning, there are commercial ones such as Amazon SageMaker [22] and Google Cloud Platform [23], which offer data labelling and neural network training and inference. However, they are not open-source and not free to use for everyone. Furthermore, they are general deep learning tools, also not offering special functionalities for HSI and thus not supporting special formats of HSI datasets nor allowing visualizations of different channels of a hyperspectral image during the labelling process. Matlab allows reading HSI datasets, but it is a commercial, non-open-source tool and does not allow data labelling [24]. The only other tool that is specialized for working with hyperspectral data (to the best of our knowledge), Hypernet [25] (developed by ESA), focuses only on neural network training and inference and does not support data labelling.

Priego and Duroy, who also identified the lack of publicly accessible labelled HSI datasets, have proposed a tool for generation of synthetic hyperspectral images [26]. While the synthetic datasets can be used as a tool for approximate evaluation of classification algorithms, they cannot fully replace the real labelled datasets.

2. Software description

Classifying hyperspectral images is – as is the case with many deep learning processes – a very compute-intensive process that benefits from dedicated hardware like GPUs. In an age where working remotely is becoming more prevalent, our application is made to be run on server hardware with dedicated hardware, with a web interface and a queuing system that processes model training requests one at a time on a first come, first-served basis, allowing users to use the system simultaneously.

2.1. Software architecture

The entire solution is deployed as a collection of microservices, where all microservices are programmed to be OS-independent. A complete architecture diagram is visible in Fig. 1. The user enters the web application through the frontend, which can be reached through a dedicated gateway service. This service also exposes the endpoint of the .NET core backend, which the user will use to interact with the database.

As database technology, we use MongoDB because of the GridFS extension, which can be used to store our datasets. Because of the unstructured documents that MongoDB uses to store data, architecture developers can choose which parameters they implement, and which performance metrics they report.

The Python backend service is not directly reachable from the outside. Instead, the user interacts with the .NET core backend as an intermediary to get Python-only functionality, such as visualizing datasets. The Python backend consists of an API, a master and a slave. The API responds to API requests from the .NET core backend, while the master continually checks the database to see whether there are new models that can be enqueued. The models are dequeued in a first come, first served manner, and are trained by a separate slave thread that is started by the master. This way, implementation errors do not influence the proper functioning of the API, and the master can retry or skip failed training attempts.

2.2. Software functionalities

HSIToolbox provides a web-based graphical interface that allows the user to go through the entire process of classifying hyperspectral images, from uploading the dataset to inspecting the result. In this section, we will go through a typical workflow that an expert might encounter in the field, and describe the software features along the way. The workflow is shown as a diagram in Fig. 2.

2.2.1. Uploading the dataset

The user starts by choosing and uploading a dataset. It is possible to upload a ground truth datafile separately, making it possible to upload both unlabelled datasets as well as datasets with limited ground truth samples available. The dataset is read by the Python backend to see whether it contains valid data.

Currently, there are two implemented data formats: the Matlab data model for all widely used academic hyperspectral image datasets, and the data model used by the Earth Observation 1 - Hyperion satellite mission.

2.2.2. Manually indicating ground truth samples

After the dataset has been uploaded and validated, the user can inspect it and give it a name. For the next step, we distinguish the cases where we have ground truth data available and want to use it from the cases where there is no labelled ground truth data available. In the latter case, the user needs to create geographical classes themselves, and indicate some pixels that belong to each class. This will serve as a starting point for the supervised deep

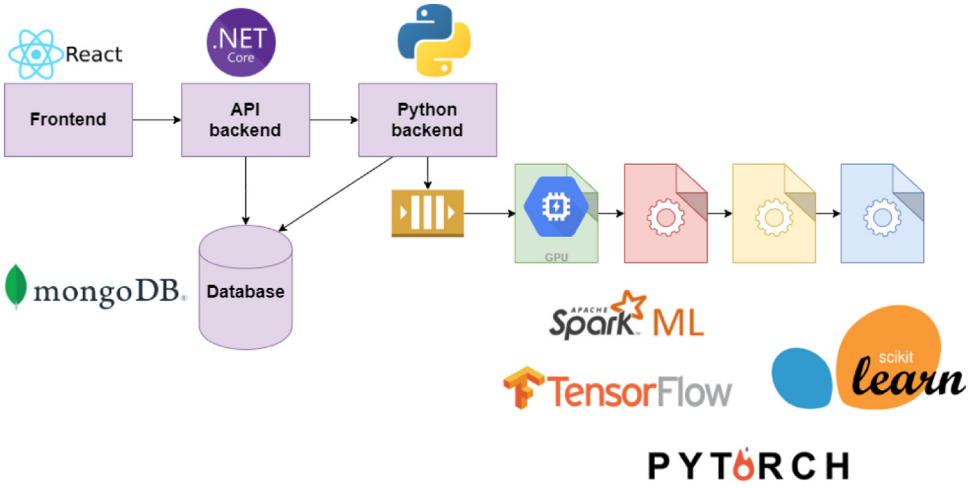


Fig. 1. Software architecture.

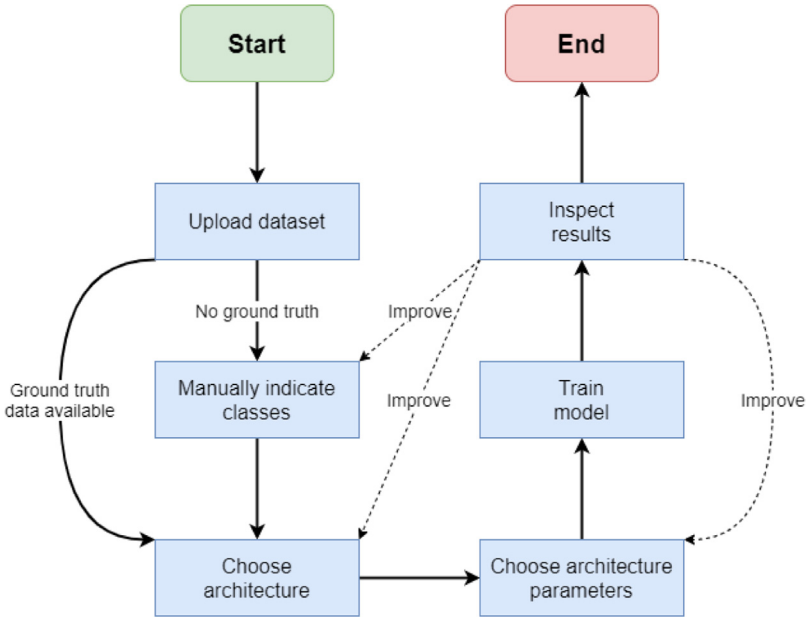


Fig. 2. Workflow diagram.

learning architecture that will be chosen later. This step can be skipped if ground truth data is available. An example of manual labels created in the application can be seen in Fig. 3.

It should be noted that the data labelling to create a ground truth completely depends on the users. Our application supports both sparse labelling and dense labelling. There is no recommendation on the number of pixels to be labelled. If users label only a few pixels, the unlabelled pixels will be marked with 0. In the experiments, these unlabelled pixels are not used for validation.

2.2.3. Choosing an architecture

One of the most important and impactful features of the HSIToolbox software is the ability to create and test different architectures. This way, the user can compare different techniques for their specific dataset, and see which technique best solves the problem at hand. It is also a benefit to machine learning researchers, as they can directly see how different architectures behave on the same dataset with identical parameters. The user is presented with a selection of architectures that have been created and added to the application, allowing expert users and researchers to work in parallel.

2.2.4. Choosing architecture parameters

At least as important as choosing a suitable architecture is defining the parameters. After the user has chosen an architecture, they are presented with a range of options. These architecture parameters are defined by the designer of the architecture, and are visualized automatically in the application as sliders and input fields. An example of configured model parameters is shown in Fig. 4

To create a new architecture, the model architect must define the two mandatory inputs: the model parameters, and the python implementation of the architecture. These must be implemented in the source code itself. Implementation details are present in the technical documentation, available in the GitHub repository. By correctly extending the necessary python classes in the source code, the model selection and custom parameters will automatically be visualized by the front-end, making the architecture

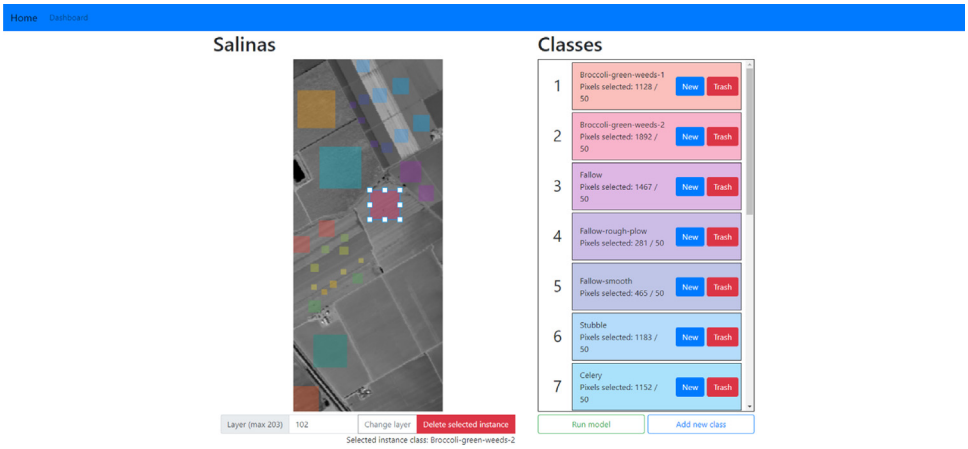


Fig. 3. Manual labelling of geographical classes in the web application.

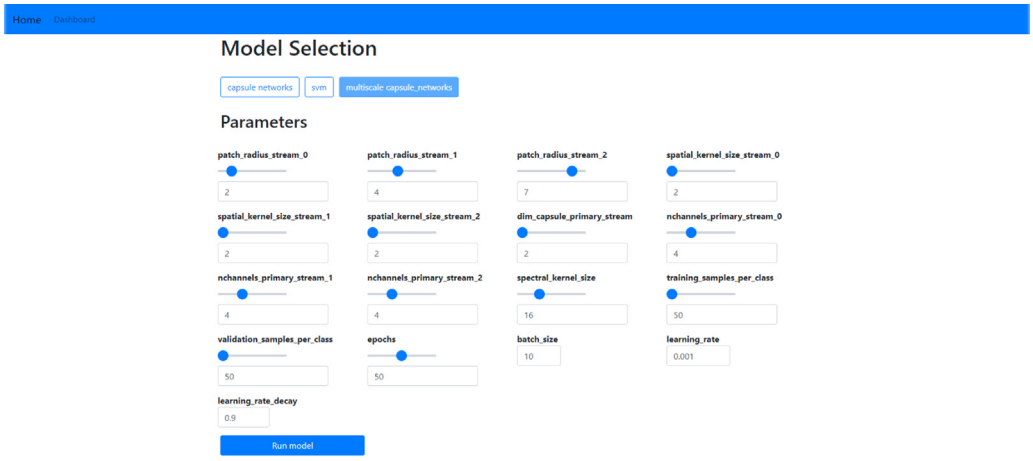


Fig. 4. Model parameter selection page.

ready for use in the application. Additionally, several examples are available in the source code for reference.

GPU/CPU hardware utilization is the responsibility of the model architect, and is therefore implemented in the Python implementation of that specific architecture. If desired, the developer can include the hardware choice as a parameter of the model.

2.2.5. Training the model

With the dataset, the architecture and its parameters, and the initial starting point for the supervised deep learning process defined, the system has all the information it needs to train a predictive model. Once the user starts the training process, it is enqueued. The user can see the status as the queue, as well as the progress of the currently running model, in the dashboard overview. As the model is trained on dedicated hardware on a separate machine, multiple users can work in parallel and can continue working while their models are running. An example of the dashboard is visualized in Fig. 5.

2.2.6. Inspecting the results

When the system has finished training the model, it is initially shown in the recent model results section of the dashboard. Additionally, the user can inspect all trained models and filter them by dataset, version, or by architecture. Clicking on a previously trained model brings up the model result view, an example of which is shown in Fig. 6.

Here, the user can inspect the model performance on the initially defined ground truth samples (that were either defined manually or taken from the ground truth data). A predictive map of the entire hyperspectral image is shown, which allows the user to visually inspect the quality of the predictive capability of the model. Along with this classification map, the number of pixels – or if the resolution of the dataset was specified, the actual area – that each class covers is shown (Fig. 7). Several metrics, like the overall and average accuracy, are calculated as well and other metrics can be added by the designer of the architecture (Fig. 8). Finally, a confusion matrix is created that visualizes how any two classes are confused with each other (Fig. 8).

2.2.7. Improving the model

Using the visualization of the predicted classification map and the metrics that are calculated over its final results, the user can further refine the model. They can choose different architecture parameters, such as increasing the number of epochs that the model is being run for, or they can choose a different architecture altogether. This way, the user can choose whether they want to maximize the predictive capability of a particular dataset, or compare the predictive capabilities of two architectures on the same dataset.

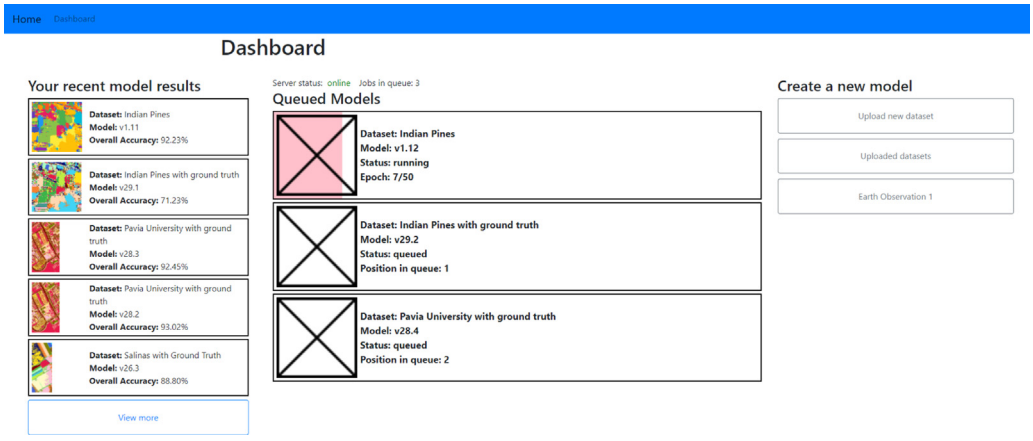


Fig. 5. Dashboard of the application.

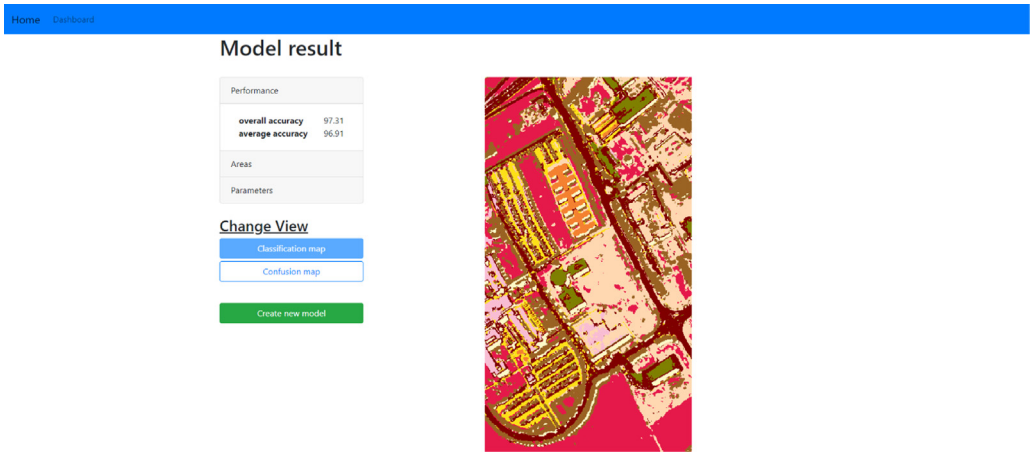


Fig. 6. A detailed view of a model's performance.

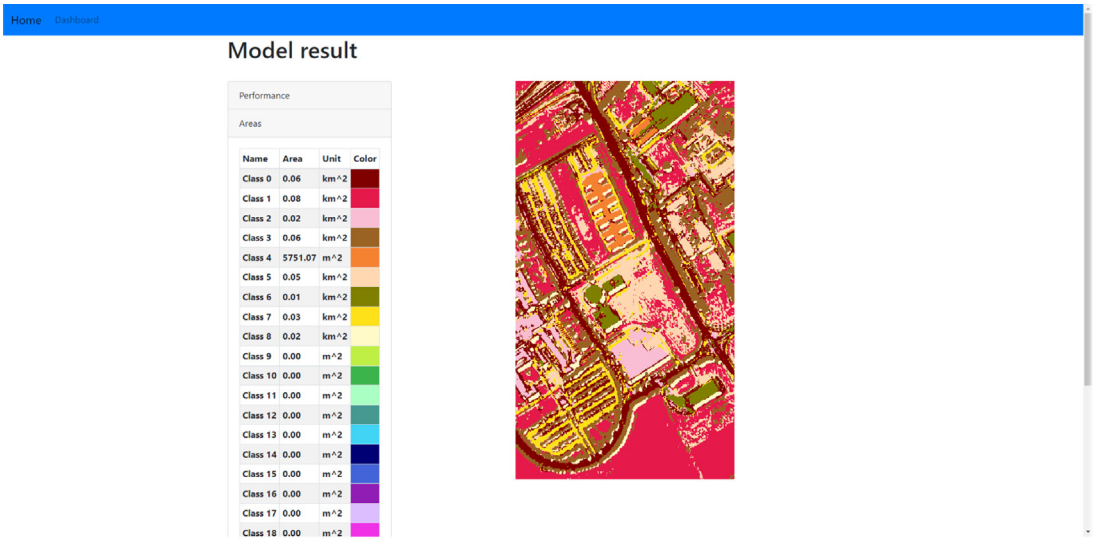


Fig. 7. The calculated areas for each classification class.

Fig. 8. The model parameters and classification confusion map.

Using the confusion map, it is sometimes apparent that two classes are easily confused with each other. This can either be because of an error in the labels of the ground truth data (the starting point provided by the user for the supervised deep learning architecture), or because of a lack of training data. Both issues can be addressed by modifying the manually defined geographical classes.

Doing any of these modifications has been made very easy, by allowing the user to create a new model from an already trained model. The class indications and architecture parameters are automatically carried over in a new model, which is defined as a new subversion of the model. By filtering the finished models for a specific version, the user can see the progress of the predictive capabilities of the model over different subversions.

3. Illustrative examples

Using the web application, the user can create a model that will autonomously train and classify a hyperspectral image. Below, we explore two datasets: the Pavia University dataset, which is often used in academic research to evaluate the performance of hyperspectral classification architectures, and the Earth Observation 1 [12] dataset, a collection of hyperspectral images gathered by NASA using satellites with hyperspectral sensors. The architecture that both datasets are being trained with is included in the software.

Both datasets are trained using a multi-stream capsule network, which uses capsules to learn hierarchical features in the spectral bands of the two hyperspectral images. For the academical dataset, a ground truth set is available which will be used to train the model: it consists of 9 geographical classes with 150 randomly chosen pixels per class for the training process, and another 150 pixels per class for validation, this is about 3.2% of the available ground truth data, showing that this particular architecture performs well even with very little training data. The classification result is shown in Fig. 9.

For the practical dataset, we manually define two classes: land and water. The image is captured over the island of Moroni, near the coast of Mozambique. Due to the poor resolution of the dataset, there is not enough data to detect more detailed classes like cities or agricultural land. An aerial view of the hyperspectral image is shown in Fig. 10(a), with the predicted classification map shown in Fig. 10(c). The manually indicated training samples are shown in Fig. 10(b), of which 50 random samples per class were picked for the classification process, and another 50 per class for validation. To inspect the accuracy of the architecture, an overlay of the satellite view and the predicted classification map is shown in Fig. 10(d).

4. Impact

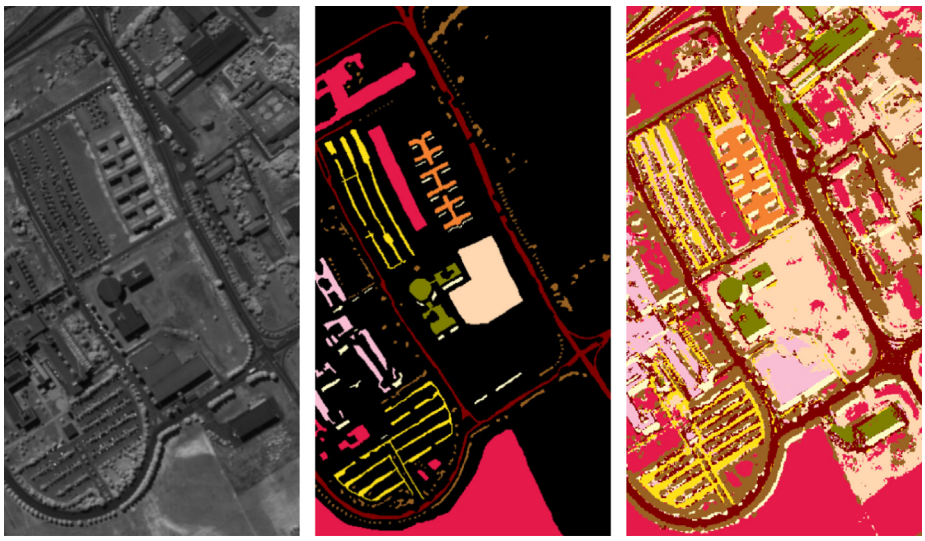
With this web application, hyperspectral classification research can finally be applied in practice on a large scale. Due to the user-friendly interface, programming and machine learning expertise is no longer needed for classifying hyperspectral images, which removes the high barrier of entry.

This software can be used to further advance the field of hyperspectral image classification, as new problems with practical, unlabelled datasets will undoubtedly come to light. This will require close collaboration between experts in the field and academical research, which this software facilitates and encourages. The web application can also be a valuable tool for researchers, as models are trained individually. This enables the researcher to perform parameter searches as well as review the impact of how training samples are chosen. The researchers can also easily add and compare different architectures by creating new python files in the backend. As this architecture implements a standardized interface, this is a plug-and-play system, allowing for accelerated development.

Additionally, this application may be used in many different research fields such as climate research and agricultural research, two fields which could certainly benefit from accurately classifying every square metre in a large area. The software is designed to be as generic as possible, to increase its applicability in many fields of study.

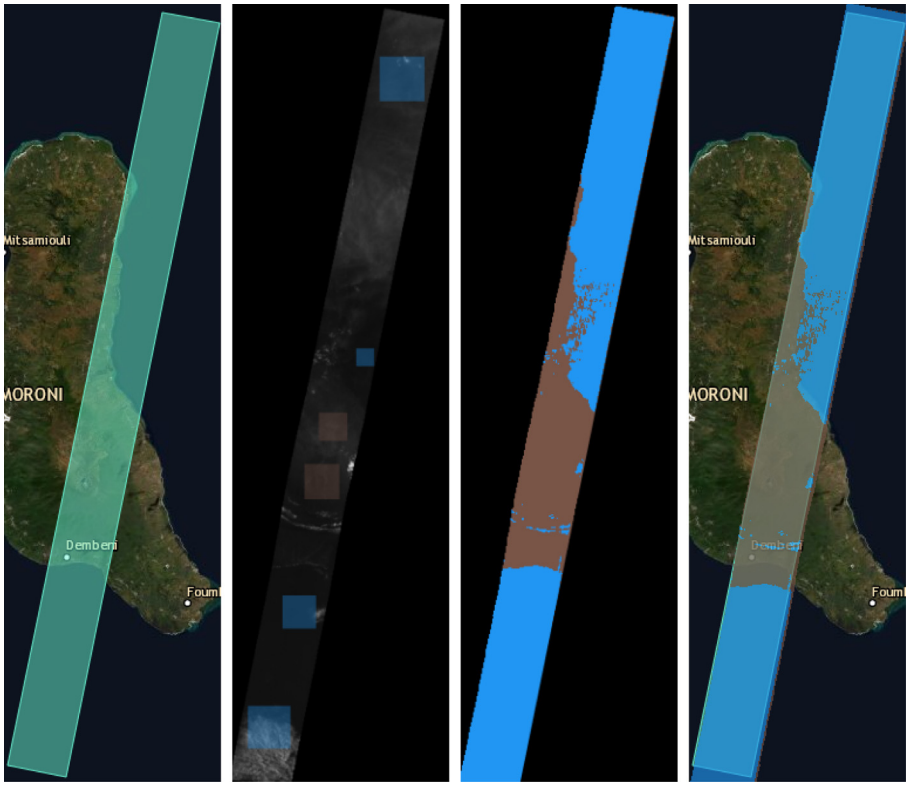
5. Conclusions

We presented HSIToolbox, a user-friendly software tool that allows domain experts without programming skills and machine learning expertise in remote sensing to classify HSI using deep learning algorithms. HSIToolbox is a web-based application with a graphical user interface, which means that it is OS-independent and can be accessed by users remotely. Dedicated computing hardware for deep learning like GPUs can be deployed on the server. Due to the developed queuing system, multiple users can access the web application at the same time and make plans for the training of different deep learning models. HSIToolbox also supports different types of data formats of HSIs and allows experts to label data for training deep learning models. This tool facilitates accurate interpretation of large-scale HSIs by producing precise classification results with deep learning classifiers. The HSIToolbox is available for download via GitHub https://github.com/zenodhaene/HyperspectralClassification.



(a) Grayscale visualization (b) Ground truth (c) Classification

Fig. 9. Classification result on satellite dataset Pavia University with manually labelled data.



(a) Map view (b) Labels (c) Classification (d) Overlay

Fig. 10. Classification result on data set EO1 with manually labelled data.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have shared the link to my code at the Attach File step.

Acknowledgements

This work was supported in part by the Flanders AI Research Programme, Belgium grant no. 174B09119, in part by the Bijzonder Onderzoeksfonds (BOF), Belgium under Grant BOF.24Y.2021. 0049.01 and in part by the "CUG Scholar" Scientific Research Funds at China University of Geosciences (Wuhan), China (Project No. 2022164).

References

[1] Ghamisi Pedram, Yokoya Naoto, Li Jun, Liao Wenzhi, Liu Sicong, Plaza Javier, et al. Advances in hyperspectral image and signal processing: A comprehensive overview of the state of the art. IEEE Geosci Remote Sens Mag 2017;5(4):37–78.

[2] Signoroni Alberto, Savardi Mattia, Baronio Annalisa, Benini Sergio. Deep learning meets hyperspectral image analysis: A multidisciplinary review. J Imag 2019;5(5):52.

[3] Huang Shaoguang, Zhang Hongyan, Pižurica Aleksandra. Subspace clustering for hyperspectral images via dictionary learning with adaptive regularization. IEEE Trans Geosci Remote Sens 2022;60:1–17.

[4] Hsieh Pi-Fuei, Landgrebe David A. Statistics enhancement in hyperspectral data analysis using spectral-spatial labeling, the EM algorithm, and the leave-one-out covariance estimator. In: Imaging spectrometry IV. Vol. 3438. International Society for Optics and Photonics; 1998, p. 183–90.

[5] Zhou Peicheng, Han Junwei, Cheng Gong, Zhang Baochang. Learning compact and discriminative stacked autoencoder for hyperspectral image classification. IEEE Trans Geosci Remote Sens 2019;57(7):4823–33.

[6] Zhong Ping, Gong Zhiqiang, Li Shutao, Schönlieb Carola-Bibiane. Learning to diversify deep belief networks for hyperspectral image classification. IEEE Trans Geosci Remote Sens 2017;55(6):3516–30.

[7] Hu Wei, Huang Yangyu, Wei Li, Zhang Fan, Li Hengchao. Deep convolutional neural networks for hyperspectral image classification. J Sens 2015;2015.

[8] Chen Yushi, Jiang Hanlu, Li Chunyang, Jia Xiuping, Ghamisi Pedram. Deep feature extraction and classification of hyperspectral images based on convolutional neural networks. IEEE Trans Geosci Remote Sens 2016;54(10):6232–51.

[9] Li Wei, Wu Guodong, Zhang Fan, Du Qian. Hyperspectral image classification using deep pixel-pair features. IEEE Trans Geosci Remote Sens 2016;55(2):844–53.

[10] Xu Yonghao, Zhang Liangpei, Du Bo, Zhang Fan. Spectral–spatial unified networks for hyperspectral image classification. IEEE Trans Geosci Remote Sens 2018;56(10):5893–909.

[11] Hang Renlong, Liu Qingshan, Hong Danfeng, Ghamisi Pedram. Cascaded recurrent neural networks for hyperspectral image classification. IEEE Trans Geosci Remote Sens 2019;57(8):5384–94.

[12] National aeronautics and space administration (NASA). 2021, [Online] https://eospso.nasa.gov/missions/earth-observing-1. [Accessed 23 December 2021].

[13] European space agency (ESA). 2021, [Online] https://sentinel.esa.int/web/sentinel/missions/sentinel-2. [Accessed 23 December 2021].

[14] Ng Andrew. MLOps: From model-centric to data-centric AI. 2021.

[15] SuperAnnotate. 2021, [Online] https://www.superannotate.com/. (Accessed 23 December 2021).

[16] Labelbox. 2021, [Online] https://labelbox.com/. (Accessed 23 December 2021).

[17] Supervise.ly. 2021, [Online] https://supervise.ly/. [Accessed 23 December 2021].

[18] LabelImg. 2021, [Online] https://github.com/tzutalin/labelImg. [Accessed 23 December 2021].

[19] CVAT. 2021, [Online] https://github.com/openvinotoolkit/cvat. [Accessed 23 December 2021].

[20] Labelme. 2021, [Online] https://github.com/wkentaro/labelme. [Accessed 23 December 2021].

[21] Vott. 2021, [Online] https://github.com/microsoft/VoTT. (Accessed 23 December 2021).

[22] Amazon SageMaker. 2021, [Online] https://aws.amazon.com/sagemaker/. [Accessed 23 December 2021].

[23] Google cloud platform. 2021, [Online] https://cloud.google.com/. [Accessed 23 December 2021].

[24] MATLAB. 2021, [Online] https://mathworks.com/products/matlab.html. [Accessed 23 December 2021].

[25] Hypernet. 2021, [Online] https://github.com/ESA-PhiLab/hypernet. [Accessed 23 December 2021].

[26] Priego Blanca, Duroy Richard. Amigo: A tool for the generation of synthetic hyperspectral images. In: 9th Workshop on hyperspectral image and signal processing: evolution in remote sensing. IEEE; 2018, p. 1–5.