Embedded High Performance Computing for Hyperspectral Image Classification

Submitted in partial fulfillment of the requirements of the degree of
Master of Technology in
Geoinformatics and Natural Resources Engineering

Submitted by
Pankaj H. Randhe
143310018

Under the guidance of
Prof. Surya S. Durbha

Centre of Studies in Resources Engineering
Indian Institute of Technology Bombay
2016

Declaration

I, hereby, declare that this report represents my ideas in my own words. Also the works of other person where have been discusses or included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Date: 01/07/2016

Pankaj H. Randhe

Approval Sheet

The dissertation report entitled "Embedded High Performance Computing for Hyperspectral Image Classification" prepared by Pankaj H. Randhe (Roll No. 143310018) under the guidance of Prof. Surya S. Durbha is hereby approved for submission.

Examiner(s)

P. Venkatachalam

S. N. J.

Supervisor

Surya DL

Chairperson

P. Venkatachalam

Date: 01/07/2016

Place: IIT Bombay

Acknowledgement

I am grateful to my supervisor Prof. Surya S. Durbha for his constant guidance and motivation while working on this topic. I extend my sincere thanks to him for his constant help throughout the duration of this project. I am also thankful to Mr. Suryakant Sawant for his help in designing 3D enclosure for the prototype. I am grateful to the Centre of Studies in Resources Engineering, for providing me all necessary support while working on this topic.

Pankaj H. Randhe
Roll No: 143310018

Abstract

Jetson TK1 is a recently launched embedded application development platform from NVIDIA, which features the Tegra K1 processor and Kepler Graphics Processing Unit (GPU). We envisage that such a system has huge potential for deploying an embedded system for on-board classification of hyperspectral images. We used a convolutional deep neural network for designing a unified model for hyperspectral image classification. Deep convolutional model hierarchically extracts spectral-spatial features from hyperspectral imagery. These features are then used by the fully connected layer of neural network to perform pixel level classification of hyperspectral imagery. Our experimental results show that Jetson TK1 based hyperspectral image classification gives promising results, and the possibility of having Jetson based embedded platform for on-board classification of hyperspectral images.

Keywords— Deep learning, convolutional neural networks, Jetson TK1, Hyperspectral image classification

Contents

List of Tables

List of Figures

Nomenclature

CPU	Central Processing Unit
GPU	Graphics Processing Unit
CUDA	Compute Unified Device Architecture
HSI	HyperSpectral Image
CNN	Covolutional Neural Network
SVM	Support Vector Machine
PCA	Principal Component Analysis
SIMT	Single Instruction Multiple Thread
SGD	Stochastic Gradient Descent

Chapter 1

Introduction

Data obtained from the Earth Observation systems such as satellite borne or air borne sensors plays a significant role in various disciplines. In particular, hyperspectral data has very high spectral resolution and therefore holds a great potential for detailed surface studies. However, high spectral resolution of hyperspectral data gives rise to high dimensionality, and hence the amount of data generated becomes enormous. Processing of such a huge data is computationally intensive task. In many applications, having the necessary information calculated in near-real time is highly desirable. Various high performance computing (HPC) approaches have been used to reduce the computation time for hyperspectral data processing. But in order to achieve on-board and on-site processing of hyperspectral data, embedded system based high performance computing is desired.

Arrival of GPU enabled platforms such as NVIDIA Jetson TK1 has opened up new avenues for near-real time on-board and on-site processing of Hyperspectral data by harnessing the power of GPU accelerated computing. The NVIDIA Jetson TK1 Development kit is the world's first mobile supercomputer for embedded systems.

1.1 Hyperspectral Remote Sensing

In hyperspectral remote sensing, images are acquired with high spectral resolution. In such images each pixel represents a spectra consisting of large number of narrow and continuously spaced spectral bands (Figure: 1.1). Hyperspectral imagery provides opportunities to extract more detailed information than is possible using traditional multi-spectral data.

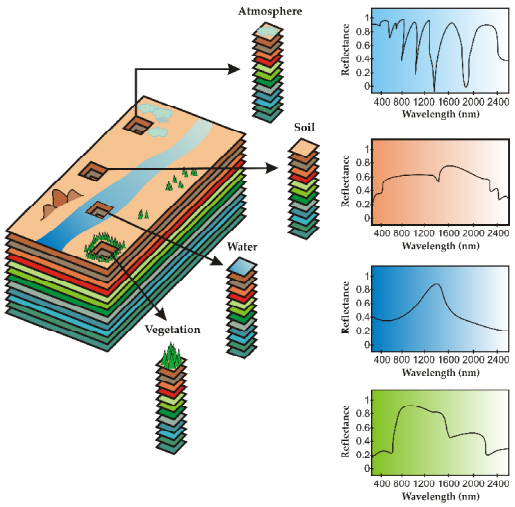


Figure 1.1: Concept of Hyperspectral Imaging (Source: [27])

The wavelength region in hyperspectral images generally covers spectral regions such as visible, near infrared, shortwave infrared, midwave infrared and longwave infrared.

1.2 Embedded HPC for Hyperspectral Image Classification: Motivation

In many scenarios such as detection of hostile weaponry for defense personnel, detection of oil spills caused by marine vessels, crop health analysis for precision farming, disaster mitigation and management, near real-time classification of the data is highly desirable [5].

The embedded GPU accelerated platform such as NVIDIA Jetson TK1 has opened up new avenues to create embedded system applications for on-board and on- site image analysis. Such a system can be useful for performing dynamic processing of hyperspectral data in the field. In order to perform onboard processing of remote sensing data on airborne devices and satellites, there are constraints on the power and energy consumption due to battery lifetime or limited power from solar panels. The fine spectral resolution and high dimensionality of hyperspectral images justifies the need of high-performance computing. Low-power technologies along with energy-aware novel computational algorithms can produce a response in real or near real time, keeping the power/energy usage to minimum [10].

The Jetson TK1 embedded platform provides a low-power and low-weight specialized hardware for high performance computing. The Tegra K1 processor on NVIDIA Jetson features Kepler GPU, and hence makes Jetson TK1 a suitable choice for running parallel versions of the algorithms, which are computationally intensive otherwise. In this work using the JetsonTK1 platform we have built a stand-alone system for hyperspectral data processing. This system consists of NVIDIA Jetson TK1 with interactive display. The system hosts an application for near real time hyperspectral image classification.

1.3 Problem Statement

In this project we explored deep learning based algorithm for hyperspectral image classification and implemented it on the embedded Jetson TK1 platform. Deep learning based algorithms are computationally intensive, and therefore need the high performance computing approaches. Jetson TK1 platform with Kepler GPU is a suitable choice for deploying deep learning algorithms.

The aim of this work is to develop and implement a parallel algorithm for supervised hyperspectral image classification, deploy and test it on Jetson TK1 and check the feasibility of it for real time or near-real time classification of hyperspectral data, while minimizing power/energy usage.

The core objectives of this work are as follows:

- To use JetsonTK1 as a platform to build embedded system applications for hyperspectral image classification. We envisage such a system as a potential solution for on-site and on-board classification of hyperspectral data in near-real time. To the best of our knowledge, embedded HPC applications based on Jetson TK1 has not yet been explored for remote sensing image classification tasks

- To evaluate and measure the computational time of the algorithm for hyperspectral image classification

- To measure the power/energy usage of the Jetson TK1 board for hyperspectral image classification algorithm

- To develop a stand-alone Jetson TK1 based HPC Tablet prototype with interactive touch screen running hyperspectral image classification application running on top of it

1.4 Report Structure

The project report is organized in the following manner.

- Chapter 2 presents the review of existing literature available on embedded system based approaches for remote sensing data processing. The chapter also explores the existing literature on various deep learning approaches, which have been used for hyperspectral image classification.

- Chapter 3 discusses the methodology we have used for performing embedded classification of hyperspectral images.

- Chapter 4 presents the discussion on the datasets used for the study and results obtained for Jetson TK1 based hyperspectral image classification.

- Chapter 5 concludes the report with conclusion of the work, and the scope for further improvements in the developed system.

Chapter 2

Literature Review

This chapter explores and reviews the available literature on embedded computing approaches for remote sensing image classification, and existing deep learning based methodologies for hyperspectral image classification. The chapter also discusses by means of literature review the need of high performance computing for deep learning based algorithms.

2.1 Embedded Computing for HSI classification

Hyperspectral data is highly voluminous and hence down linking, storing and processing of such huge data has always been a challenging task. Embedded high performance computing solutions for hyperspectral image classification can make it possible to use such systems for on-board and on-site classification where rapid processing is required for near-real time result generation.

On-board classification of hyperspectral images can be useful in the following ways.

- On board classification of hyperspectral data can be very useful as it can process hyperspectral data on-board and only the thematic classified maps can be sent to the ground stations.

- On-board image classification of hyperspectral data can lead to reduced down-link data transmission by sending only the thematic or classified maps and can be useful in the scenarios where rapid decision making is required.

- On-board image classification can be useful in tasking the on-board sensors on the fly in response to the dynamic events and changing needs.

Embedded system based solutions have been explored for on-board processing of remote sensing data for various purposes.

Steve Chien, et al. (2009) [9] described various on-board image classification algorithms for Earth Observing One (EO-1) spacecraft for Hyperion hyperspectral data for volcano, flood and cryosphere analysis. The EO-1 uses Autonomous Science Craft Experiment (ASE) [39] which is on-board flight software that has components for science processing, mission planning, and onboard execution of the algorithms. These components enable the EO-1 satellite to perform various on-board processing tasks. The science-processing component is responsible for rapid downlink, downlink of alerts, and onboard decision making. The onboard mission-planning component facilitates EO-1 to change future observations, by tasking the sensors, based on detected events. The onboard execution component is responsible for the mission plans. EO-1

deployed an onboard algorithm for thermal detection and summarization. This algorithm uses the infrared bands of the Hyperion instrument to evaluate images for very hot signatures. The EO-1 is deployed with onboard flood classification algorithm for onboard recognition of major flooding events, and also onboard classification algorithm for Snow, Water, Ice, and Land classification.

The HyspIRI mission [40] by NASA has a hyperspectral and thermal infrared imaging capability on-board. With HyspIRI mission NASA aims at achieving rapid delivery of science products. NASA is exploring direct broadcast methodology in which instrument downlinks data as it is being acquired so that the ground stations can receive it in near real time for rapid processing. The data to be sent by the Hyspiri mission exceeds the downlink capacity by approximately 65x [9]. To deal with this problem, NASA is exploring the options for performing onboard processing so that the produced results, being considerably small in size than the raw data, can be fit within the downlink capacity, and sent to the ground stations by direct broadcasting, minimizing the latency in the availability of the datasets.

Halle W., et al. (2002) [10] proposed an on-board image classification system using neural network classifier for BIRD (Bispectral InfraRed Detection) mission for the fire recognition and the detection of hot spots from the multi-spectral images acquired by IR-Sensors and the stereo line scanner WAOSS (Wide-Angle-Optoelectronic-Scanner) on board [10]. BIRD system used on-board data processing system for reducing the data to be down linked from the satellite by generating the thematic maps from the multispectral classification algorithm. BIRD system used a special hardware having the neural network processor called NI1000. The NI1000 chip was integrated into the dedicated computer for Payload Data Handling System. The classification system discussed by Halle W. et al. has two parts: one for on-board classification and other is for on-ground classification for verification. They used radiation values from the sensors as features for performing the classification. It was suggested that such on-board system is advantageous because it speeds up the availability of the required information for the event of fire detection. The system eliminates the time required for on-ground data processing procedure usually performed at ground stations. Also the authors suggested that the on-board system results in the data reduction, which ultimately can help in lowering down the cost of data transmission to the ground stations.

Paula J., Pingree (2010) [11] discussed Support Vector Machine (SVM) based image classification algorithm, with reconfigurable FPGA technologies, for NASA's onboard image processing capabilities. They explored hybrid FPGAs like Xilinx Virtex-4FX60 and Virtex-5, for running diverse software applications on embedded processors taking advantage of reconfigurable hardware resources. Such hardware/software designed systems have low power requirements, and bears low cost than general-purpose SBCs. JPL have developed and deployed support vector machine (SVM) classification algorithms on board spacecraft to identify high-priority image data for down linking to the ground stations [11]. These algorithms in turn help in

on-board data analysis to enable rapid reaction to dynamic events. The authors described that SVM classification algorithms are deployed on-board using Single Board Computational platforms such as the RAD6000 and Mongoose V processors. However, these legacy processors on SBCs have only limited computing power, and hence can operate only on subset of the hyperspectral image bands. It was suggested that FPGA coprocessors as an ideal candidate for implementations of onboard classification algorithms, and can provide significant improvement in capability and accuracy of onboard classification systems. According to the authors SVMs are well suited for onboard classification applications. Further, it was suggested that the asymmetry of computational effort in the training and testing stages of SVM makes them ideal for on-board classification. Classifying new data points requires orders of magnitude less computation than training because training a SVM classifier requires solving a quadratic optimization problem which requires $O(n^3)$ operations, where n is the number of training examples. On the other hand, classifying a new data point with a trained SVM requires only O(n) operations. The classification system was deployed on embedded PowerPC 405 processor using the Xilinx EDK tools. FPGA hardware was used for streaming the image data to the software. The SVM hardware process performs the SVM operation on the image and the results were written back to the PowerPC 405 processor.

G. Leon, et al. (2015) [6] studied the emerging low-power multi threaded architectures from ARM and NVIDIA and suggested that these can be a practical approach for hyperspectral image processing as compared to a standard high-performance multi core processor. The experimental results showed that low-power GPUs deliver reasonable performance and high power/energy gains. The authors suggested that small GPUs from NVIDIA in the Jetson TK1 embedded development kit feature high hardware parallelism and found them appropriate for data-parallel operations. The authors performed the experiments on a variety of stand-alone computing platforms and a conventional server equipped with recent processors from ARM, Intel and NVIDIA. The hyperspectral image used by the authors for experiments was collected by the AVIRIS instrument over the World Trade Center area in New York City on September 16, 2001. Experiment results demonstrated that low-power GPUs could deliver reasonable performance and high power/energy gains. They implemented the RX detector on low power GPU platforms and evaluated their performances in terms of power and energy consumptions. The study is relevant for remote sensing applications because, the RX detector is composed of basic dense linear algebra operations which are also present in other remote sensing algorithms such as, estimation of the number of end members, end-member extraction, dimensionality reduction or abundance estimation. The authors suggested that the results from their experiments with low-power architectures carry beyond the RX detector, and are applicable to many compute bound algorithms for remote sensing applications.

2.2 Deep Learning Approaches for Hyperspectral Image Classification

Deep learning refers to the family of methods that uses deep architectures to learn high-level feature representations hierarchically [28].

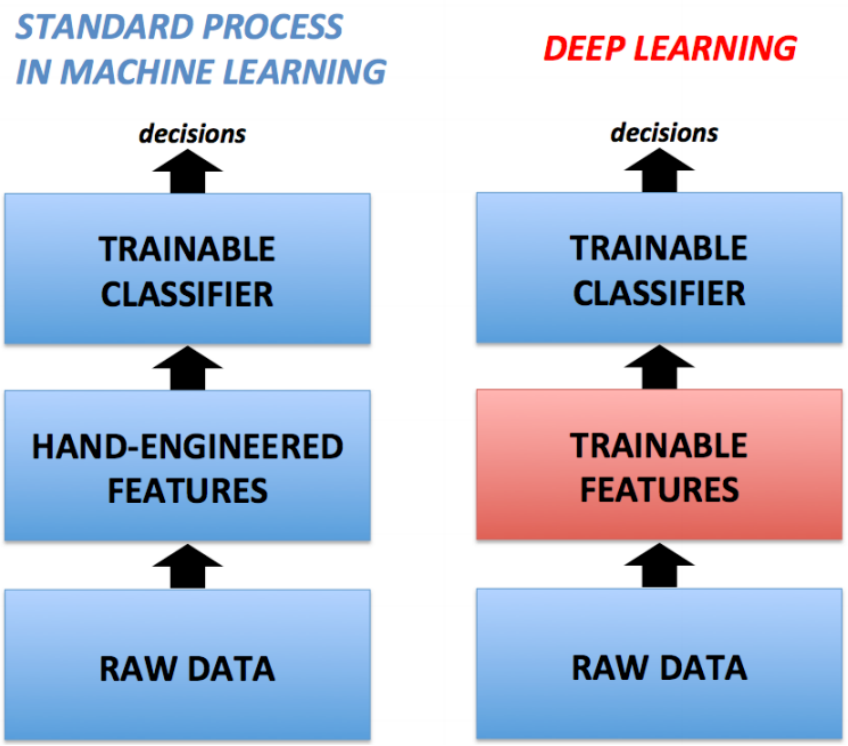


Figure 2.1: Standard Machine learning v/s Deep Learning (Source: [28])

In general, during the classification approaches spectral and spatial features are extracted from images before performing classification as these help in increasing the discriminative capacity of the classifier. However, deep neural networks can be used to build high level features (Figure: 2.1) hierarchically. This technique can be very useful where there is a need for a unified or end-to-end framework for image classification [2]. Recently, some applications of deep learning for hyperspectral image classification is demonstrated in which, spectral vectors [7], spatial and spectral information [1] [2], structured features resembling various spectral band-pass filters [3], stacked auto encoders [8] are used.

Romero et al. (2014) [1] used the method of deep feature extraction based on convolutional neural networks for classifying the Indian Pines hyperspectral imagery. The authors have successfully demonstrated the performance of the extracted representations in a challenging AVIRIS hyperspectral image classification problem, and compared the results with standard

dimensionality reduction methods like principal component analysis and kernel principal component analysis.

According to the authors, deeper the network higher is the accuracy. They also found out that including the max-pooling stage after the convolutional layer is extremely beneficial. The authors employed the architecture with smallest possible symmetric receptive field 3×3 pixels and 200 outputs per layer. The results obtained by the authors show that networks can extract powerful discriminative features, when the receptive field takes into account the neighboring pixels. The performance gain with respect to the number of layers is upper-bounded because each pooling stage reduces the spatial resolution which results in spatially coarse output features. Results by the authors reveal that convolutional neural networks are very effective at encoding spectral-spatial information from the images.

Makantasis, Konstantinos, et al. (2015) [2] used deep convolutional neural networks along with multilayer perceptron for hyperspectral image classification. According the authors, CNNs stand out because they do not need complex handcrafted features. These networks can hierarchically construct high-level features, in an automated way. For handcrafted features it is rarely known which features are important for the problem at hand. They compared their experimental results against RBF kernel SVM and Linear kernel SVM, firstly using the raw hyperspectral data and secondly, with the reduced data using principal components. They found out that CNN based classifier present superior classification accuracy in all datasets.

Viktor, et al. (2015) [3] proposed CNN model which was able to learn structured features, which roughly resembled with different spectral band-pass filters. The authors carried out their experiment on the Indian Pines hyperspectral image dataset and used hyperbolic tangent activations in all layers and, the soft-max function in the last layer. They found out that using dropout regularization in the fully connected layers, does not significantly improve the classification accuracy. The network was trained with batch size of 50 samples using the gradient descent. The input to the network consisted of the eight-connected neighborhood of a hyperspectral pixel, to account for the spatial information context. They also used a simple augmentation technique for hyperspectral data to elevate the number of training samples. The augmentation technique used the per-spectral-band standard deviation of the samples in the training set which belong to the class, to expand the number of the training data. The augmentation technique found out to be useful by the authors, since it resulted in the improvement in the classification accuracies.

Wei Hu, et al. (2015) [7] suggested CNN model where every pixel in the hyperspectral image is regarded as a 2 dimensional image whose height is equal to 1, and width is equal to the number of bands or the feature vectors. The authors did not consider the spatial correlation and only concentrated on the spectral signatures of the pixels.

Chen, Yushi, et al. (2014) [8] verified the appropriateness of stacked auto-encoders by following classical spectral information-based classification. In addition authors have proposed a new way of hyperspectral image classification, which used spatial information along with the spectral information. The authors proposed a deep learning framework to merge spatial and spectral information. The model architecture used by the authors is a hybrid of principal component analysis (PCA), deep learning architecture, and logistic regression. They used stacked auto-encoders (Figure: 2.2) for getting useful high-level features. For spectral information, a raw pixel of the hyperspectral image is taken into consideration. For spatial information, authors extracted the first several principal components of a pixel and its neighborhood region to get the spatial information.

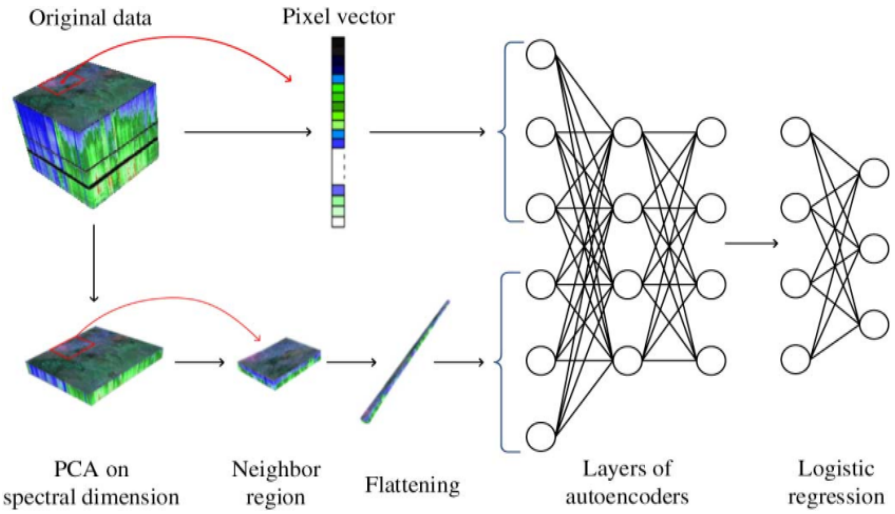


Figure 2.2: Stacked Auto Encoders used by Chen, Yushi, et al. (Source: [8])

The authors compared their results with SVM based classifier where extended morphological profile (EMP) used for extracting the spatial information, and found that stacked auto encoders with logical regression gives consistently reach higher accuracy than the EMP with SVM classifier.

2.3 Deep Learning and High Performance Computing

In the previous section we reviewed the methods, which used deep learning methods for hyperspectral image classification. Deep learning methods are generally compute intensive and need a high performance computing architecture for near real time image processing applications such as embedded classification of hyperspectral images on-board the satellites or airborne vehicles.

Benefits of Deep learning based approaches are as follows:

- Robust
  No need to design the features ahead of time because features are automatically learned and robust to natural variations in the data [26]

- Generalizable
  The same neural net approach can be used for many different applications and data types [26]

- Scalable
  Performance improves with more data and method is massively parallelizable [26]

In this study a Convolutional Neural Network, one of the deep learning approaches, is used for designing the classifier for hyperspectral image classification. Convolutional neural networks are good candidates for GPU based parallel computing. In convolution layers, feature maps are independent of each other and each feature map uses same set of weights hence network can learn in parallel [18]. Training CNN on GPU takes advantage of shared memory of GPU to reduce the communication overhead between CNN units.

At their core convolutional neural networks use the dense matrix-matrix multiplications. In neural networks output activations of the neurons are calculated by multiplying the weight matrix with the input activations matrix (Figure: 2.3). The GPUs are good candidates for the matrix-matrix multiplications because matrix-matrix multiplications are highly parallel operations.

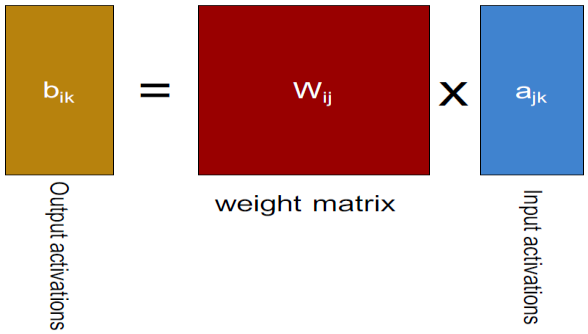


Figure 2.3: Matrix-matrix multiplication in neural networks (Source: [26])

In addition to the matrix-matrix multiplications, another parallelizable part of the convolutional neural networks is the convolution operations. The convolution operation (Figure: 2.4) can be regarded as the iterative operation with 6 dimensional loops.

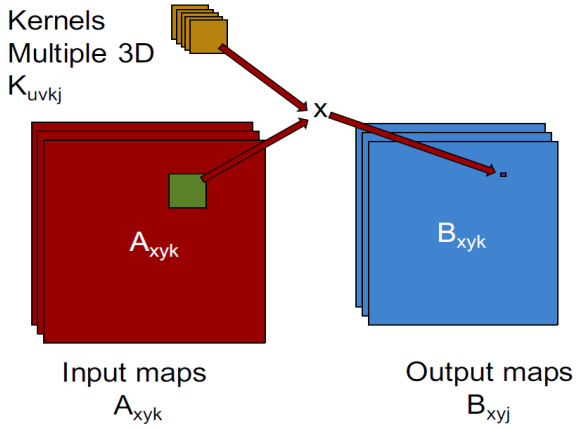


Figure 2.4: Convolution Operation (Source: [26])

In the 6D loop described below, input map can be the input data or the feature maps generated from the previous convolution layer.

6D Loop:
for each output map j
for each input map k
for each pixel x, y
for each kernel element u, v
$B_{xyj} += A_{(x-u)(y-v)k} * K_{uvkj}$

In neural networks, batching leads to the re-use of weights resulting in matrix-matrix multiplications rather than matrix-vector multiplications because in batching same weights are shared among all the samples of the batch.

Chapter 3

Methodology

This chapter discusses the hardware, tools and algorithm that have been used for performing embedded hyperspectral image classification.

3.1 Jetson TK1 Embedded Platform

Jetson TK1 is NVIDIA's embedded Linux development platform having Tegra K1 SOC (CPU+GPU+ISP in a single chip). Jetson TK1 comes pre-installed with Linux4Tegra Operating System, which is modified version of Ubuntu 14.04 with pre-configured drivers. Tegra K1 is NVIDIA's latest mobile processor. It comprises of a Kepler GPU with 192 cores, an NVIDIA 4-plus-1 quad-core ARM Cortex-A15 CPU with capacity of over 300 GFLOP/s of 32-bit floating point computation, integrated video encoding and decoding support, image/signal processing, and many other system-level features [29]. Due to its inherent support for image processing, Jetson TK1 has potential to be used for hyperspectral image processing. Tegra TK1's power consumption is in the range of 5 Watts for real workloads [29] and thus has the potential to be used as an onboard computing platform.

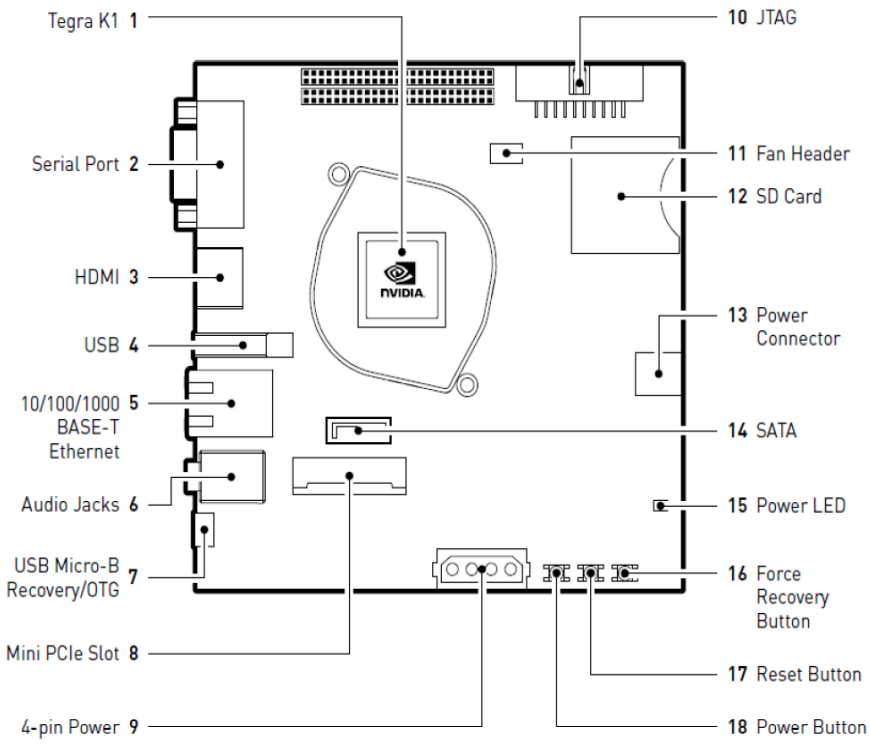


Figure 3.1: Jetson TK1 architecture
(Source: http://linuxgizmos.com/nvidia-jetson-tk1-most-advanced-hacker-sbc/)

The board is 5″ wide by 5″ long single board computer (SBC) with, 2 GB of RAM, 16 GB 4.51 eMMC memory (Figure: 3.1). It provides GPU accelerated low-power and low-weight specialized hardware for high performance computing. With Tegra K1 processor on board, Jetson TK1 features Kepler GPU making Jetson TK1 a suitable choice for running algorithms which need substantial parallel computing and need to be implemented on low power platforms.

Since its release, Jetson has been used for compute-intensive embedded projects such as drones, autonomous robotic systems, mobile medical imaging, and intelligent video analytics [13]. Jetson has been extensively used for computer vision applications such as face and gesture recognition, due to its high performance and low-energy support for deep learning. Further, it is also used to process imagery from high-definition cameras, and simultaneously perform world-mapping and obstacle detection operations on a 180° Light Detection and Ranging (LIDAR) for autonomous navigation [14]. G. Leon, et al. [24], investigated power consumption of Jetson TK1 board for RX anomaly detector, and the experimental results showed that Jetson's GPUs deliver reasonable performance and high power/energy gains.

3.2 Deep learning stack on NVIDIA's Jetson TK1

We used the deep learning stack (Figure: 3.2) for developing the model for hyperspectral image classification. It includes the hardware, libraries and frameworks.

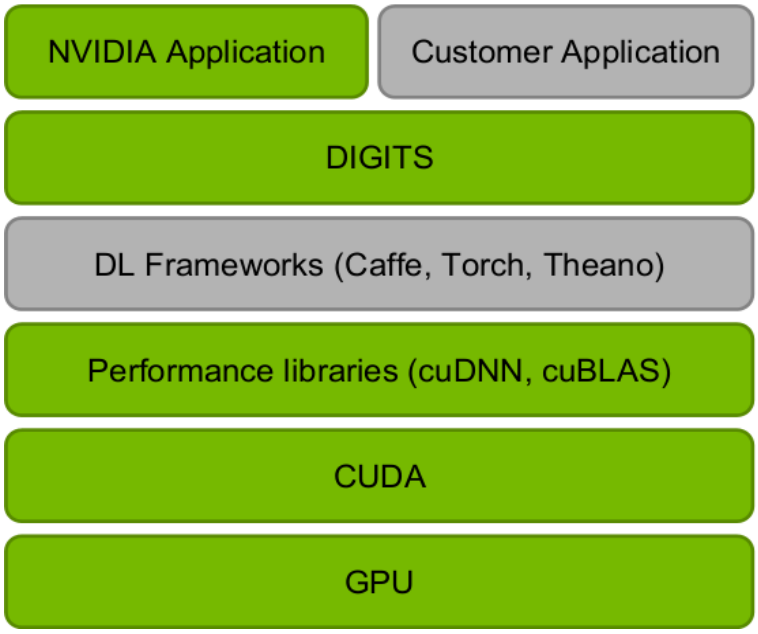


Figure 3.2: NVIDIA Deep Learning Stack (Source: [28])

3.2.1 Graphic Processing Unit

GPU computing is the use of GPU together with CPU to accelerate general purpose and scientific applications [30]. Hyperspectral image processing is a good candidate for GPU based computing due to high dimensionality of data. GPU computing, due to its parallel computing capabilities, gives many times faster results as compared to CPU based processing. It is the many-core architecture of GPU, which provides the large number of threads to achieve parallel processing while the CPU has limited number of cores optimized for serial processing [30].

The reason behind the discrepancy in floating-point capability between the CPU and the GPU is that the GPU is specialized for compute-intensive, highly parallel computation - exactly what graphics rendering is about - and therefore designed such that more transistors are devoted to data processing (Figure: 3.3) rather than data caching and flow control.

Figure 3.3: CPU v/s GPU (Source: [31])

For running an algorithm on a GPU, the algorithm has to be optimized and modified in way that it can be run on GPU and can take an advantage of GPUs parallel computing capabilities. To achieve this GPUs are provided with their programming APIs.

- **General Architecture of GPU**

The NVIDIA GPU (Figure: 3.4) at its core consists of the set of multi-processors and each multiprocessor has a set of parallel thread processors. GPU interacts with CPU using its own instruction set and has its own device memory. The thread processors run synchronously using the Single Instruction Multiple Thread (SIMT) strategy meaning all the processors execute same program on the different data in parallel [32].

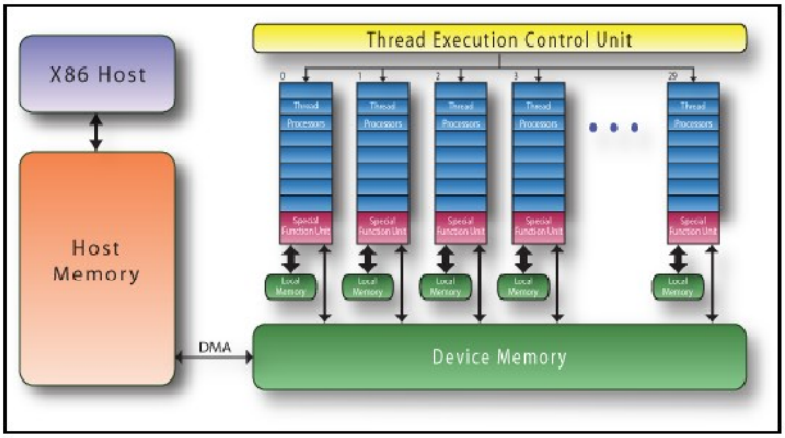


Figure 3.4: General architecture of GPU (Source: [32])

GPU implements the hierarchical memory pattern. The GPU has its own memory called as device memory, which could range up to 4GB today [32]. GPU programs typically require streaming access to large data sets that would overflow the size of a reasonable cache. The way in which GPU achieves performance better than CPU is attributed to its memory handling strategy. Generally the memory accesses slowdown the processing of CPU, to deal with this CPU uses cache memory to speed up the data fetching. But this strategy is not efficient for the GPU because GPU requires streaming access to the large datasets, which cannot be accommodated in the cache. The GPU deals with this problem by multi-threading approach. When the thread requires memory access it is blocked until the value is fetched form the memory. In multi-threading paradigm GPU executes another thread meanwhile the earlier thread is doing the memory operations, achieving high degree of parallelism.

The multiprocessor SIMT unit creates, manages, schedules, and executes threads in groups of 32 parallel threads called warps [33]. Warps are the primitive unit of scheduling in GPU [34].

The major difference between the CPU and GPU is that GPU, unlike CPU, doesn't rely on the large caches for the performance improvement rather they increase the performance by using multi-threading [34]. As in SIMT architecture, the same program is executed for each data element, there is a lower requirement for sophisticated flow control, and because it is executed on many data elements and has high arithmetic intensity, the memory access latency can be hidden with calculations instead of big data caches [35].

- **Architecture of Kepler GPU on NVIDIA's Jetson TK1**

As discussed earlier Jetson TK1 board hosts the NVIDIA's 192 core GPU with Kepler micro-architecture (Figure: 3.5).

Figure 3.5: Kepler GPU Micro architecture (Source: [41])

Kepler micro-architecture contains L1/L2 Unified Cache, 64 KB Configurable Shared Memory and L1 Cache, 48 KB Read-Only Data Cache and 1536 KB L2 Cache. Kepler is designed to maximize computational performance with superior power efficiency. In Kepler architecture warp consists of 32 parallel threads and each streaming multi-processor can handle 4warps and 8 instruction dispatch units where each warp allows two independent instructions per cycle. The Kepler micro-architecture distinguishes itself from the old architecture because of its dynamic parallelism feature which enables the Kepler GPUs to dynamically spawn new threads by adapting to the data without going back to the host CPU. This enables more of a program to be run directly on the GPU, as kernels now can independently launch additional kernels dynamically as needed. The Kepler GPU architecture is 3X times more power efficient compared to the Fermi architecture [41] and thus can be a suitable choice for power constrained environments.

3.2.2 GPU Programming Model CUDA

CUDA is a general-purpose parallel computing platform and programming model and comes with a software environment that allows developers to use C as a high-level programming language and also other languages such as C++, FORTRAN, Java, Python, etc. CUDA at its core have three key abstractions - a hierarchy of thread groups, shared memories, and barrier synchronization - that are simply exposed to the programmer as a minimal set of language extensions [35].

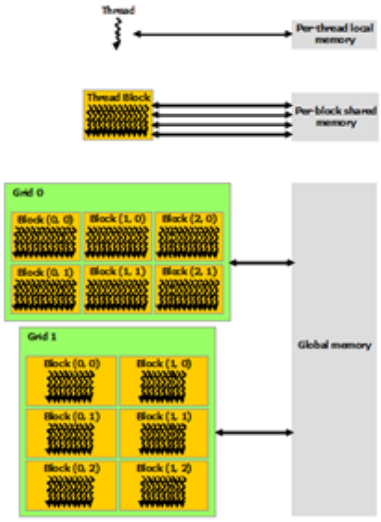


Figure 3.6: CUDA Thread Hierarchy (Source: [31])

The CUDA programming model (Figure: 3.6) follows simple hierarchy at software as well as hardware level. Each thread is executed on the streaming processor (SP) and has its own local memory and registers. The group of threads together builds a thread block and thread blocks are executed on multiprocessors. Threads in the thread block have access to the shared memory of the block. Several concurrent thread blocks can reside on a multiprocessor constrained by multiprocessor resources. The kernel (which is executed by the host) launches the grid, which comprises several thread blocks and only one kernel can be executed on each device at a time.

3.2.3 CuDNN Performance Library for Deep Learning

cuDNN (CUDA Deep Neural Networks) is a library of primitives for building deep neural networks. CuDNN provides implementations of the routines that arise frequently in deep learning applications, such as convolution, pooling, soft-max and neuron activation functions like sigmoid, Rectified linear (ReLU) and Hyperbolic tangent (TanH).cuDNN improves the performance of such routines by using GPU based matrix multiplications.

cuDNN library allows easy development of neural network applications by harnessing state-of-the-art performance and lets the developers focus on the application part, without having to write customized code. cuDNN library primitives can work across the full range of NVIDIA GPUs, from low-power embedded GPUs like Tegra K1 to high-end server GPUs like Tesla K40.

3.2.4 Caffe Framework

Caffe (Convolutional Architecture for Fast Feature Embedding) is free and open source framework for building convolutional neural network systems. Caffe is maintained and developed by Berkeley Vision and Learning Center (BVLC) at University of Berkley.

The architecture of the Caffe framework is described below.

- Data Storage
  Caffe stores and processes data in the form of 4 dimensional data structure called blobs. Caffe provides the various ways of storing the data such as in-memory storage, database based storage using LevelDB and LMDB databases.

- Layers
  Caffe model consists of the different types of the layers. Data layer loads the data into the model. Convolution layers, pooling layers, fully connected layers are used for designing the network model. The model generally ends with the loss layer during the training phase and accuracy layer during the testing phase.

- Training a Network
  Caffe uses fast and standard stochastic descent algorithm for training a network. Various hyper parameters such as learning rate, regularization method, and number of iterations are configured in the form of the solver file of the model.

3.3 Convolutional Neural Networks

Convolutional Neural Network architectures allow encoding spatial properties into the architecture, unlike normal fully connected neural networks [23]. In contrast to the normal neural networks convolutional networks take as input a three dimensional volume of data.

- Convolutional neural networks detect the same feature everywhere irrespective of the position in the image and are space invariant in nature

- Convolutional neural networks requires less number of parameters during the training as compared to the fully-connected networks

3.3.1 Architecture Overview

Generally convolutional neural networks consist of the input layers, convolutional layers, pooling layers and the fully connected layers (Figure: 3.7).

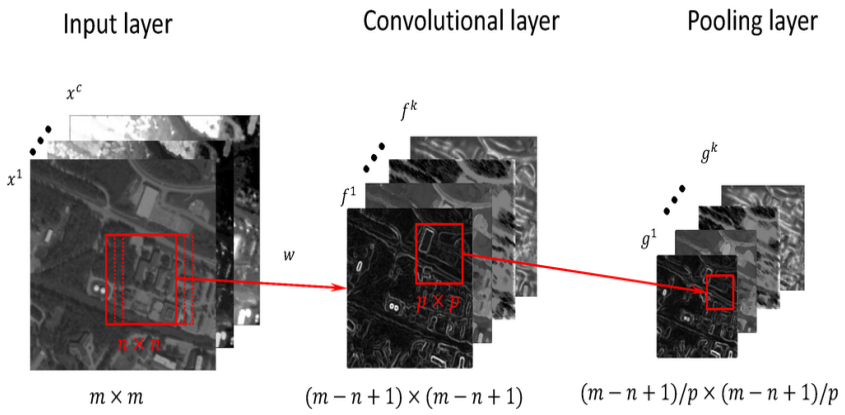


Figure 3.7: Convolution and Pooling layers (Source: http://www.mdpi.com/2072-4292/8/4/329/ag)

- Input layer holds raw pixel values of the image with the dimension c*h*w, where c corresponds to the number of channels, h is height of the image and w is the width of the image.

- Convolutional layer compute the output of the neurons that are connected to local regions in the input layer, each computing a dot product between their weights and a small window or region of pixels they are connected to in the input layer. This results in the generation of output volume whose dimension depends on the kernel size and the number of feature maps.

- Pooling layer performs the down-sampling operations and prepares the condensed feature maps from the output of the convolutional layers.

- Fully Connected layers have connections to all the neurons in the previous layer and are generally used for calculating the class scores for the input layer data.

All different types of layers and their configuration parameters are discussed in details in the following section.

3.3.2 Convolutional Layers

The convolutional layer typically consists of a set of learnable filters or the kernels. Every filter is like a small spatial window along width and height, which extends through the full depth of the input layer. During the convolution operation each filter is slid across the width and height of the input volume and dot products between the entries of the filter and the input at any position are computed. The filter slides over the width and height of the input volume to produce a two-

dimensional activation map that gives the responses of that filter at every spatial position. Each convolution layer has a set of filters and every filter produces a separate two-dimensional activation map (also called as feature map). A set of stacked two-dimensional activation maps produces the output volume.

Convolutional layers make the network to learn filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some color on the first layer, or eventually entire objects or patterns in higher layers of the network.

a. Local Receptive Fields

Local receptive fields (Figure: 3.8) in CNN correspond to the window of input image pixels for a hidden neuron. Each connection from a hidden neuron to the pixels in the local receptive field learns a weight and the hidden neuron learns an overall bias.

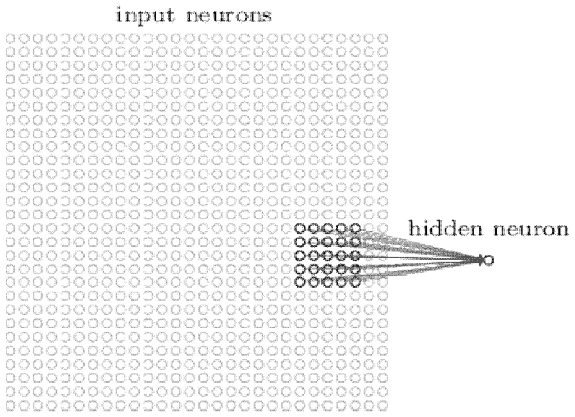


Figure 3.8: Local Receptive field (Source: [24])

While moving the local receptive field over the input volume, the following parameters need to be configured.

- **Window Size:** Window size refers to the spatial extent of the local receptive fields
- **Stride Length:** Stride length refers to the offset for moving a window

b. Shared Weights and Biases

In convolution layer local receptive fields share a same set of weights and biases in their connection to the hidden neuron (Figure: 3.9). Such a map (with shared weights and biases) from input layer to the hidden layer is called as a feature map.

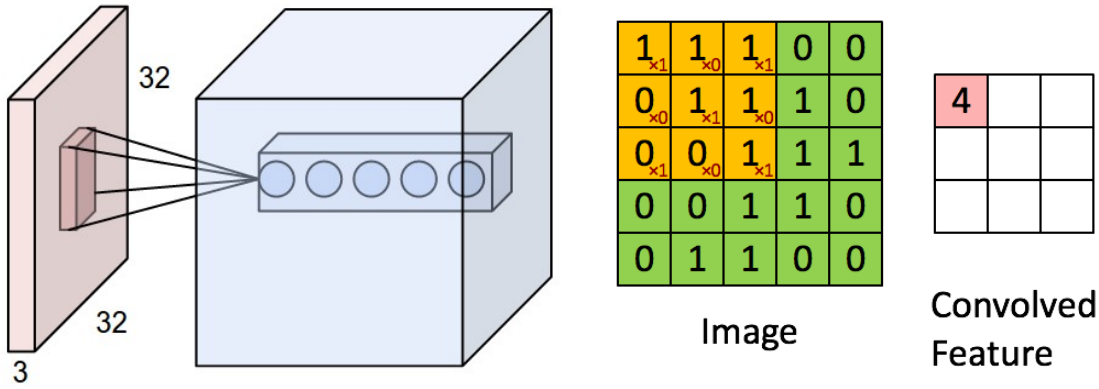


Figure 3.9: Convolution concept (Source: [23])

Different feature maps detect different kind of features from an input volume. Convolutional neural networks require less number of parameters as compared to the fully connected layers due to the sparse connectivity between the layers.

3.3.3 Pooling Layers

Pooling layers (Figure: 3.10) prepare the condensed feature maps from the output feature maps of the convolution layers. Pooling is applied to separately to each feature map and resizes it spatially; hence the number of feature maps in pooling layers is equal to the number of feature maps in the convolution layer.

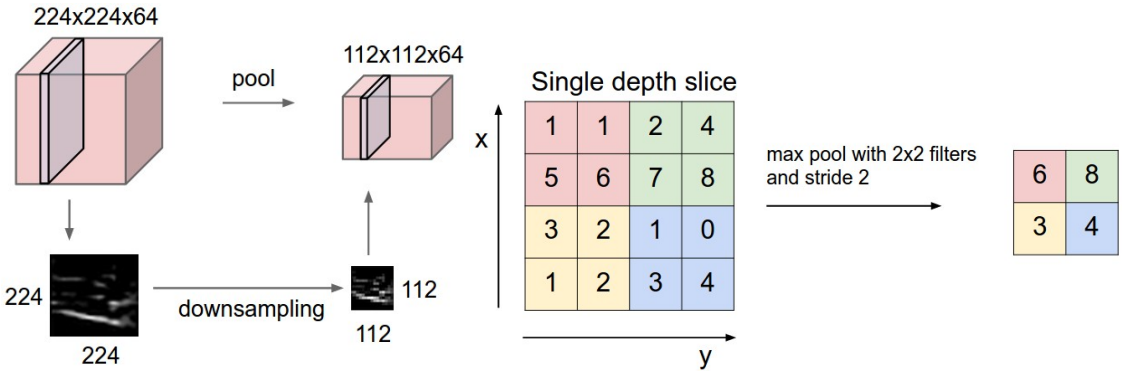


Figure 3.10: Max Pooling Concept (Source: [23])

Pooling layers progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, which in turn helps in reducing the over fitting [23].

For pooling layers the following parameters need to be configured.

- **Window Size**

  Window size refers to the spatial extent of filters applied for down-sampling. This is generally set 2 or 3; sizes with larger receptive fields are too destructive [23].

- **Stride Length**

  Stride length refers to the offset for moving a window for pooling operations

- **Pooling Approach**
  - **Max-pooling**: It is a generally used approach which outputs the maximum activation from the window region. Unlike convolution layers pooling layers deal more with the presence or absence of the feature in the local receptive field [24].
  - **L2 pooling**: In L2 pooling the square root of the sum of the squares of the activations in the window region is taken.[24]

3.3.4 Fully Connected Layers

In fully connected layers the neurons have connections to all activations in the previous layer (Figure: 3.11), as seen in regular neural networks.

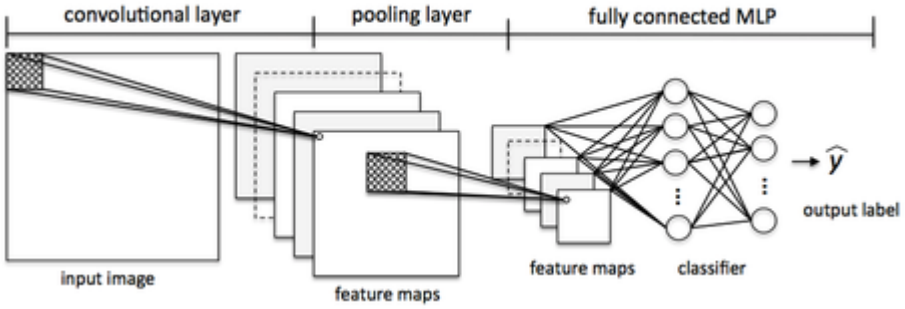


Figure 3.11: Fully connected layers preceded by feature maps (Source: https://learndatascience.net/)

The upper-layers in CNN based classification models generally are fully connected and correspond to a traditional multi layer perceptron. The input to the first fully connected layer is the set of all features maps from the previous layer in the network.

3.4 CNN based Hyperspectral Image Classification on Jetson TK1

The hyperspectral image classification algorithm developed on Jetson TK1 uses Convolutional Neural Network (CNN) based deep learning approach. Convolution layers are used to extract

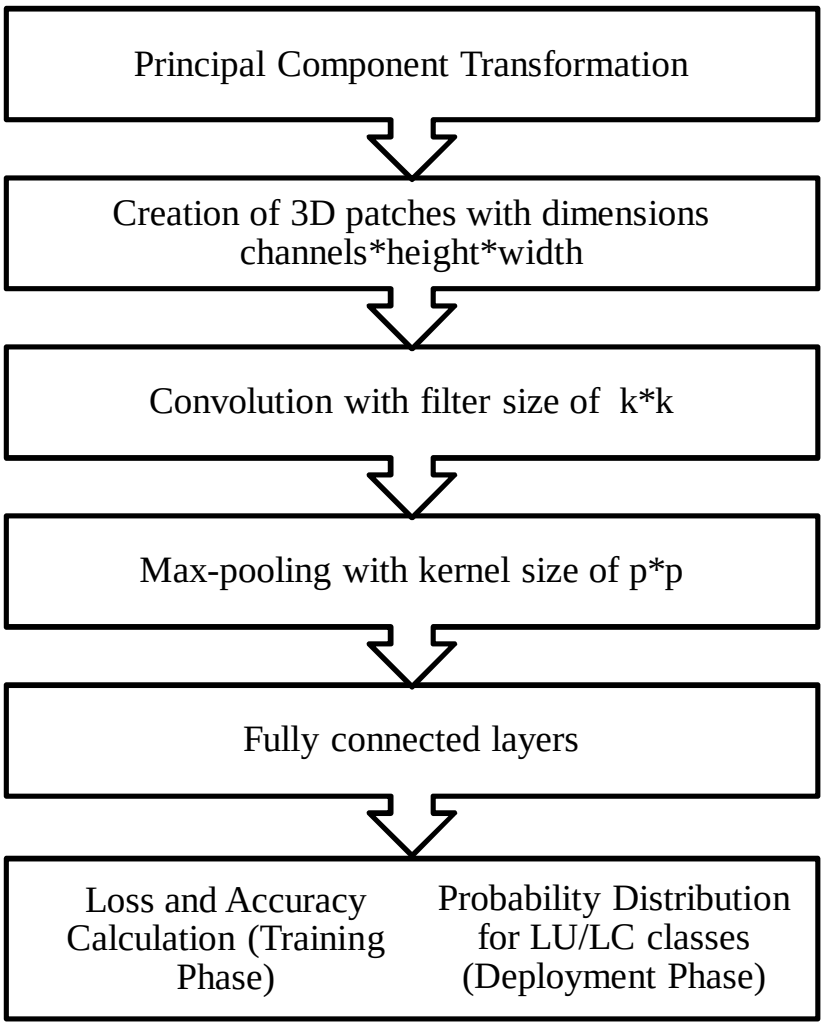


Figure 3.12: Algorithm flow chart for CNN based hyperspectral image classification

spatial-spectral features out of the hyperspectral image followed by fully connected layers for assigning class labels to every individual image pixel (Figure: 3.12). For this study we took into consideration the neighborhood of a pixel and created 3D patches with dimension (channels*height*width), where channels correspond to number of feature vectors, height and width is of the neighborhood around each pixel.

The basic principle behind creating training patches is that the neighborhood pixels in hyperspectral images tend to show close spectral curves due to spatial correlation [7]. Convolutional Neural Networks use local receptive fields, which is a window of input pixels for

a hidden neuron. Local receptive fields use shared weights and biases for the hidden neurons in a layer hence same features are detected everywhere irrespective of the pixel position. Different sets of weights are used to generate various feature maps corresponding to various features. A Convolution layer is followed by a pooling layer. Pooling layers prepare the condensed feature maps from the outputs of the feature maps. Pooling layer deals more with the presence or absence, of the features rather than their spatial location [15]. Sequence of convolution layers and pooling layers act as a spectral-spatial feature extractors for the hyperspectral image. The extracted features are then fed to the fully connected layers of neurons for pixel-based classification.

The principal components were extracted so as to capture the maximum variance in the image data. Principal component analysis with whitening (in whitening covariance matrix of the transformed data is an identity matrix [16]) is performed to ensure that the transformed components of data are uncorrelated. The convolution layer of the classifier model is trained on 3D patches. We have considered the neighborhood of 5 pixels (patch size 5 is chosen as optimal according to spatial resolution of the images, taking into consideration the linear features) with feature vectors transformed along the first 20 principal components (PCs). The images are padded with the edge pixels in order to take care of the boundary pixels during patch generation.

The network consists of convolution layers followed by fully connected layers. Convolution part of the network contains a convolutional layer followed by a pooling layer. The convolutional layer generates predetermined number of feature maps and has kernel size of k*k. The pooling layer uses a max-pooling kernel of size p*p. Features extracted from convolutional part of the network are fed to the multilayer perceptron consisting of fully connected hidden layer of neurons and output layer of neurons. The number of neurons in the output layer corresponds to the number of land-use/land-cover classes in the image. A Sigmoid function is used as an activation function for the fully connected layers of the network. The network model uses soft-max loss function, and loss minimization is done through the stochastic gradient descent (SGD). The model is trained with appropriate initial learning rate (generally equals to 0.001), which is gradually decayed and optimized over the number of iterations.

- **Prototype of the Jetson TK1 based device built for Hyperspectral Image Processing**

We have developed a prototype of a tablet, in order to make Jetson TK1 suitable for carrying out in-field and near-real time analysis of hyperspectral imagery. The images used in the study are archived images, but considering the scenarios where the images are being down linked from UAVs to the field, such a device will be highly useful for carrying out the classification of the image data. This kind of setup would be really useful for rapid decision making. The CNN models can be trained before hand and deployed on the device, and then these models can readily perform classification of the incoming hyperspectral image streams.

The device can be mounted on UAVs or balloons where it can take images from on-board hyperspectral sensors and classify them on-board. In such case, the thematic maps can be sent to the ground stations.

The prototype of the tablet is built using the Jetson TK1, a touch screen display and 3-D printed enclosure for enclosing a Jetson TK1 and a LCD touch screen. The touch screen we have used is a 7 inches resistive touch screen. The touch screen interacts with Jetson TK1 through HDMI driver board for the display and through USB touch controller for the touch controls. The touch screen is powered from the Jetson TK1's GPIO pins with 12 Volts DC power supply.

- **Application and User Interface**

A Jetson TK1 based application is developed for performing the classification of the hyperspectral images. The application uses HTML, CSS and JavaScript for front end and the Python Flask to handle server side tasks.

Through the interactive user interface of the application a user can:

- Select and prepare the dataset for training, validation and testing.
- Train the CNN model using the training dataset and monitor the training loss, validation loss and validation accuracies through the graph showing validation loss and validation accuracies over the iterations.
- Upload a hyperspectral image and classify it using the pre-trained model of the convolutional neural network.

Chapter 4
Results and Discussion

4.1 Datasets

The experiments are carried out on the following four Hyperspectral imagery datasets namely – University of Houston and neighboring urban area, Indian Pines scene, Salinas scene and University of Pavia dataset.

a. University of Houston and neighboring area

The hyperspectral imagery of University of Houston campus and surrounding urban area had been provided during Data Fusion Contest (DFC) 2013 [19]. The image consists of 144 spectral bands in the 380 nm to 1050 nm region and has spatial resolution of 2.5m. The image was acquired by the National Center for Airborne Laser Mapping (NCALM) over the University of Houston campus and the neighboring urban area on June 23, 2012 with average height of the sensor above ground was 5500ft. The image has 15 land use and land cover classes. Training and test datasets were provided separately for the image. Further, the image (Figure: 4.1) contains the shadowed area for which no training pixels were available but testing pixels were provided for the area during the contest. We have used the same training dataset for our purpose.

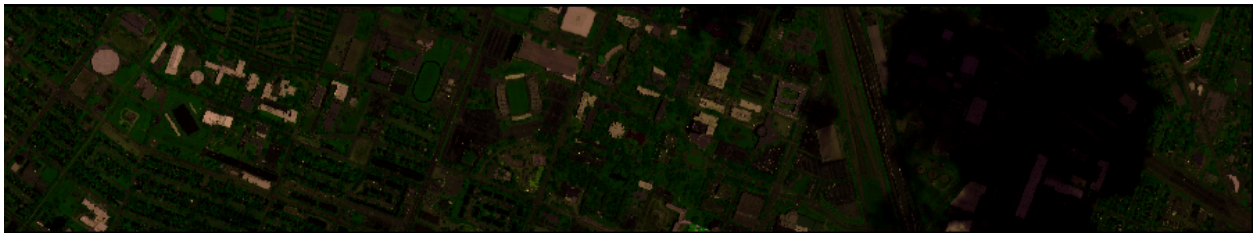


Figure 4.1: University of Houston image

The details about the ground truth data and the land-use land-cover classes are provided in the Table 4.1.

Table 4.1: Ground truth summary of University of Houston dataset

Sr. No.	Class	Training Samples	Testing Samples
1	Healthy grass	198	1053
2	Stressed grass	190	1064
3	Synthetic grass	192	505
4	Tree	188	1056
5	Soil	186	1056
6	Water	182	143
7	Residential	196	1072
8	Commercial	191	1053
9	Road	193	1059
10	Highway	191	1036
11	Railway	181	1054
12	Parking lot 1	192	1041
13	Parking lot 2	184	285
14	Tennis court	181	247
15	Running track	187	473

b. Indian Pines

This hyperspectral image was gathered by AVIRIS sensor over the Indian Pines test site in North-Western Indiana. The image consists of 224 spectral reflectance bands in the wavelength range 400 to 2500 nm. The area covered in the image is of dimensions 145*145 pixels (Figure: 4.2). The Indian Pines scene contains two-thirds agriculture, and one-third forest or other natural perennial vegetation. There are two major dual lane highways, a rail line, as well as some low density housing, other built structures, and smaller roads. The ground truth available is designated into sixteen classes and is not all mutually exclusive [38].

The details about the ground truth data and the land-use land-cover classes are provided in the Table 4.2.

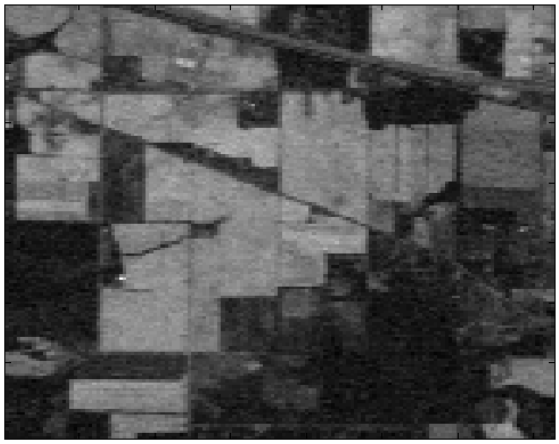


Figure 4.2: Indian Pines image (Source: [38])

Table 4.2: Ground truth summary of Indian Pines dataset

Sr. No.	Class	Samples
1	Alfalfa	46
2	Corn-notill	1428
3	Corn-mintill	830
4	Corn	237
5	Grass-pasture	483
6	Grass-trees	730
7	Grass-pasture-mowed	28
8	Hay-windrowed	478
9	Oats	20
10	Soybean-notill	972
11	Soybean-mintill	2455
12	Soybean-clean	593
13	Wheat	205
14	Woods	1265
15	Buildings-Grass-Trees-Drives	386
16	Stone-Steel-Towers	93

c. Salinas scene

This hyperspectral imagery was collected by the 224-band AVIRIS sensor over Salinas Valley, California, and is having a high spatial resolution of 3.7-meters. The area covered is of dimension 512* 217 pixels (Figure: 4.3). The imagery includes vegetables, bare soils, and vineyard fields. Salinas's ground truth contains 16 classes [38].

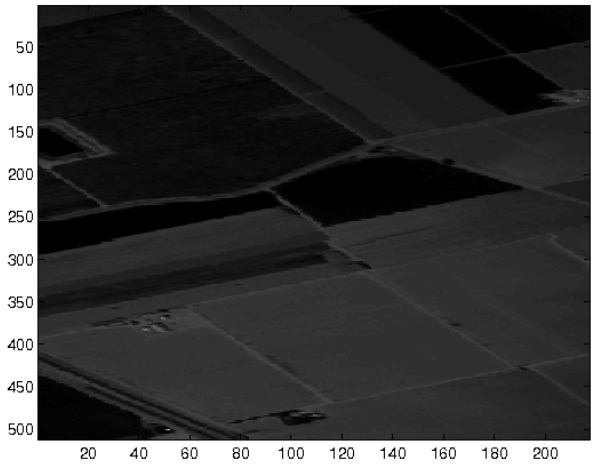


Figure 4.3: Sample band of Salinas scene (Source: [38])

The details about the ground truth data and the land-use land-cover classes are provided in the Table 4.3.

Table 4.3: Ground truth summary of Salinas dataset

Sr. No.	Class	Samples
1	Brocoli_green_weeds_1	2009
2	Brocoli_green_weeds_2	3726
3	Fallow	1976
4	Fallow_rough_plow	1394
5	Fallow_smooth	2678
6	Stubble	3959
7	Celery	3579
8	Grapes_untrained	11271
9	Soil_vinyard_develop	6203
10	Corn_senesced_green_weeds	3278
11	Lettuce_romaine_4wk	1068
12	Lettuce_romaine_5wk	1927
13	Lettuce_romaine_6wk	916
14	Lettuce_romaine_7wk	1070
15	Vinyard_untrained	7268
16	Vinyard_vertical_trellis	1807

d. Pavia University

This imagery is acquired by ROSIS sensor over Pavia, northern Italy. The imagery contains 103 spectral bands for Pavia University. The hyperspectral image of Pavia University has dimensions of 610*610 pixels (Figure: 4.4), but some of the samples in the image contain no information and have been discarded before the analysis. The spatial resolution of the image is 1.3 meters. The image is designated into 9 different land use land cover classes [38].

Figure 4.4: Sample band of University of Pavia dataset (Source: [38])

The details about the ground truth data and the land-use land-cover classes are provided in the Table 4.4.

Table 4.4: Ground truth summary of Pavia University dataset

Sr. No.	Class	Samples
1	Asphalt	6631
2	Meadows	18649
3	Gravel	2099
4	Trees	3064
5	Painted metal sheets	1345
6	Bare Soil	5029
7	Bitumen	1330
8	Self-Blocking Bricks	3682
9	Shadows	947

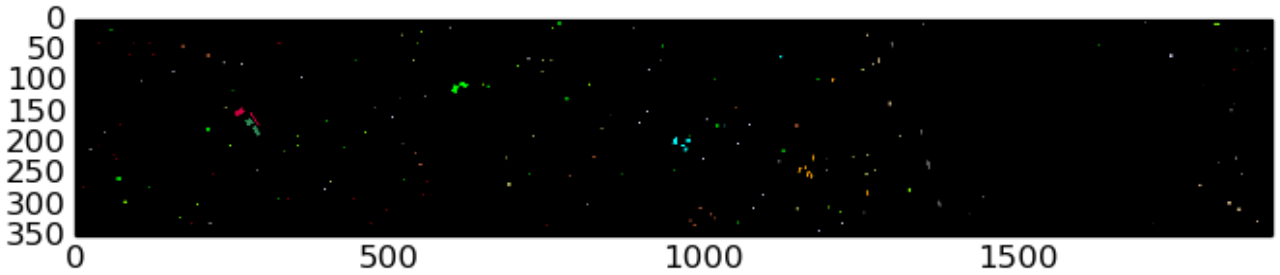
4.2 Classification Results and Accuracy Assessment

The above mentioned datasets of hyperspectral imagery are classified using the convolutional neural network architecture. For the University of Houston dataset training dataset was provided separately hence the same dataset was used for training the CNN classifier. For rest of the hyperspectral imagery the ground truth data pixels are divided into the training and testing datasets. We trained the classifier on 20% of the ground truth samples and remaining 80% of the samples are used for testing.

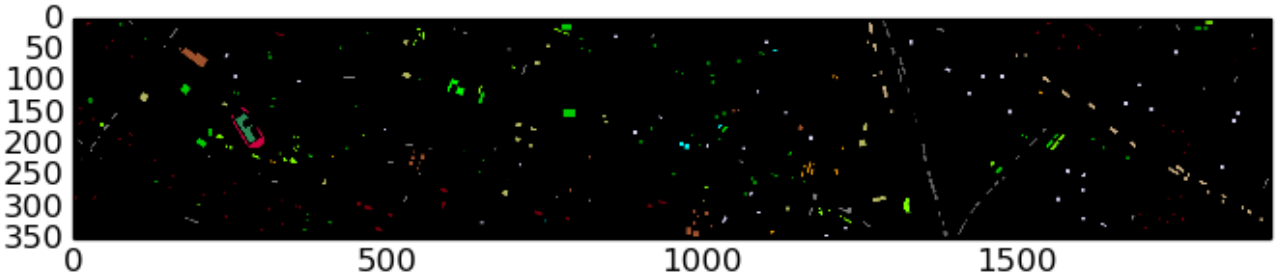
The division of ground truth data into the training and testing datasets is done using the stratified random split method. Stratified random split ensures that the division ratio is maintained across the samples of all the land-use/land-cover classes. The overall classification accuracy for the different datasets is summarized in the Table 4.5. The ground truth along with classification output of the datasets – University of Houston, Indian Pines, Salinas and Pavia University - are shown in Figure: 4.5, Figure: 4.6, Figure: 4.7 and Figure: 4.8 respectively.

Table 4.5: Classification accuracies of CNN on different datasets

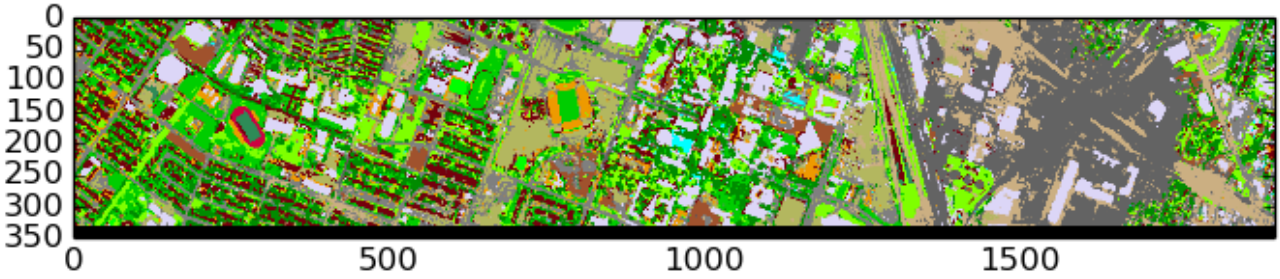
Dataset	Overall Accuracy
University of Houston (excluding shadowed region)	91.22 %
University of Houston (including shadowed region)	82.97 %
Indian Pines	86.62 %
Salinas scene	93.10 %
University of Pavia	94.89 %



(a)



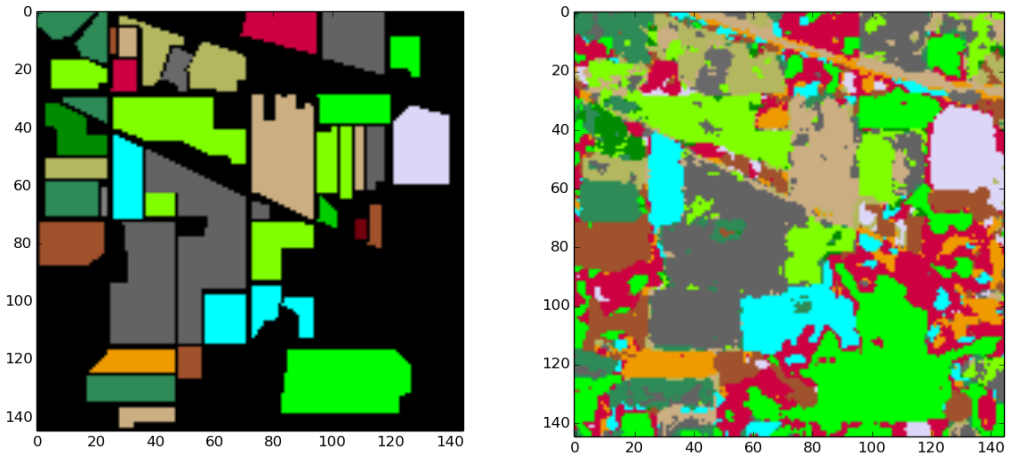
(b)



(c)

Healthy grass	Stressed grass	Synthetic grass	Tree	Soil	Water	Residential	Commercial
Road	Highway	Railway	Parking lot 1	Parking lot 2	Tennis court	Running track	

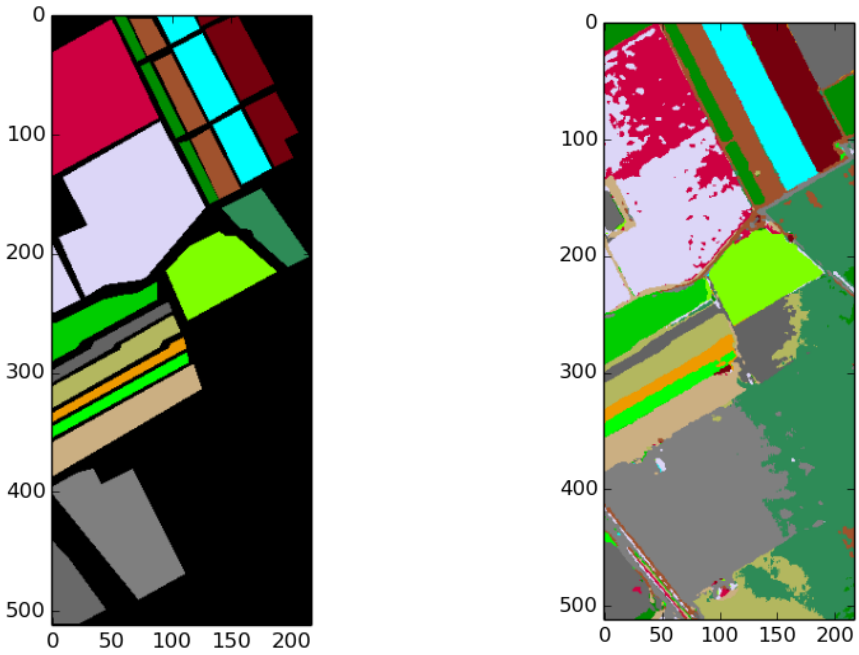
Figure 4.5: (a) Training pixels for University of Houston HSI (b) Testing pixels for University of Houston HSI (c) Classified image of University of Houston



(a) **(b)**

Alfalfa	Corn-notill	Corn-intill	Corn	Grass-pasture	Grass-trees	Grass-pasture-mowed	Hay-windrowed
Oats	Soybean-notill	Soybean-mintill	Soybean-clean	Wheat	Woods	Buildings-Grass-Trees-Drives	Stone-Steel-Towers

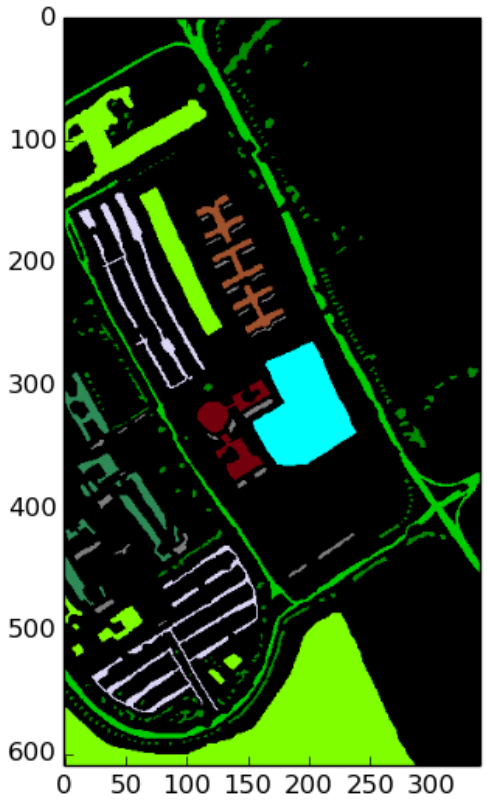
Figure 4.6: (a) Ground truth pixels for Indian Pines HSI (b) Classified image of Indian Pines



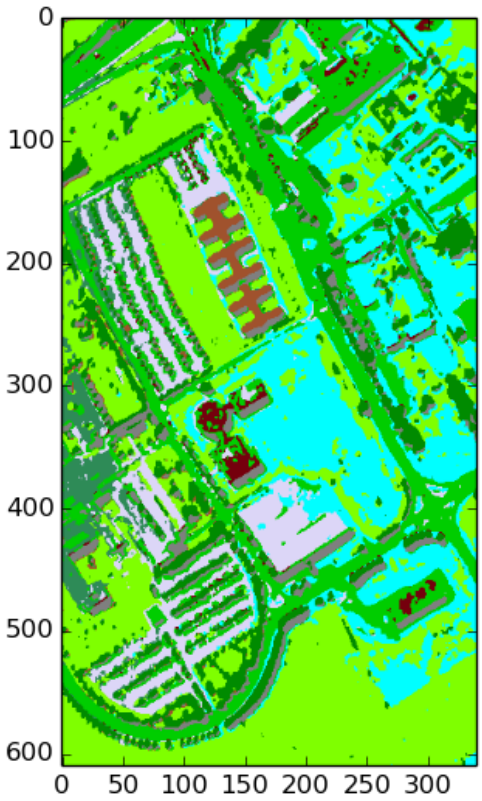
(a) **(b)**

Brocoli_green_weeds_1	Brocoli_green_weeds_2	Fallow	Fallow_rough_plow	Fallow_smooth	Stubble	Celery	Grapes_untrained
Soil_vinyard_ develop	Corn_senesced_green_weeds	Lettuce_romaine_4wk	Lettuce_romaine_5wk	Lettuce_romaine_6wk	Lettuce_romaine_7wk	Vinyard_untrained	Vinyard_vertical_trellis

Figure 4.7: (a) Ground truth pixels for Indian Salinas HSI (b) Classified image of Salinas



(a)



(b)

Asphalt	Meadows	Gravel	Trees	Painted metal sheets	Bare Soil	Bitumen	Self-Blocking Bricks	Shadows

Figure 4.8: (a) Ground truth pixels for Pavia University HSI (b) Classified image of Pavia University

4.3 Analysis of Computation Time

We executed the hyperspectral image classification algorithm in CPU mode and GPU mode. The comparison of the execution times in the prediction phase is described in the Figure: 4.9.

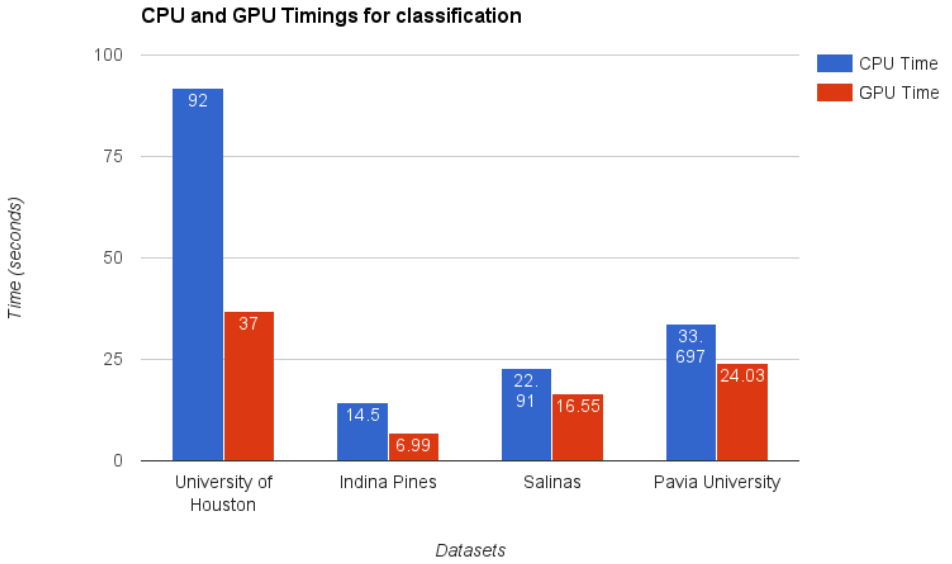


Figure 4.9: Comparative analysis of total time taken for image classification in CPU and GPU mode

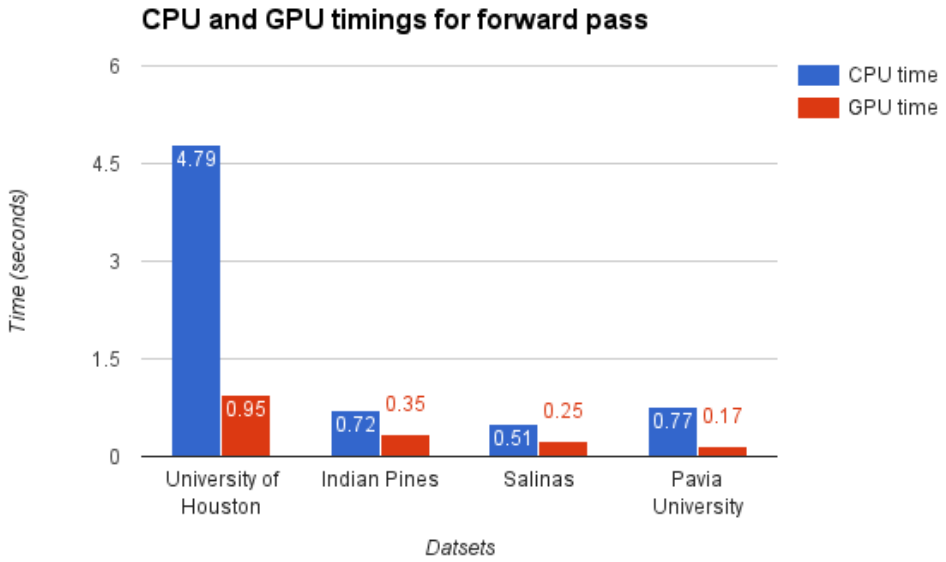


Figure 4.10: Comparative analysis of time taken for a forward pass in GPU and CPU modes

The time taken for the forward pass during the prediction phase of the classification algorithm is also measured and comparative analysis is presented in the Figure: 4.10. Due to the memory limitations of the GPU image pixels were grouped into the batches and prediction is done on the batches of image pixels.

4.4 Analysis of Power Consumption of Jetson TK1

The power consumption of the Jetson TK1 board is measured and analyzed under the various scenarios. The initial power consumption is recorded during boot-up and changes in the power consumption readings of the board are noted while running the hyperspectral image classification algorithm. The power readings of the Jetson TK1 board starts changing as soon as the algorithm starts executing on the board.

For taking the power measurements of the Jetson TK1 board DC current flowing into the board is measured using the standard millimeter. The Jetson TK1 was given a constant power supply of 12 Volts using the switching power adapter with maximum current reading of 5.0 Amperes. The summary of the power draw of the Jetson TK1 board in various situations is summarized in the Table 4.6.

Table 4.6: Power usage summary of Jetson Tk1 board in different scenarios

Operation	Voltage (V)	Min. Current (A)	Min. Power (W)	Max. Current (A)	Max. Power (W)
Idle Mode w/o HDMI and USB	12	0.2	2.4	0.2	2.4
Idle Mode with HDMI and USB	12	0.25	3	0.25	3
Training CPU Mode	12	0.49	5.88	0.54	6.48
Classification CPU Mode	12	0.48	5.76	0.57	6.84
Training GPU Mode (720000 kHz)	12	0.49	5.88	0.55	6.6
Classification GPU Mode (720000 kHz)	12	0.5	6	0.57	6.84
Training GPU Mode (852000 kHz)	12	0.49	5.88	0.56	6.72
Classification GPU Mode (852000 kHz)	12	0.5	6	0.75	9

The GPU clock frequency of Jetson's TK1 can be set in the range of 72000 kHz to 852000 kHz. It is observed that increase in the clock frequency affects the power consumption of the board but also reduces the computational time taken for the forward pass of the CNN based hyperspectral image classification algorithm (Figure: 4.11).

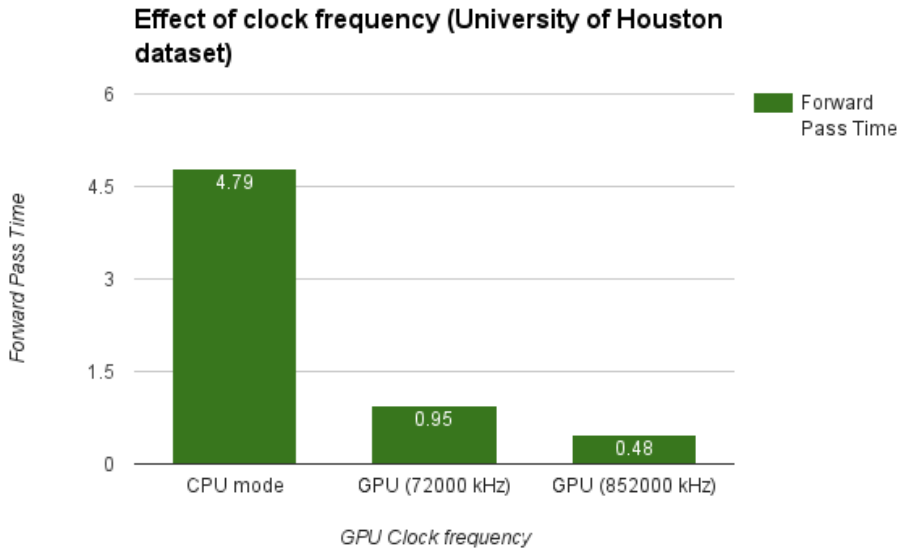


Figure 4.11: Effect of GPU clock frequency on forward pass time

4.5 Prototype of the Jetson TK1 based device for HSI classification

Given below are the snippets of the prototype of a Jetson TK1 based device which hosts the application for hyperspectral image classification. Jetson TK1 (Figure: 4.12) and touch screen are integrated into a 3D printed case (Figure: 4.13).

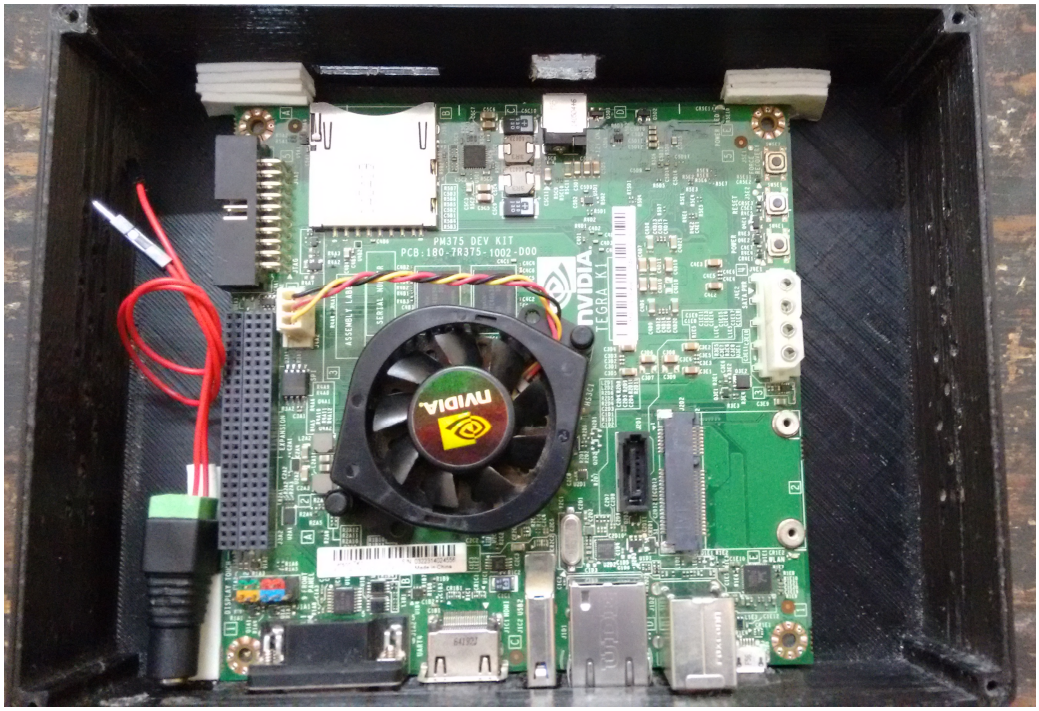


Figure 4.12: Jetson TK1 board inside enclosure

Figure 4.13: Prototype of Jetson TK1 based tablet

The device prototype hosts the application which provides the user interface for preparation of dataset for training, validation and testing. The user can upload ground truth data and generate training, validation and testing databases (Figure: 4.14). The application provides the modules for training-validation and the prediction (Figure: 4.15).

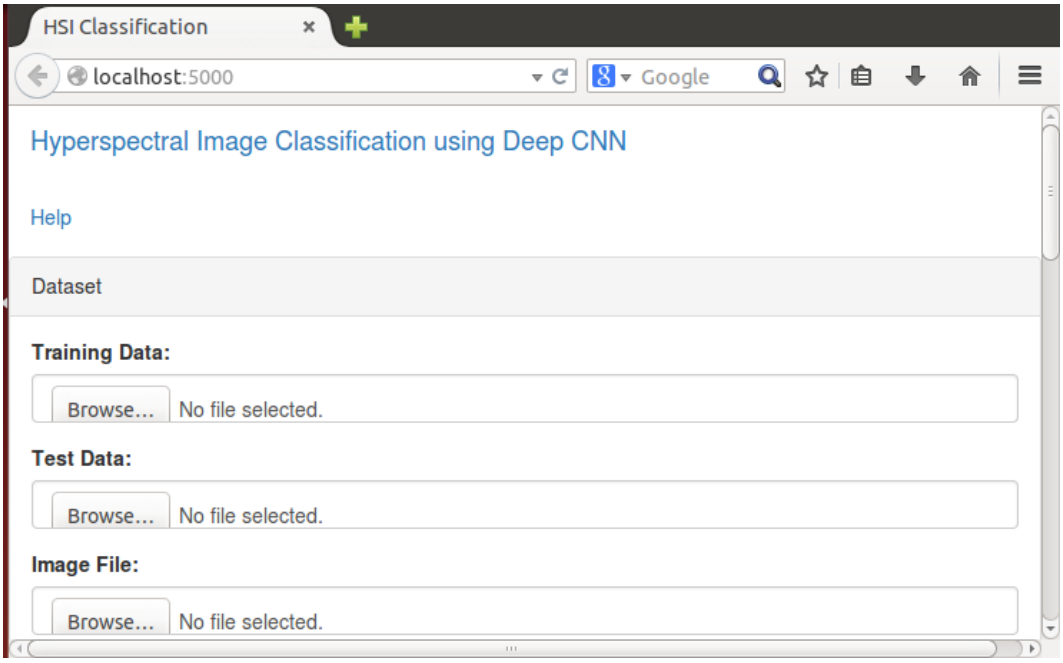


Figure 4.14: Dataset preparation module

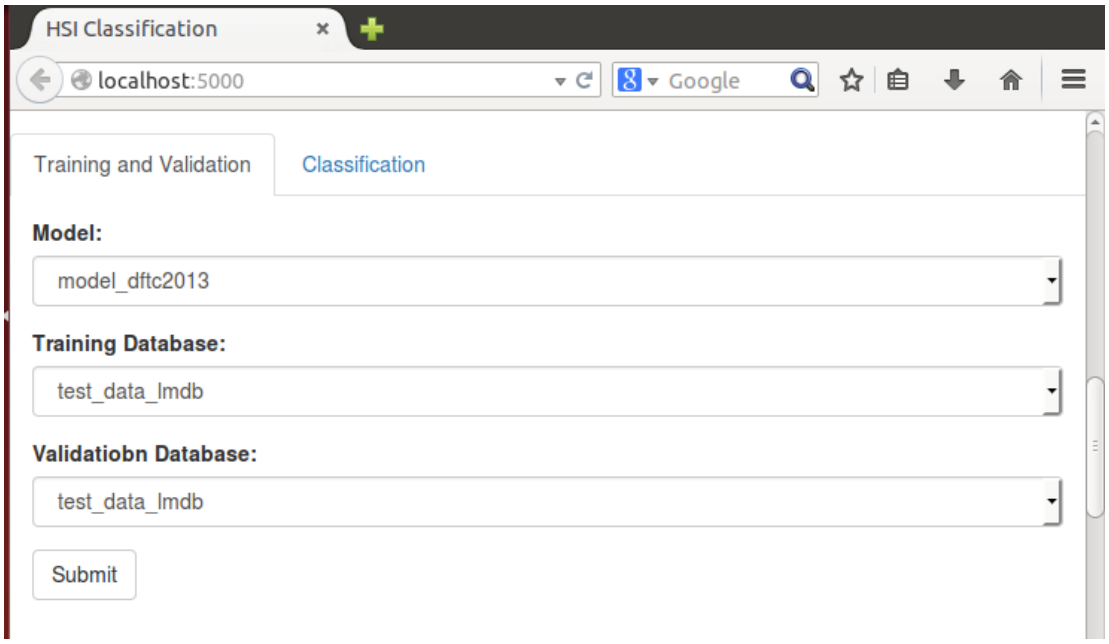


Figure 4.15: Modules for training and classification

The training module of the application allows the user to train the CNN based classification model on the generated training database and validates the model periodically over validation database. The summary of training and validation is displayed in the form of the graph once the

training is completed (Figure: 4.16). The application uses the Caffe deep learning framework for training the hyperspectral image classifier and once the training is completed model is stored in the *.caffemodel* file which can then be used by the classification module to classify hyperspectral images (Figure: 4.17).

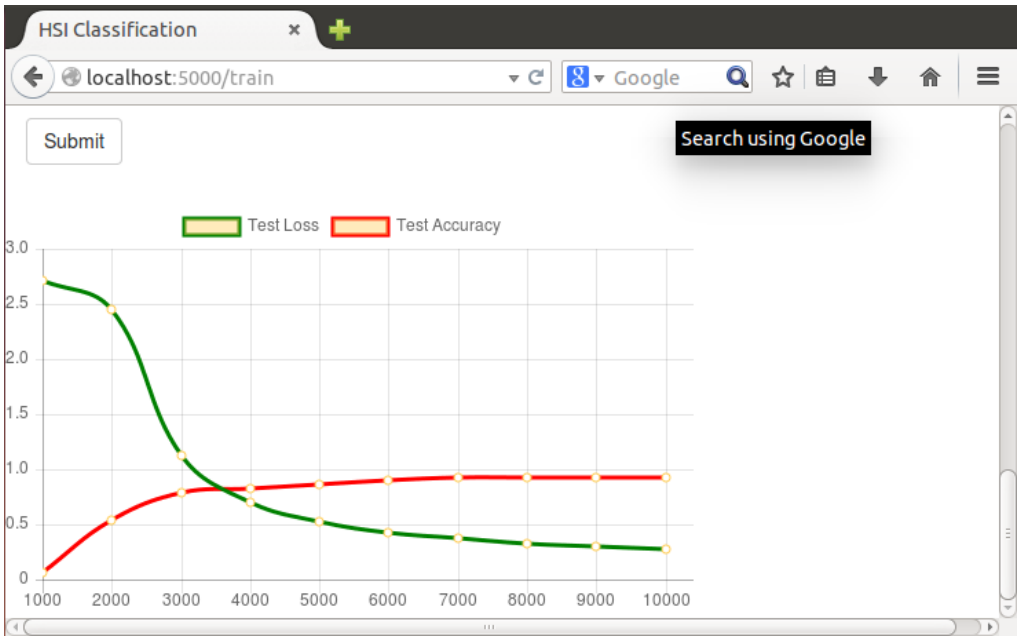


Figure 4.16: Training summary showing validation loss and accuracy

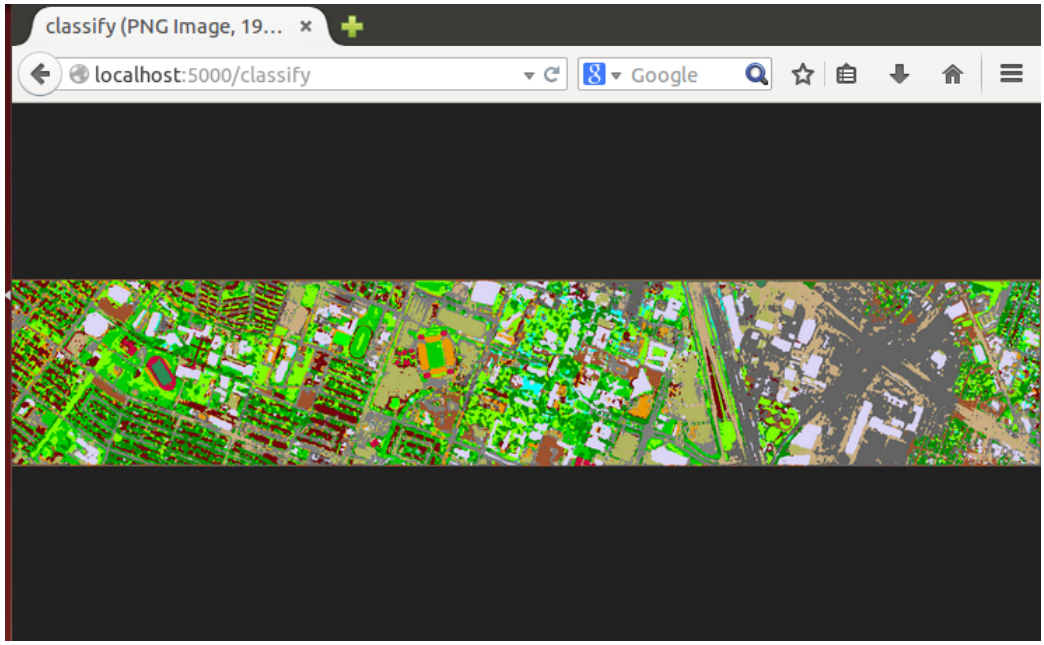


Figure 4.17: Classification output for University of Houston dataset derived on Jetson TK1

Chapter 5

Conclusion and Future Work

Hyperspectral images due to their high spectral resolution are very content rich and can be used for detailed surface studies. Processing of hyperspectral images is a computationally intensive task due to high dimensionality of these images. The advent of low-weight and low-power hardware devices with GPUs such as NVIDIA Jetson TK1, can progressively bridge the gap towards real-time analysis of high dimensional data in many application domains, including remote sensing.

In this study we introduced and developed a convolutional neural network based model on Jetson TK1, targeting embedded system application for onboard and on site classification of hyperspectral images. In recent times, there is a growing interest on using open source frameworks like Caffe with GPU enabled hardware for computer vision tasks, but these frameworks are hardly explored for hyperspectral data processing. We used Caffe deep learning framework, deployed on NVIDIA Jetson's Tegra K1 processor, for hyperspectral image classification. We developed unified deep learning based model, wherein convolutional layers are used for spectral-spatial feature extraction without a need of hand-crafted features, followed by fully connected neural network layers for pixel classification. We were able to successfully train and deploy convolutional deep neural network for rapid hyperspectral image classification on Jetson TK1, which demonstrates its usefulness for near real time classification applications. This kind of specialized low-power HPC system can be used for on-board processing and in turn help in obtaining analysis results quick enough for rapid decision making.

As a future work we would like to incorporate the following improvements into the existing work.

- To attach a hyperspectral sensor to the Jetson TK1 based device and perform the classification of hyperspectral images acquired through the sensor

- To mount the Jetson TK1 based device on UAVs or balloons, along with hyperspectral sensor, and perform the on-board classification

- To attach the wireless adapter to the Jetson TK1 and put in place a mechanism that can acquire and classify the images down-linked from airborne vehicles such as UAVs and balloons

- To develop additional applications based on Jetson TK1 such as application for calculating hyperspectral indices, hyperspectral end-member extraction, etc.

REFERENCES

[1] Romero, Adriana, Carlo Gatta, and Gustavo Camps-Valls."Unsupervised deep feature extraction of hyperspectral images."IEEE GRSS Workshop on Hyperspectral Image and Signal Processing (WHISPERS). IEEE. Vol. 6. 2014.

[2] Makantasis, Konstantinos, et al. "Deep supervised learning for hyperspectral data classification through convolutional neural networks." Geoscience and Remote Sensing Symposium (IGARSS), 2015 IEEE International. IEEE, 2015.

[3] Viktor Slavkovikj, Steven Verstockt, Wesley De Neve, SofieVan Hoecke, and Rik Van de Walle. 2015. Hyperspectral Image Classification with Convolutional Neural Networks. In Proceedings of the 23rd ACM international conference onMultimedia (MM '15). ACM, New York, NY, USA, 1159-1162.DOI=http://dx.doi.org/10.1145/2733373.2806306

[4] Jun Yue, Wenzhi Zhao, Shanjun Mao &Hui Liu (2015)Spectral–spatial classification of hyperspectral images using deep convolutional neural networks, Remote Sensing Letters, 6:6, 468-477, DOI: 10.1080/2150704X.2015.1047045

[5] Plaza, Antonio, et al. "Commodity cluster-based parallel processing of hyperspectral imagery." Journal of Parallel and Distributed Computing 66.3 (2006): 345-358.

[6] León, G., et al. "Exploring the performance–power–energy balance of low-power multicore and many core architectures for anomaly detection in remote sensing." The Journal of Supercomputing 71.5 (2015): 1893-1906.

[7] Wei Hu, Yangyu Huang, Li Wei, Fan Zhang, and Hengchao Li, "Deep Convolutional Neural Networks for Hyperspectral Image Classification," Journal of Sensors, vol. 2015, Article ID258619, 12 pages, 2015. doi:10.1155/2015/258619

[8] Chen, Yushi, et al. "Deep learning-based classification of hyperspectral data." Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of 7.6 (2014): 2094-2107.

[9] Chien, Steve, et al. "Onboard Science Product Generation on the Earth Observing One Mission and Beyond."

[10] Halle, W., et al. "Autonomous onboard classification experiment for the satellite BIRD." International Archives Of Photogrammetry Remote Sensing and Spatial Information Sciences 34.1 (2002): 63-68.

[11] Pingree, Paula J. Advancing NASA's On-Board Processing Capabilities with Reconfigurable FPGA Technologies. INTECH Open Access Publisher, 2010.

[12] Mark Harris. "Jetson TK1: Mobile Embedded Supercomputer Takes CUDA Everywhere". [Online] Available: https://devblogs.nvidia.com/parallelforall/jetson-tk1-mobileembedded-supercomputer-cuda-everywhere/#more-3108

[13] Computer Vision and Deep Learning. [Online]. Available: https://developer.nvidia.com/embedded-computing

[14] Dustin Franklin. Low-Power Sensing and Autonomy with NVIDIA Jetson TK1. [Online] Available: https://devblogs.nvidia.com/parallelforall/low-power-sensingautonomy-nvidia-jetson-tk1/

[15] Michael Nielsen. Neural Networks and Deep Learning. [Online]. Available: http://neuralnetworksanddeeplearning.com/chap6.html#introducing_convolutional_networks

[16] PCA and Whitening. [Online]. Available: http://deeplearning.stanford.edu/wiki/index.php/Exercise:PCA_and_Whitening
[17] Y. Nesterov. A Method of Solving a Convex Programming Problem with Convergence Rate O(1/k√). Soviet Mathematics Doklady, 1983.
[18] Liu, Tianyi, et al. "Implementation of Training Convolutional Neural Networks." arXiv preprint arXiv:1506.01195 (2015).
[19] 2013 IEEE GRSS Data Fusion Contest, [Online]. Available:http://www.grss-ieee.org/community/technical-committees/datafusion/
[20] Jia, Yangqing, et al. "Caffe: Convolutional architecture for fast feature embedding." Proceedings of the ACM International Conference on Multimedia.ACM, 2014.
[21] TORCH: A Scientific Computing Framework for Luajit. [Online]. Available: http://torch.ch
[22] R. N. Sahoo. Hyperspectral Remote Sensing. [Online]. Available: http://www.iasri.res.in/ebook/GIS_TA/M2_4_HYSRS.pdf
[23] CS231n Convolutional Neural Networks for Visual Recognition. [Online]. Available: http://cs231n.github.io/convolutional-networks/#overview
[24] Michael Nielsen. Neural Networks and Deep Learning. [Online]. Available: http://neuralnetworksanddeeplearning.com/chap6.html
[25] Convolutional Neural Networks (LeNet). [Online]. Available: http://deeplearning.net/tutorial/lenet.html#lenet
[26] Deep Learning on GPUs. [Online]. Available: http://on-demand.gputechconf.com/gtc/2015/webinar/deep-learning-course/intro-to-deep-learning.pdf
[27] Paz, Abel, Antonio Plaza, and Javier Plaza. "Comparative analysis of different implementations of a parallel algorithm for automatic target detection and classification of hyperspectral images." SPIE Optical Engineering+ Applications. International Society for Optics and Photonics, 2009.
[28] Alison B Lowndes. Deep Learning. [Online]. Available: http://www.robots.ox.ac.uk/~seminars/seminars/Extra/2015_09_03_GeoffBallew.pdf
[29] Jetson TK1: Mobile Embedded Supercomputer Takes CUDA Everywhere. [Online]. Available: http://devblogs.nvidia.com/parallelforall/jetson-tk1-mobile-embedded-supercomputer-cuda-everywhere/
[30] Rotenberg, et al. Dense DSM Generation Using GPU. [Online]. Available: http://www.ifp.uni-stuttgart.de/publications/phowo13/250Simard.pdf
[31] CUDA Toolkit Documentation. [Online]. Available: http://docs.nvidia.com/cuda/ cuda-c-programming-guide/index.html#axzz3Y45snTiD
[32] Michael Wolfe. The PGI Accelerator Programming Model on NVIDIA GPUs. [Online] Available: https://www.pgroup.com/lit/articles/insider/v1n1a1.htm
[33] Vlad Kindratenko. NCSA GPU programming Tutorial. [Online] Available: http://www.ncsa.illinois.edu/People/kindr/projects/hpca/files/NCSA_GPU_tutorial_d3.pdf
[34] David Luebke. GPU Architecture Implications and Trends NVIDIA Research. [Online] Available: http://s08.idav.ucdavis.edu/luebke-nvidia-gpu-architecture.pdf
[35] CUDA Toolkit Documentation. [Online]. Available: http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#axzz3Y45snTiD
[36] Plaza, Antonio, Javier Plaza, and Sergio Sánchez. "Parallel implementation of end member extraction algorithms using NVIDIA graphical processing units." *2009 IEEE International Geoscience and Remote Sensing Symposium*. Vol. 5. IEEE, 2009.

[37] Wu, Xianyun, et al. "Real-time implementation of the pixel purity index algorithm for end member identification on GPUs." *IEEE Geoscience and Remote Sensing Letters* 11.5 (2014): 955-959.
[38] Hyperspectral Remote Sensing Scenes. [Online]. Available: http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes
[39] S. Chien, R. Sherwood, D. Tran, B. Cichy, G. Rabideau, R. Castano, A. Davies, D. Mandl, S. Frye, B. Trout, S. Shulman, D. Boyer, "Using Autonomy Flight Software to Improve Science Return on Earth Observing One, Journal Aerospace Computing, Information, & Comm, April 2005, AIAA.
[40] R. Green, G. P. Asner, S. Ungar, R. Knox, "Results of the Decadal Survey Hypsiri Imaging Spectrometer Concept Study: a high signal to noise ratio and high uniformity global mission to measure plant physiology and functional type," Proc IEEE Intl Geoscience & Remote Sensing Symp, Boston, MA, July 2008
[41] Andrew Kuebler & Joshua Lake. NVIDIA Kepler GPU Microarchitecture. [Online]. Available: http://meseec.ce.rit.edu/551-projects/spring2014/4-2.pdf