Deep Learning for Hyperspectral Image Classification on Embedded Platforms

Siddharth Balakrishnan*, David Langerman†, Evan Gretok‡ and Dr. Alan D. George§
Department of Electrical and Computer Engineering, University of Pittsburgh
Room 1238D, Benedum Hall
NSF Center for Space, High-performance, and Resilient Computing (SHREC)
Pittsburgh, PA 15261
{*sid.bala, †dal181, ‡ewg13, Alan.George}@pitt.edu

***Abstract*— Hyperspectral image (HSI) analysis refers to the processes used to identify and classify objects photographed using equipment that can image photons from a broad range of the electromagnetic spectrum. Downlinking such large images from space on radiation-resistant platforms with limited on-board computing power takes a large amount of time, memory, and other mission-critical resources. Performing such analysis in space before downlinking all images will save these resources by enabling a subset of images of interest to be downloaded rather than the entire set. The goal of this study is to benchmark and evaluate HSI-classification methods which incorporate deep learning on embedded platforms with limited computing resources. Support Vector Machine (SVM), Multi-Layer Perceptron (MLP), and Convolutional Neural Network (CNN) are the classification methods used in this study. These algorithms were executed on a desktop PC and two embedded platforms: the ODROID-C2 and the Raspberry Pi 3B. Accuracy, run-time, and memory benchmarks determined the optimal model for each platform. Based on results gathered in this research, CNN classification is recommended for the desktop PC due to its high accuracy of 97%. MLP classification is recommended for the embedded platforms under study, as it showcased the shortest run-time and second-highest accuracy.**

Keywords—hyperspectral image analysis, deep learning, embedded platforms, performance benchmarking

I. Introduction

Numerous images of Earth are taken from satellites and other spacecraft. These images are usually very high in resolution and bit-depth. As a result, downloading every single image for analysis on earth is inefficient in terms of communication time and processing power in an already computation-constrained environment [1]. The primary goal of this research is to benchmark and evaluate deep-learning apps for hyperspectral image (HSI) classification on embedded platforms. From this study, the optimal HSI-classification method for platforms with limited computing capabilities can be determined. This study also serves as a framework for conducting such analysis on-board a spacecraft, which could allow a subset of images of interest to be downloaded – saving time, memory, and other mission-critical resources.

Hyperspectral images are taken with a hyperspectral camera that collects amplitude readings from a subset of spectral bands (various wavelengths in the electromagnetic spectrum) for each pixel in the image [2]. Patterns can be extracted from these amplitudes in order to classify each pixel in the image. By doing so, the HSI data of interest can be recognized for further investigation, depending upon the context in which HSI is being used. HSI imagery is used in a wide variety of applications such as astronomy and space surveillance. For example, [3] uses an adaptive optics-compensated telescope to acquire HSI.

Three classification methods were used for HSI analysis in this study: Support Vector Machine (SVM); Multi-Layer Perceptron (MLP); and Convolutional Neural Network (CNN). SVM is a machine-learning model that creates a margin in a transformed input space, splitting the input data into two classes using hyperplane in multidimensional space [4]. SVMs are inherently a two-class classifier. As a result, in order to conduct predictions for a multi-class dataset, a one-versus-all method is used for each class. A MLP is a deep-learning model that is capable of modeling nonlinear functions. MLPs consist of "fully connected layers" in which every node is connected with respective weights determined when a model is trained [5]. Similar to MLPs, CNNs are a deep-learning model, but they contain convolutional layers, where each layer transforms one set of feature maps. The last few layers of a typical CNN are normally "fully connected layers" that mirror an MLP in functionality. However, the convolutional layers in CNNs generally increase their accuracy over MLPs, because these convolutional layers tend to extract relevant features from the image and discard noise and extraneous information [5] [6].

The classification models were all trained on the desktop PC and the final prediction was executed on the three platforms listed below in Table I.

II. Related Work

In [7], a review of different hyperspectral image analysis techniques using deep learning concluded that there is minimal evidence suggesting that deep-learning models outperform reference methods, but they are quite competitive. The literature reviewed in this paper noted that a widely used reference model

TABLE I. Platform Comparison

	ODROID-C2	**Raspberry Pi 3B**	**Desktop PC**
Architecture	ARM Cortex -A53 Quad Core	ARM Cortex -A53 Quad Core	x86 Quad Core
Processor	Amlogic S905	Broadcom BCM2837	Intel i7-6700 vPro
Clock	1.5 GHz	1.2 GHz	3.4 GHz
RAM	2 GB DDR3	1 GB DDR2	16 GB DDR3
OS	Ubuntu 16.04 (Mate)	Raspbian 8	Windows 10 Pro

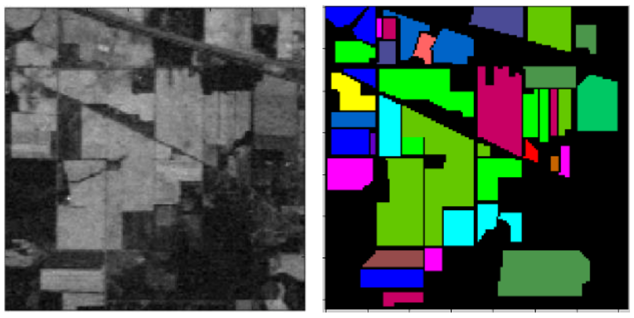


Fig. 1. Ground Truth Matrix Visualization from Original Image

is the SVM model. It was found that proposed and reference models in this work obtained an overall training accuracy of over 95%.

In [8], supervised HSI classification algorithms along with Markov Random Fields post-processing is investigated to assess accuracy of various models before and after post-processing. The HSI classification algorithms include SVM and CNN models. The CNN model was based on a model developed in [5]. Moreover, these models were run on a server with Nvidia GeForce GTX 1080 and Tesla K40c GPUs [5].

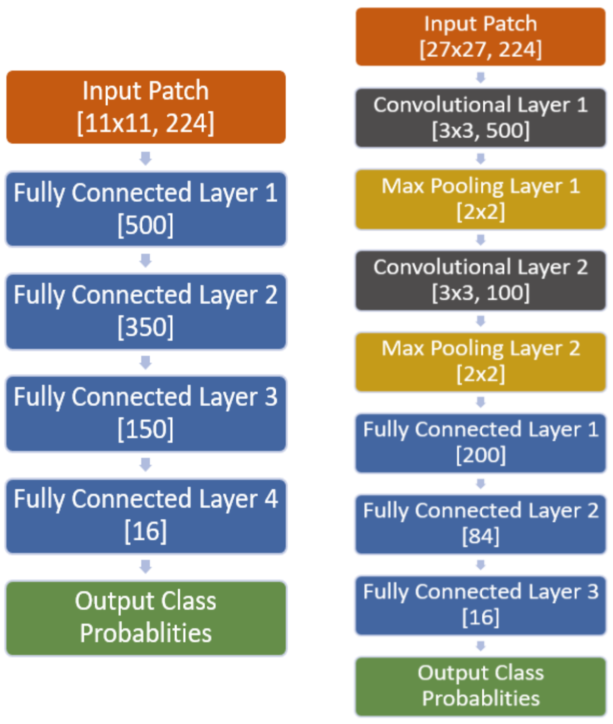
The models developed in [5] are part of a study in which deep-learning networks are developed for land-cover classification with HSI. These models deal with the challenges of high-dimensionality HSI datasets effectively by extracting band-specific spectral and spatial features. These extracted features are then used to perform pixel-wise landcover classification. The models are claimed to outperform the highest reported accuracies on popular HSI data sets such as the Indian Pines data set. As a result, the MLP and CNN models used in this study were also based on the same models that were developed by [5].

III. Methods

This section discusses the HSI dataset, models, and methods used to train, test, and make final predictions. Furthermore, the parameters and architectures for each model are detailed.

A. *Dataset*

The Indian Pines HSI dataset was leveraged to train and test the classification algorithms under study. This dataset is widely used by HSI researchers in testing proposed classification algorithms. The dataset represents an image of farmland in northwestern Indiana and consists of 224 spectral bands with 16 different agricultural classes [9]. The ground truth of the dataset identifies different types of land cover, ranging from buildings to a variety of vegetation, in the farmland. The ground-truth matrix constructed from the dataset was converted into an image to visualize the different types of landcover, shown in Fig. 1.


Input Patch [11x11, 224]
Fully Connected Layer 1 [500]
Fully Connected Layer 2 [350]
Fully Connected Layer 3 [150]
Fully Connected Layer 4 [16]
Output Class Probablities
Input Patch [27x27, 224]
Convolutional Layer 1 [3x3, 500]
Max Pooling Layer 1 [2x2]
Convolutional Layer 2 [3x3, 100]
Max Pooling Layer 2 [2x2]
Fully Connected Layer 1 [200]
Fully Connected Layer 2 [84]
Fully Connected Layer 3 [16]
Output Class Probablities

(a) MLP Architecture [5] (b) CNN Architecture [5]

Fig. 2. Model Architectures Used in this Study

B. *Models*

The Scikit-learn Python library was leveraged to develop the SVM [10]. The parameter values for the SVM model are summarized in Table II(a). A radial basis function (which uses the squared Euclidean distance between feature vectors) was used as kernel to develop the SVM. The value of C in Table II(a) indicates the relative weight coefficient. It should be noted that the tolerance value may differ on other platforms. The values for each parameter were chosen using a five-fold, cross-validation technique, a resampling procedure in which the shuffled dataset is split into five groups and each group is used as the test set while the model is trained on the rest of the dataset. For each instance that the model is trained, parameters are tuned over the specified range of values, and the model with the optimal parameters is selected to maximize accuracy.

The TensorFlow framework was used to construct and train the MLP and CNN deep-learning models in this study. These

TABLE II. Parameters Used for SVM, MLP and CNN

(a) SVM	*(b) MLP*	*(c) CNN*
Gamma: 2^{-8}	Patch Size: 1	Patch Size: 27
C: 2^{7}	Batch Size: 200	Batch Size: 200
Tolerance: 1e-14	Learning Rate: 0.01	Learning rate: 0.01

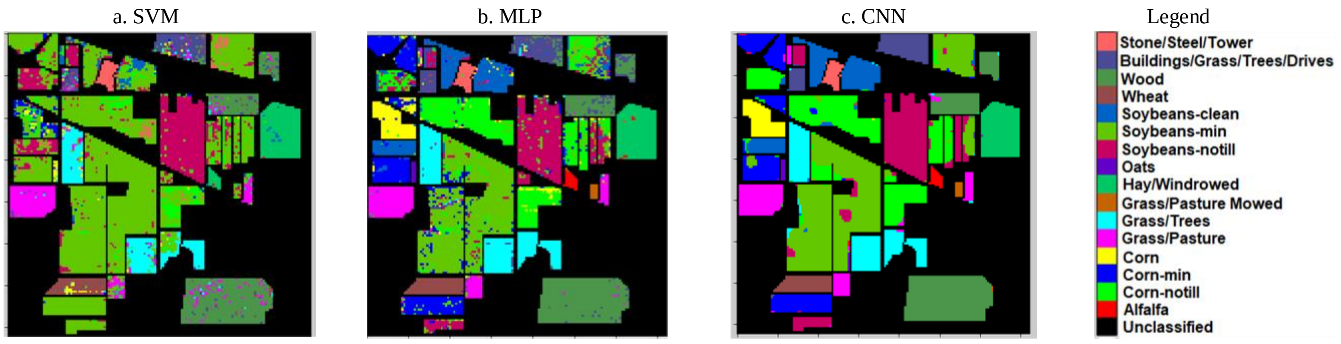


Fig. 3. Image Outputs

models were based on land-cover classification models developed by the Satellite Application Center from the Indian Space Research Organization [5] [11]. The MLP model can be described by a weighted, directed acyclic graph. The output of each nodal layer is a function of the sum of inputs modified by a nonlinear transfer function such as a sigmoid, which was the activation function used in [9]. The architecture of the MLP model was set so that each patch of the image could be used as input to the model with the architecture shown in Fig. 2a. The parameters used to construct the model are summarized in Table II(b). The model was trained for 50,000 epochs.

Similarly, each patch of the image was used as an input to the CNN with the architecture shown in Fig. 2b. Typical CNNs contain alternating layers of convolutional filters and max-pooling layers. The final layers of most CNNs are fully connected layers, used for classification [12].

C. *Training, Testing, and Prediction*

The raw dataset was pre-processed to include a border to avoid loss of data at the output. The pixels in the image were divided into 80% training and 20% testing sets. These sets were used to develop and validate the SVM, MLP, and CNN models on the desktop PC before porting the trained models to the ODROID-C2 and Raspberry Pi 3B platforms.

For the final prediction, each pixel from the data was fed into the trained models and the predicted outputs were used to reconstruct an image as shown in Fig. 3. The labels predicted for each pixel were compared with the pre-defined labels in the ground-truth image to determine the accuracy of the prediction. During prediction, run-time and memory benchmarks were calculated. These predictions and calculations were executed on the desktop PC, ODROID-C2, and Raspberry Pi 3B platforms.

After the models were trained and tested, the entire data set was used to conduct the final predictions. The accuracies of the models were noted, and the run-time and memory benchmarks were averaged over ten trials.

IV. Experimental Results

Accuracy, run-time, and memory benchmarks on the desktop PC, ODROID-C2, and Raspberry Pi 3B are detailed in this section. The overall accuracy of each model tested in this study is compared to the accuracy of other models from the literature. Inter-class accuracies for each of the models are also reported.

TABLE III. Accuracy of Models

	Ours	Others
SVM	62%	68% [8]
MLP	85%	82% [5]
CNN	97%	96% [5]

TABLE IV. Inter-Class Accuracies for Each Model

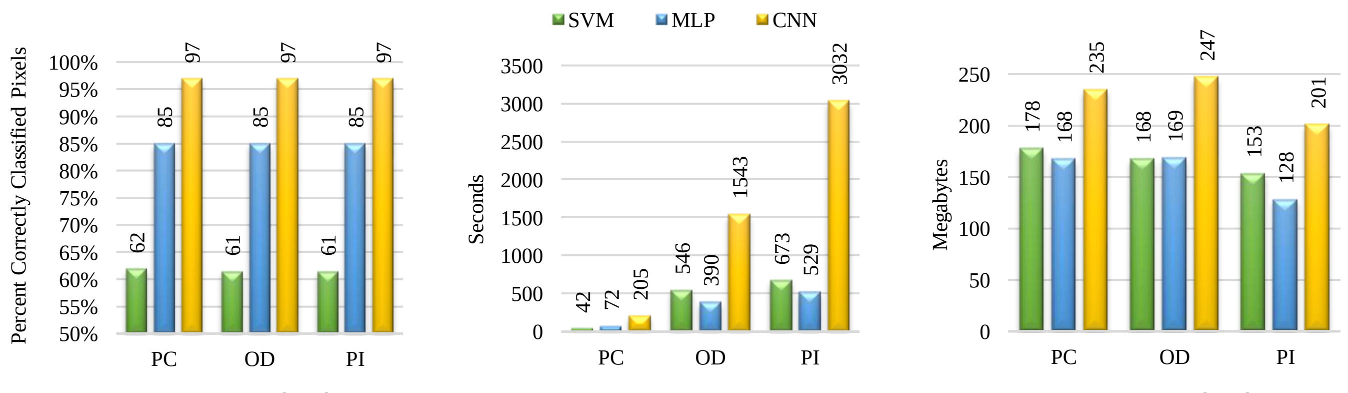
Classes		SVM	MLP	CNN
1.	Stone/Steel/Tower	2.2%	95.7%	100%
2.	Build/Grass/Tree/Drives	15.1%	76.3%	81.2%
3.	Wood	2.7%	82.3%	63.2%
4.	Wheat	8.9%	92.8%	76.7%
5.	Soybean-Clean	75.2%	95.0%	78.5%
6.	Soybean-Min	79.6%	97.3%	99.6%
7.	Soybean-Notill	7.1%	96.4%	100%
8.	Oats	93.5%	98.1%	71.2%
9.	Hay/Windrowed	5.0%	100%	100%
10.	Grass/Pasture Mowed	79.8%	90.1%	93.7%
11.	Grass/Trees	89.0%	75.4%	90.1%
12.	Grass/Pasture	15.2%	89.0%	70.5%
13.	Corn	81.5%	99.5%	73.1%
14.	Corn-Min	89.4%	91.0%	66.1%
15.	Corn-Notill	61.7%	83.2%	90.6%
16.	Alfalfa	100%	100%	60.8%

A. *Accuracy Benchmarks*

The accuracy results shown in Fig. 4a were determined through comparison of the model's pixel-wise prediction with the preset pixel classifications from the ground-truth matrix. The results include accuracies for the SVM, MLP, and CNN models on the desktop PC, ODROID-C2, and Raspberry Pi 3B. The output images for each model are shown in Fig. 3. Comparing the output images in Fig. 3 with the ground-truth image in Fig. 1 reflects the accuracies of these models, which are summarized in Table III. Out of the 16 agricultural land classes identified in the ground-truth matrix, inter-class accuracies (percent of pixels that were correctly identified within the class) were also calculated and summarized in Table IV.

1. *Support Vector Machines*

The SVM model in this study performed with an accuracy of 62% on the desktop PC with an Intel i7-6700 vPro quad-core processor, and an accuracy of 61% on the embedded platforms with ARM Cortex-A53 quad-core processors. The model from [8] achieved an accuracy of 68% on a server with the Nvidia GeForce GTX 1080 and the Tesla K40c. The higher accuracy achieved by [8] is likely due to training the model on a GPU instead of a CPU. The reason for the disparity between the accuracy of the SVM on the desktop PC and the embedded platforms in this research is the random variations of model accuracies within the ten trials.



(a) Accuracy Benchmarks (b) Run-Time Benchmarks (c) Memory Benchmarks

Fig. 4. Accuracy, Run-Time, and Memory Benchmarks of Each Algorithm on All Tested Platforms

2. Multi-Layer Perceptron

The MLP model used in this study achieved an accuracy of 85% on the desktop PC and the embedded platforms, while the model from [5] achieved an accuracy of 82% on a desktop PC with dual Intel Xeon E5-2630 v2 processors and a Nvidia Tesla K20c GPU. The greater accuracy in this study, despite the use of a GPU in [5], is likely due to the usage of the latest version of Adagrad (which is the parameter update algorithm used for this model), as the study in [5] was conducted in 2016 [13].

3. Convolutional Neural Network

Lastly, the CNN model in this study performed with an accuracy of 97% on the desktop PC and the embedded platforms. The model from [5] had an accuracy of 96% on a desktop PC with dual Intel Xeon E5-2630 v2 processors and a Nvidia Tesla K20c GPU. Similar to the MLP, the discrepancies are likely due to the version differences of Adagrad.

Comparing the accuracies of the SVM, MLP, and CNN reveals that the SVM model provided the lowest accuracy, while the CNN model offered the best overall accuracy when identifying the pixels in the Indian Pines dataset. The inter-class accuracies of the SVM model also had a large range (97.8%) and standard deviation (38.3%). The massive spread of the SVMs inter-class accuracies can be attributed to the SVM being over-trained on the classes with a higher number of samples and therefore not able to identify classes with relatively fewer number of samples as accurately. By contrast, the deep-learning models had markedly better inter-class accuracies, even in classes with small sample sizes. This outcome resulted in the deep-learning models having a smaller range and standard deviation than the SVM. The range and standard deviation are 24.6% and 7.9%, respectively, for the MLP, and 39.2% and 13.7%, respectively, for the CNN.

B. *Run-Time Benchmarks*

Run-time benchmarks were recorded for each classification model and averaged over ten trials. The CNN was consistently the slowest on all platforms with a run-time of 205 seconds on the desktop PC, 1543 seconds on the ODROID-C2, and 3032 seconds on the Raspberry Pi 3B. The SVM was fastest on the desktop PC at 42 seconds, while the MLP was fastest on the ODROID-C2 and Raspberry Pi 3B at 390 seconds and 529 seconds, respectively. The run-time values are summarized in Fig. 4b.

1) Desktop PC vs. Embedded Platforms

An overview of the run-time benchmarks of the three models reveals that all three models ran faster on the desktop PC than the embedded platforms by at least a factor of ten. This outcome was expected due to the abundance of computational capacity available on a desktop PC compared to that of the embedded platforms. The Intel i7-6700 vPro quad-core processor on the desktop PC uses hyperthreading technology that enables the processor to run two threads in each core at once. By contrast, the ARM Cortex-A53 quad-core processor in the embedded platforms does not use that technology. As a result, the Intel i7-6700 vPro quad-core processor displays performance gains when compared to the ARM Cortex-A53 quad-core processor [14]. Apart from hyperthreading, it should also be noted that the desktop PC is clocked at 2.3 times the ODROID-C2 and 2.8 times the Raspberry Pi 3B. Higher clock speed, coupled with larger cache size and improved memory management technology present in the desktop PC, all further contribute to better performance on the PC.

The CNN was consistently the slowest algorithm on all platforms. The reason for this is two-fold: the computational cost of the convolutional layers in the architecture of the CNN model and the use of the spectral bands of a 27×27-pixel input patch to predict the class for each pixel. The SVM and MLP models only used the spectral bands of the pixel being predicted (i.e., surrounding pixels have no impact on the output).

Moreover, it should be noted that the SVM model ran faster than the MLP model on the desktop PC but slower than the MLP model on the embedded platforms. When conducting the final predictions, SVMs are inherently slower than MLPs since 16 one-versus-all SVMs have to be executed for prediction of each pixel in this dataset, whereas MLP only has to be executed once since the single model in Fig. 2a can act as a multi-class classifier. The much larger cache on the desktop PC enables execution of 16 one-versus-all SVMs during prediction, while the embedded platforms likely have to keep calling back to memory. The constant referencing to memory in the embedded platforms contributes to a jump in run-time for SVM, resulting in the prediction taking longer than MLP [14].

2) ODROID-C2 vs. Raspberry Pi 3B

While the ODROID-C2 and the Raspberry Pi 3B share the same architecture, the compatibility issues of the Tensorflow wheels with the operating systems must be noted. The armv7l kernel on the Raspberry Pi 3B is incompatible with the 64-bit architecture of the ARM Cortex-A53 quad-core processor. As a result, the 32-bit version of the Tensorflow wheel was used on the Raspberry Pi 3B, while the 64-bit version of the wheel was used on the ODROID-C2 (which has the aarch64 kernel) [15]. This setup explains why all models take more time for prediction on the Raspberry Pi 3B than the ODROID-C2.

C. *Memory Benchmarks*

On average, the MLP model consumed the least amount of memory on all platforms tested in this study (168 MB on desktop PC, 169 MB on ODROID-C2, and 128 MB on Raspberry Pi 3B). The average memory usage of the SVM model was the second highest, with these benchmarks: 178 MB on desktop PC; 168 MB on ODROID-C2; and 153 MB on Raspberry Pi 3B. Lastly, the CNN model had the highest memory usage (235 MB on desktop PC, 247 MB on ODROID-C2, and 201 MB on Raspberry Pi 3B). Memory usage was relatively consistent across all platforms. These results are summarized in Fig. 4c.

The percent differences of the memory usage between the platforms relative to the desktop PC are summarized in Table V. The memory benchmarks on the desktop PC and the ODROID-C2 differed by less than 6% from the memory benchmark on the desktop PC. However, the difference between the memory benchmarks on the desktop PC and the Raspberry Pi 3B were more than 14% greater than the benchmark on the desktop PC. The greater percent difference of the desktop PC versus Raspberry Pi 3B compared to the desktop PC versus ODROID-C2 can be attributed to the ODROID-C2 having nearly twice as much memory bandwidth as the Raspberry Pi 3B (4000 MB/s and 2000 MB/s, respectively) [15]. Lastly, SVM's one-vs-all classification method caused the model to use more memory than MLP, despite the complexity of the MLP model itself [16] [17].

D. *Discussion*

Evaluating the accuracy, run-time, and memory benchmarks across platforms reveals the best model for each

TABLE V. PERCENT DIFFERENCE OF MEMORY BENCHMARKS ON EMBEDDED PLATFORMS VS. DESKTOP PC

	SVM	**MLP**	**CNN**
Desktop PC vs. ODROID-C2	5.62%	0.60%	5.11%
Desktop PC vs Raspberry Pi 3B	14.04%	23.81%	14.47%

TABLE VI. ALGORITHM EVALUATION FOR EMBEDDED PLATFORMS

Accuracy per Second (%/sec)	**ODROID**	**Pi**	**Accuracy per RAM Used (%/MB)**	**ODROID**	**Pi**
SVM	0.11	0.09	SVM	0.37	0.40
MLP	**0.22**	**0.16**	MLP	**0.50**	**0.66**
CNN	0.06	0.03	CNN	0.39	0.48

platform. The abundance of processing power, memory capacity (RAM), and memory bandwidth on the desktop PC compared to the embedded platforms means that the accuracy of these HSI classification algorithms should be maximized using the CNN. Also, the CNN is recommended for the PC due to the high accuracy of 97% that was achieved at speeds 7.5 times faster than the fastest embedded platform (ODROID-C2). Despite the two embedded platforms used in this study having the same architecture, the differences in memory management of the processors, kernels, clock speeds (1.5 GHz on ODROID-C2 vs 1.2 GHz on Raspberry Pi 3B), and RAM (2 GB on ODROID-C2 vs 1.2 GB on Raspberry Pi 3B) contributed to the performances of the models on these platforms being different. The ODROID-C2 having more memory bandwidth than the Raspberry Pi 3B resulted in the models taking longer to conduct inference on the Raspberry Pi 3B. Furthermore, the 32-bit TensorFlow wheel used on the Raspberry Pi 3B due to compatibility issues with the kernel also contributed to longer run-times. Lastly, the older DDR2 RAM on the Raspberry Pi 3B may also have contributed to models running relatively slowly on the platform, as the newer generations of DDR3 RAM present on the ODROID-C2 and desktop PC are much faster. The accuracy-per-time and accuracy-per-memory metrics, shown in Table VI, were maximized in order to select the best algorithm for each embedded platform. The MLP model was determined to be the best algorithm for the ODROID-C2 and the Raspberry Pi 3B, as it showcased the shortest run-times (390 seconds and 529 seconds, respectively) and the second-highest accuracy benchmark of 85%. An increase of 12% in accuracy, in the authors' opinion, is not justified for the CNN on the embedded platforms due to the massive run-time increase of 400% for the ODROID-C2 and 570% for the Raspberry Pi 3B.

V. CONCLUSIONS

Hyperspectral imaging in space can reveal much useful information about our world. HSI analysis techniques have been developed and are often executed on computationally tractable environments on Earth. However, conducting these analyses in computation-constrained environments on-board a spacecraft would be extremely beneficial, enabling users to intelligently downlink a subset of data rather than the entirety. Benchmarking different machine-learning algorithms for HSI analysis on different platforms with varying performance capabilities allowed the authors to determine the best algorithms to run on embedded platforms.

SVM, MLP, and CNN models were benchmarked on a popularly used Indian Pines HSI dataset. The models were all trained on the desktop PC. Accuracy, run-time, and memory benchmarks were collected on the desktop PC, ODROID-C2, and Raspberry Pi 3B platforms for the final prediction of pixel classifications in the hyperspectral image. The desktop PC has a more powerful processor than the embedded platforms and as a result had the best accuracy and run-time benchmarks. Considering the relative abundance of processing power on the desktop PC and the benchmarks collected, it is evident that the CNN model is the best model of the three models investigated in this study. However, the increase in accuracy was not justified on the embedded platforms, due to a substantial increase in run-time. As a result, the MLP model was selected to be the optimal model to run on embedded platforms with constrained performance capability.

VI. FUTURE WORK

The next steps are to conduct this HSI classification on other embedded platforms, such as the Digilent ZedBoard to leverage the ARM Cortex-A9 architecture that is currently being used in space apps [18]. Performing this analysis on an embedded GPU is another avenue to explore more efficient methods of conducting HSI analysis on resource-constrained platforms.

ACKNOWLEDGMENTS

This research was funded by the NSF SHREC Center (c/o IUCRC grant CNS-1738783), Swanson School of Engineering, and the Office of the Provost at the University of Pittsburgh.

REFERENCES

[1] A. J. Pellish, *Radiation 101: Effects on Hardware and Robotic Systems,* MD: NASA Goddard, 2015.

[2] HySpex, *Hyperspectral Imaging,* Oslo, Norway: Norsk Elektro Optikk, 2016.

[3] K. Hege, D. O'Connell, W. Johnson, S. Basty and E. Dereniak, "Hyperspectral imaging for astronomy and space surveillance," *Proceedings of SPIE - The Internation Society for Optical Engineering,* vol. 5159, pp. 380-391, 2004.

[4] A. Shmilovici, "Support Vector Machines," *In Data mining and knowledge discovery handbook,* pp. 231-247, 2009.

[5] A. Santara, K. Mani, P. Hatwar, A. Singh, A. Garg, P. Kirti and P. Mitra, "BASS Net: Band-Adaptive Spectral-Spatial Feature Learning Neural Network for Hyperspectral Image Classification," *arXiv,* vol. 1612, 2016.

[6] K. Makantasis, K. Karantzalos, A. Doulamis and N. Doulamis, "Deep Supervised earning for Hyperspectral Data Classification through Convolutional Neural Networks," *IEEE International Geoscience and Remote Sensing Symposium (IGARSS),* pp. 4959-4962, 2015.

[7] H. Petersson, D. Gustafsson and D. Bergstrom, "Hyperspectral Image Analysis Using Deep Learning - A Review," *2016 Sixth International Conference on Image Processing Theory, Tools, and Applications (IPTA),* 2016.

[8] X. Cao, F. Zhou, L. Xu, D. Meng, Z. Xu and J. Paisley, "Hyperspectral Image Classification with Markov Random Fields and a Convolutional Neural Network," *IEEE Transactions on Image Processing,* vol. 27, no. 5, pp. 2354-2367, 2018.

[9] M. Baumgardner, L. Beihl and D. Landgrebe , "220 Band AVRIS Hyperspectral Image Data Set," *Purdue University Research Repo,* 2015.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel and e. al., "SciKit Learn: Machine Learning in Python," *JMLR,* vol. 12, pp. 2825-2830, 2011.

[11] M. Abadi, A. Agarwal, P. Barham, E. Brevdo and e. al., "TensorFlow: A System for Large-Scale Machine Learning," *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16),* pp. 265-283, 2016.

[12] W. Hu, Y. Huang, L. Wei, F. Zhang and H. Li, "Deep Convolutional Neural Networks for Hyperspectral Image Classification," *Journal of Sensors,* pp. 1-12, 2015.

[13] S. Ruder, "An Overview of Gradient Descent Optimization Algorithms," *axXiv,* vol. 1609, 2017.

[14] VERSUS, "ARM Cortex-A53 vs Intel Core i7-6700 | Mobile chipset comparison," 2018. [Online]. Available: https://versus.com/en/arm-cortex-a53-vs-intel-core-i7-6700. [Accessed 22 09 2018].

[15] M. Plauth and A. Polze, "Are Low-Power SoCs Feasible for Heterogenous HPC Workloads?," in *Euro-Par 2016: Parallel Processing Workshops Lecture Notes in Computer Science,* Grenoble, France, Springer, 2017, pp. 763-774.

[16] E. Mizutani and E. S. Dreyfus, "On Complexity Analysis of Supervised MLP-learning for Algorithmic Comparisons," *International Joint Conference on Neural Networks (IJCNN'01),* vol. 1, pp. 347-352, 2001.

[17] M. Gardner and S. Dorling, "Artificial Neural Networks (the multilayer perceptron) - a review applications in the atmospheric sciences," *Atmospheric Environment,* vol. 32, no. 14-15, pp. 2627-2636, 1998.

[18] C. Wilson and A. D. George, "CSP Hybrid Space Computing," *Journal of Aerospace Information Systems,* vol. 15, no. 4, pp. 215-227, 2018.