SSC17-WK-56

Satellite Identification Imaging for Small Satellites Using NVIDIA

Nick Buonaiuto, Craig Kief, Mark Louie, Jim Aarestad, Brain Zufelt
COSMIAC at UNM
Albuquerque, NM; 916-539-1526
nick.buoniauto@COSMIAC.org

Rohit Mital, Dennis Mateik
Stinger Ghaffarian Technologies Inc.
Colorado Springs, CO; 719-201-6996
rmital@sgt-inc.com

Robert Sivilli, Apoorva Bhopale
Air Force Research Laboratory, Space Vehicles Directorate
Albuquerque, NM; 505-846-1813
robert.sivilli.1@us.af.mil

ABSTRACT

The Nvidia Tegra X1 (TX1) is a credit-card size system-on-a-chip (SoC) that contains an entire suite of input, output, and processing hardware. It is designed to take advantage of Nvidia's graphics processing unit (GPU) architecture and CUDA (formerly Compute Unified Device Architecture) parallel computing platform in order to provide a deep learning capability within a small form factor. The novelty of such a small size makes the TX1 capable of being deployed onboard a satellite, or as the primary instrument of a CubeSat. Accompanying software exists to optimize the TX1 for image processing tasks such as image recognition, object detection and location, and image segmentation. Such on-board processing power would make an equipped satellite able to execute complex decisions based on the images it receives during flight. This paper describes the effort to achieve these image processing tasks on the ground based on original datasets, with the motivation that models could be trained to be deployed onboard spacecraft containing cameras and GPU hardware. Though the distances of space make high-resolution images difficult to obtain from orbital assets, compact devices such as the Nvidia TX1 (and the newer TX2) demonstrate the potential for a spacecraft to achieve increased situational awareness based on streams of collected images.

I. BACKGROUND

As more and more satellites are launched (including smaller CubeSats), the requirements to more accurately and quickly identify objects from orbit will become increasingly important. To be most responsive, there is a need to move the analysis of objects from ground-based systems to onboard assets that can quickly identify not only what is flying by (e.g. recognizing junk versus a CubeSat), but also the types of satellites being observed (e.g. communications, imaging, friendly, hostile, etc.). This level of space situational awareness (SSA) is becoming more critical each day. At the same time that the SSA demand for actionable information is growing, the funding for large space programs is shrinking, which leads to a premium being placed on efficiency. What has to occur for the warfighter to get the best information most quickly, to provide time to make the best decisions, is for the advancement of the capabilities that enable data-collecting instruments to also have the ability to process and transform data into intelligence. Opportunities are made possible by hardware such as Nvidia graphics processing units to provide machine learning type analysis in real time, so that processed information (and not raw data) are downloaded to ground stations for action.

The purpose of this project is to experiment with GPU hardware for image processing and inference within a small form factor, such as that of a nanosatellite or CubeSat. Nvidia's graphics processing hardware is capable of massively parallel computation, which is realized by several accompanying software libraries and deep learning frameworks, discussed briefly in Section II.[1]

The Nvidia TX1 has height and width of 3.5 x 2 inches, with a depth of 1.5 inches including the heatsink and fan.

The system includes the following specifications and input/output (I/O) ports, seen in Table 1.[2]

Table 1: NVIDIA TX1 Module Specifications.[2]

Feature	Specifications
GPU	NVIDIA Maxwell, 256 CUDA cores
CPU	Quad ARM A57/2 MB L2
Video	4K x 2K 30 Hz Encode (HEVC) 4K x 2K 60 Hz Decode (10-Bit Support)
Memory	4Gb 64 bit LPDDR4 25.6 GB/s
Display	2x DSI, 1x eDP 1.4 / DP 1.2 / HDMI
CSI	Up to 6 Cameras (2 Lane) CSI2 D-PHY 1.1 (1.5 Gbps/Lane)
PCIE	Gen 2 | 1x4 + 1x1
Storage	16 GB eMMC, SDIO, SATA
Other	UART, SPI, I2C, I2S, GPIOs
USB	USB 3.0 + USB 2.0
Connect.	1 Gigabit Ethernet, 802.11ac WLAN, Bluetooth

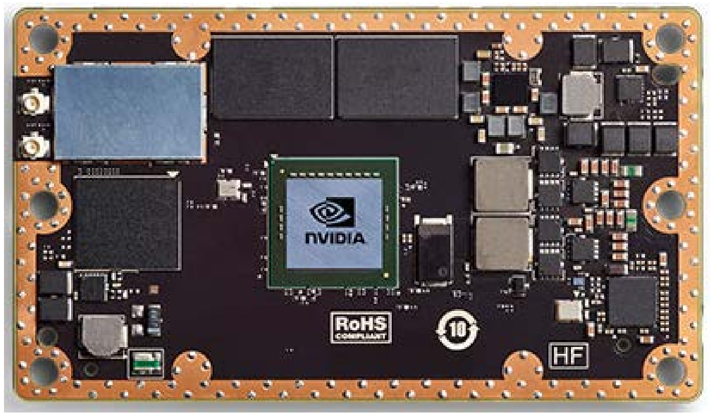


Figure 1: Nvidia TX1 module with heatsink removed, actual size.[2]

The small size of the TX1 can be seen in Figure 1. When mounted onto a 7 x 7 inch printed circuit board containing the actual I/O device ports, the system is known as the Nvidia Jetson TX1 development board, as seen in Figure 2. For software, Nvidia's Jetson-Inference suite contains all of the instructions, code, and other packages necessary for both training models on powerful local computers, or on GPU-optimized Amazon Web Services (AWS) instances, and then deploying trained models onto a Jetson TX1 or TX2.[3] Training utilizes Nvidia's Deep Learning GPU Training System (DIGITS), which is a software package meant to be deployed on ground systems equipped with Nvidia GPUs built with at least Maxwell or Pascal microarchitectures.[4] However, once the trained models are uploaded to the Jetson unit, all further processing of newly collected data occurs onboard. This means computationally expensive model training can be accomplished on the ground, and then fully trained and optimized model files are uploaded to the deployed spacecraft to enable onboard processing of mission data.

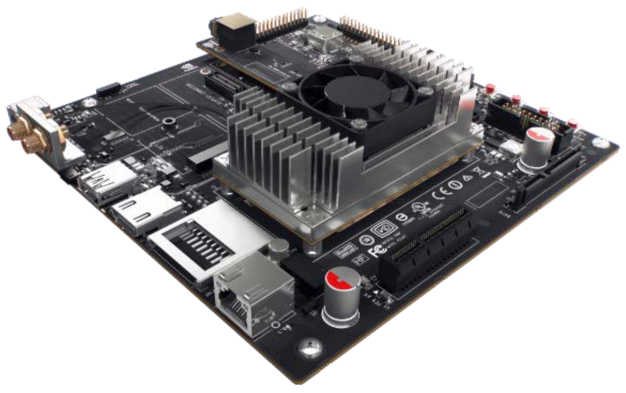


Figure 2: Jetson TX1 development board; the silver heatsink with black fan covers the TX1 SoC module.[5]

II. INTRODUCTION AND EXPERIMENTAL CONFIGURATION

In 2016, work began at COSMIAC Research Center at the University of New Mexico in Albuquerque to explore using GPU processing within small form factors. With COSMIAC's history and experience with small satellites, aerial drones, and computer graphics, the field of image processing was a natural area of interest as increasingly more images are being generated and saved by innumerable devices. COSMIAC routinely works with the Air Force Research Laboratory (AFRL) at Kirtland Air Force Base in Albuquerque. COSMIAC also collaborates with SGT, Inc. in Greenbelt, Maryland on a variety of projects involving enterprise ground station solutions and space situational awareness. SGT has a long history working with the National Aeronautics and Space Administration (NASA), among other federal partners.

The purpose of working with Nvidia image processing is to make use of the enhanced capability of graphics processing units to enable the massively parallel processing of real-time big datasets, such as the images and video collected by aerial drones, autonomous vehicles, and orbital spacecraft. Moreover, the Nvidia Corporation produces and supports small form factor hardware that could fit into such platforms, as well as the software to execute image processing and object inference workloads on such embedded hardware. For this paper, image processing and inference will refer to any of the following three tasks, each of which utilizes a particular package of the Jetson-Inference software suite:

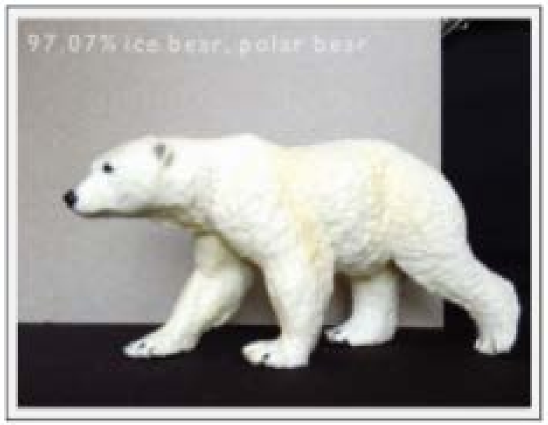


Figure 3: Example of image recognition with imageNet; this example model gives 97.07% probability that this image contains a polar bear or ice bear .[3]

The first image processing task is image recognition with the imageNet software package, seen in Figure 3. A satellite camera sensor system could be trained to recognize and respond to a variety of objects or situations it observes in space. Models can be trained to recognize images to a high degree of accuracy. The Jetson-Inference suite contains the imageNet package for image recognition. This includes the ImageNet database of 1000 labelled images, which can be used as a database for image recognition training.[6] The ImageNet database of 1000 images should not be confused with the imageNet Jetson package; they are separate and distinct entities despite working together and having very little difference in naming style. The Jetson-Inference software also incorporates the AlexNet and GoogLeNet convolutional neural networks for training image classifiers.[7, 8] Convolutional neural networks (CNNs) form the basis of deep learning, and AlexNet and GoogLeNet are examples of CNNs which have been created and optimized for classifying objects in images. Each competed in the ImageNet Large Scale Visual Recognition Challenge in 2012 and 2014, respectively, which is a competition to create the best-optimized classification models based specifically on the ImageNet database. These two CNN image classification models were themselves prefigured by the LeNet-5 convolutional neural network created in 1998.

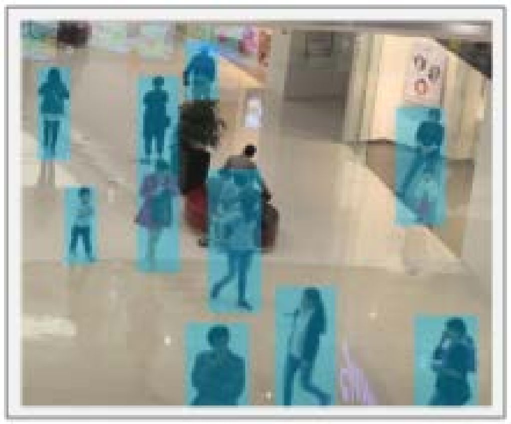


Figure 4: Example of object detection and location with detectNet; this example model locates pedestrians in a public area.[3]

The second image processing task is object detection and location with detectNet, seen in Figure 4. Similar to image recognition, a sensor system could be trained to detect objects and locate their in-frame coordinates. The Jetson-Inference suite also contains the detectNet package for detecting specific objects or in-camera events, and extracting their geometric bounding boxes within the image. As with imageNet object recognition, detectNet models are trained using collections of labelled images, such as the ImageNet database of 1000 object images. However, instead of merely identifying objects, detectNet goes further and also locates them within the picture by drawing a bounding box. Additionally, these bounding boxes will track and stay hovering over objects as they move in a video stream or live camera feed.

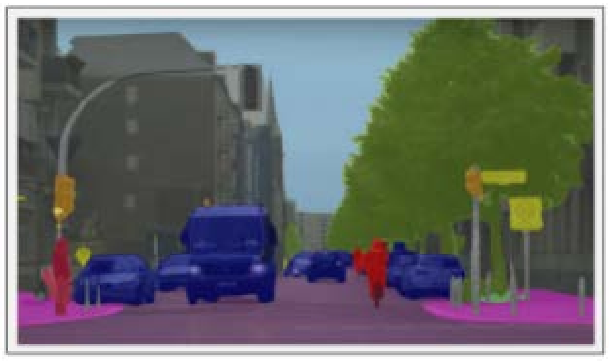


Figure 5: Example of image segmentation with segNet; this example model identifies and separates a multitude of different object types in an urban setting.[3]

Finally, the third image processing task is image segmentation, seen in Figure 5. Each object within an image frame can be identified and separated by type. The Jetson-Inference suite contains the segNet package for image segmentation. In a simple implementation, image segmentation can be done merely to separate the ground from the sky, as with video taken from an aerial drone. However, more objects can be incorporated in addition to just ground and sky (including Earth, space, and satellites), leading to more complex segmentations of multiple types of objects as seen in Figure 5. Segmentation incorporates both object recognition and detection.

To perform image processing and object inference according to the Nvidia Jetson-Inference guide, the required hardware includes two systems with Nvidia GPUs for training and deployment: [3]

First, the training GPU (the "host") must have Nvidia Maxwell or Pascal architecture at minimum—these are the two most recent types of Nvidia GPU architectures available to consumers, with the upcoming Volta microarchitecture scheduled to be released in 2018. The number of CUDA cores, and therefore parallel processing capability, is governed by which version of Nvidia microarchitecture a GPU was built with. Or, instead of costly in-house graphics processing units, a less-costly GPU-optimized AWS cloud compute instance could be utilized instead. Either way, the objective is to train using a powerful GPU with a massive number of CUDA cores and for parallelization. The training instance is optimized for 64-bit Ubuntu 16.04 or 14.04. This project utilized a host computer with an Intel i7-7700 processor (3.6 gigahertz with four hyperthreaded cores), 32 gigabytes (Gb) double data rate fourth-generation 2400 megahertz (DDR4-2400) random access memory (RAM), an M.2 form-factor peripheral component interconnect express (PCI-E) non-volatile memory express (NVME) solid state drive (SSD), and an Nvidia GTX 1050 Ti GPU with 4Gb of video random access memory (VRAM) and Pascal architecture.

Note that an Ubuntu virtual machine will not work as a training system without extra configuration—this is due to the host GPU not being available automatically to the virtual machine. Therefore, the DIGITS software installed in a VM will not be able to detect that an Nvidia GPU is actually present, without further installation and configuration beyond the scope of the Nvidia Jetson-Inference guide.

Second, the deployment GPU (the "Jetson" or "TX1") must be an Nvidia TX1 Developer kit with JetPack 2.3 software or newer, or Nvidia TX2 Developer kit with JetPack 3.0 or newer, and Ubuntu 16.04. This project utilizes the Jetson TX1 development board. The Jetson TX1 and TX2 hardware were designed specifically to complement the JetPack software, and therefore the hardware and software must be used together. The small form factor of the Jetson TX1 makes it interesting for satellite deployments, and the processing capability of Nvidia GPU hardware makes it interesting for analyzing big image data.

For software requirements, the Nvidia JetPack is the software development kit (SDK) for the Jetson TX1/TX2 development boards. Installation of the JetPack software onto the training system (the host) can take place during the same process as flashing the Ubuntu operating system and JetPack software onto the Jetson TX1. This flashing process will also install the CUDA toolkit, the CUDA Deep Neural Network (cuDNN) package, and the TensorRT software package onto the Jetson. Onto the host will also be installed cuDNN, the Nvidia Caffe (NVcaffe) software package, and the DIGITS software package. Deep learning networks typically consist of two phases: training and inference. Training occurs on the powerful host computer, and inference would occur in the field (in orbit) on the Jetson TX1.[9]

Caffe is a deep learning framework originally developed at University of California, Berkeley for image classification.[10] Nvidia's implementation of Caffe is at the core of the TX1's deep learning processing ability. Other examples of deep learning frameworks include Theano, Torch, and TensorFlow.[11] Caffe is installed on the host system. Deep learning frameworks such as Caffe are what conduct the conversion of data (e.g. images) into tensor objects, as well as the mathematical operations for optimizing the neural network. For simplicity, tensors can be considered to be *N*-dimensional array objects that are inputs into a neural network. The "learning" and optimization occurs as the neural net adjusts over subsequent passes of data through, in order to correct error between its output and the expected output.[12] For image classification, neural nets are used in order to train and deploy models to be able to recognize and detect various specified objects within an image.

The CUDA Deep Neural Network (cuDNN) is a suite of Nvidia software libraries for GPU-acceleration of deep learning frameworks such as Caffe. As images are fed into the Caffe neural net framework, the cuDNN software accelerates this processing by utilizing the massively parallel processing capability of the Nvidia GPU hardware. GPU acceleration is accomplished in part by parallelization across the multiple CUDA cores within the Nvidia GPU microarchitecture.[13] CUDA formerly stood for "Compute Unified Device Architecture", but this acronym is no longer used. CuDNN is installed on both the host system and on the TX1.

As previously mentioned, Nvidia's Deep Learning GPU Training System (DIGITS) is the software package for training neural networks for the tasks of image classification, detection, and segmentation, as seen in Figure 6. DIGITS is Nvidia software that provides the main interface for training models on the host computer, both via command line and web browser, and is installed on the host system.[4] However, while the Jetson TX1 has network access, it will be able to access the DIGITS server of the host via web browser.

The TensorRT software package is Nvidia's deep learning inference optimizer and runtime engine for deploying neural network applications. TensorRT also provides great advantages in terms of power reduction so as to make its use in deployments very advantageous for small satellites with limited power budgets. As with cuDNN, TensorRT is Nvidia software which is designed to optimize the deployment of neural nets onto GPU hardware. TensorRT is installed onto the Jetson, and is what deploys the trained neural net model from DIGITS on the host to the Jetson TX1, as seen in Figure 7.[14]

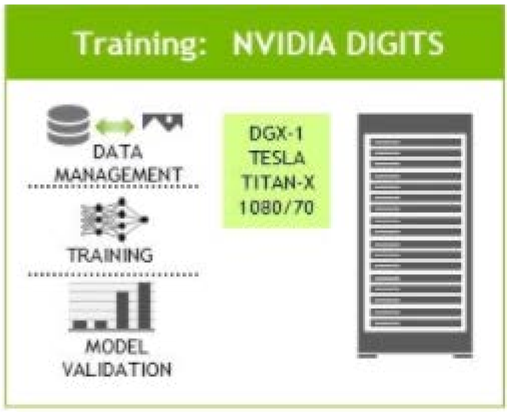


Figure 6: Training with DIGITS on host system—deep neural net models (such as AlexNet and GoogLeNet) are trained using vast image repository databases (such as ImageNet).[3]

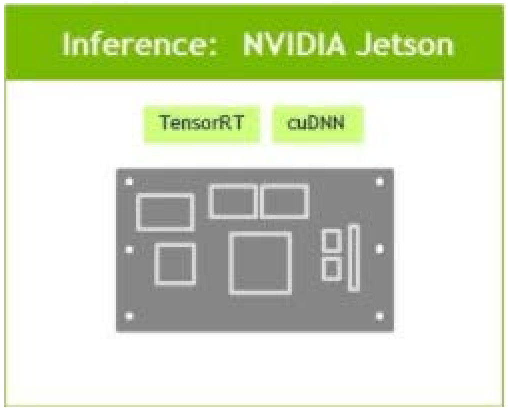


Figure 7: Inference with TensorRT and cuDNN on deployed Jetson system—camera inputs record image data for subsequent processing, and also permit real-time image processing.[3]

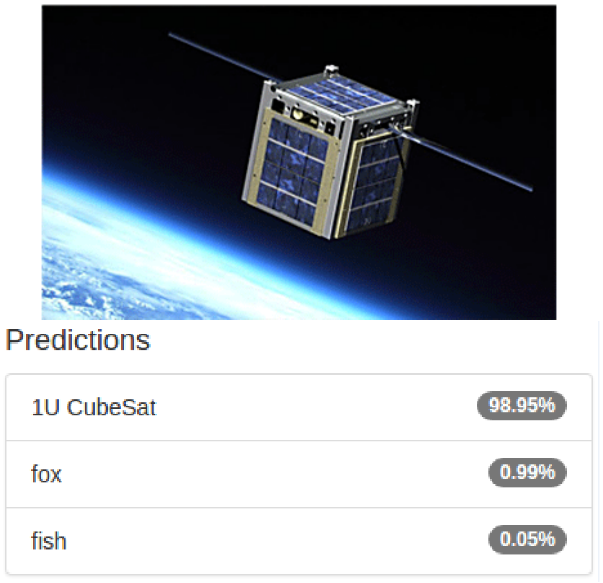
III. EXPERIMENTAL RESULTS

The experiments consisted of the three image processing tasks: recognition using imageNet, detection using detectNet, and segmentation using segNet. The goal at all times was to achieve proof of concept for each task on the TX1, which has a compact form factor that makes it ideal for small satellites.

The first task consisted of image recognition with imageNet. The imageNet package performs image recognition by accepting an input image and outputting the percentage that the content of the image belongs to a particular class. The AlexNet and GoogLeNet neural networks are utilized, which are trained on the ImageNet database of 1000 objects. The 1000 objects are arranged as a directory of images organized into subfolders, with the subfolder names being the image class labels. The interface for imageNet training is DIGITS, via either the command line or web browser.

To demonstrate the ability to classify new objects, it is possible to add items to the existing ImageNet database of 1000 objects. Subfolders containing hundreds of images of 1U- and 3U-sized CubeSats were created at COSMIAC in order to augment the existing model. However, training hundreds of images through a convolutional neural network is a very computationally expensive process, and is likely to be non-trivial using anything but the latest GPU hardware with at least 8Gb of VRAM with Pascal (or newer e.g. Volta in 2018) Nvidia microarchitecture. The host computer for this experiment is equipped with an Nvidia GTX 1050 Ti GPU which, while having only 4Gb of VRAM, has the latest Pascal architecture and cost only $150 at the time of the experiments. For example, quickly training a rough model to classify 1U-sized CubeSats and just *two* other objects (foxes and fish, selected randomly from the ImageNet 1000 objects) took approximately 30 minutes. Re-training for the full 1000 objects plus additional items would take considerably longer, in proportion to the total number of items in the training image database. Naturally, the solution to these computationally expensive challenges is financially expensive GPU hardware—for example, COSMIAC recently upgraded to the Nvidia GTX 1080 Ti GPU with 11Gb of video memory, which cost just over $700 for a baseline version.

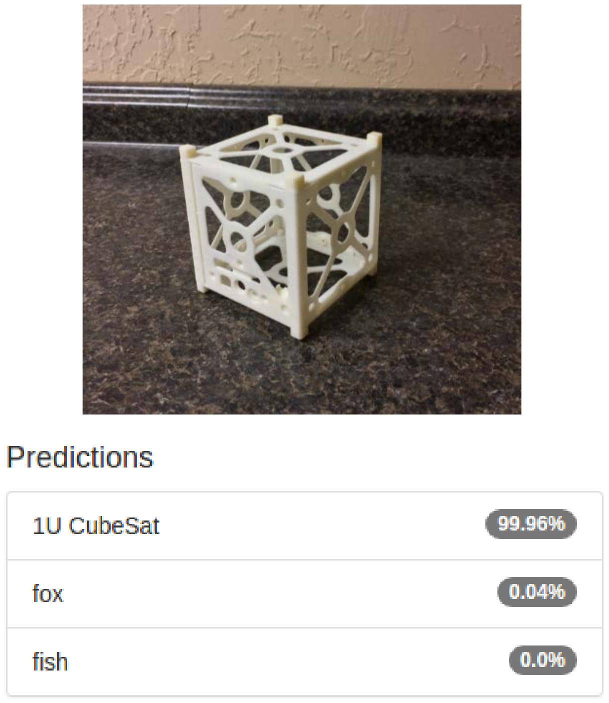
Examples of the results of our CubeSat-Fox-Fish image classification model are seen as follows in Figures 8, 9, 10, and 11:



Predictions

1U CubeSat	98.95%
fox	0.99%
fish	0.05%

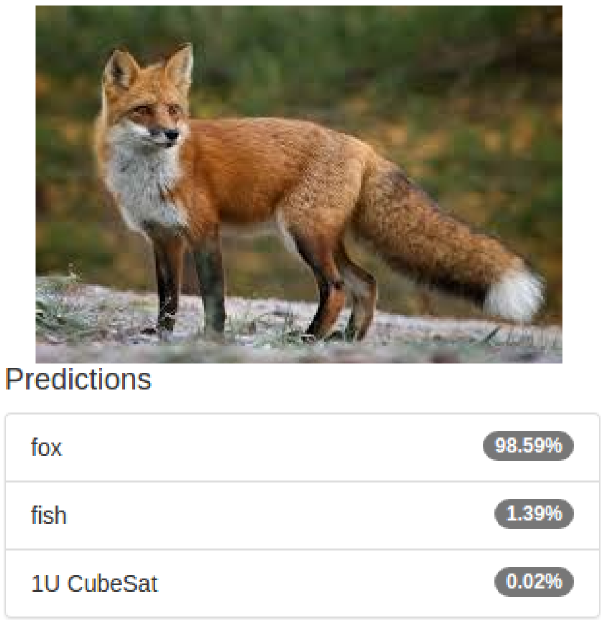
Figure 8: imageNet classification results for 1U-sized CubeSat image—the model calculates there is a 98.95% chance this image contains a 1U CubeSat; photo credit: Montana State University.



Predictions

1U CubeSat	99.96%
fox	0.04%
fish	0.0%

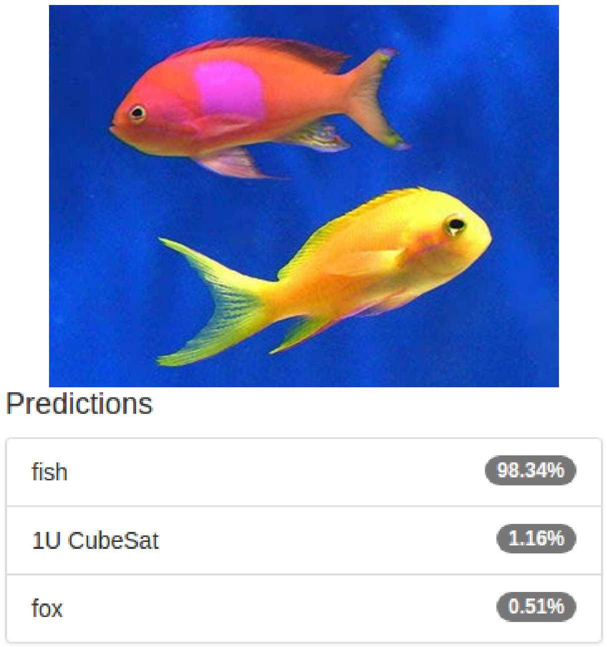
Figure 9: imageNet classification results for a 3D-printed 1U-sized CubeSat frame created at COSMIAC—the model calculates there is a 99.96% chance this image contains a 1U CubeSat; photo credit: COSMIAC.



Predictions

fox	98.59%
fish	1.39%
1U CubeSat	0.02%

Figure 10: imageNet classification results for image of a fox—the model calculates there is a 98.59% chance this image contains a fox; photo credit: Google Images.



Predictions

fish	98.34%
1U CubeSat	1.16%
fox	0.51%

Figure 11: imageNet classification results for image of fish—the model calculates there is a 98.34% chance this image contains fish; photo credit: Google Images.

The second task consisted of object detection and localization with detectNet. The detectNet package takes a 2D image as input, locates specified objects within the image frame and creates bounding boxes around them, and then produces a list of coordinates of the detected bounding boxes. In order to train an object detection model, a pretrained ImageNet recognition neural network model such as AlexNet or GoogLeNet is first used, in which the training images contain bounding box coordinate labels.[3]

Expanding upon the simple image classification provided by imageNet, detectNet not only identifies multiple types of objects in an image, it also locates and provides their coordinates. This enables a colored bounding box to be drawn over each object of a specified type. Bounding boxes will even track moving objects in a video or live stream, and will appear and disappear as objects enter and leave the field of view. The ability to identify and locate multiple different types of objects within a single image provides increased potential capability compared to simply classifying an entire image as probably being one object or another.

Nvidia offers pre-trained detectNet models for several types of objects, including pedestrians, bags or luggage, faces, airplanes, liquid container bottles, chairs, and dogs. Unfortunately, no readily available or open source libraries for detecting small spacecraft exist. For practice, we first experimented with the pedestrian and luggage pre-trained models. From the third floor of the COSMIAC research lab, the TX1 was able to detect two pedestrians and a backpack, as seen in Figures 12 and 13:

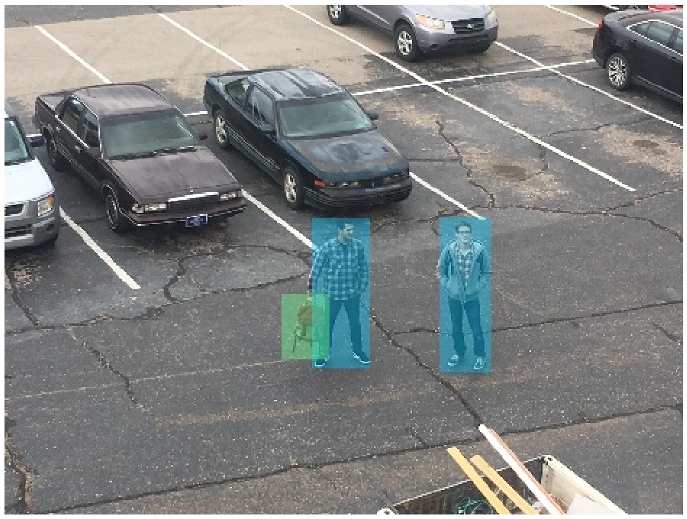


Figure 12: Nick Buonaiuto (holding backpack) and Casey Ottesen in the COSMIAC parking lot demonstrating pedestrian and luggage detection; photo credit: Brian Zufelt.

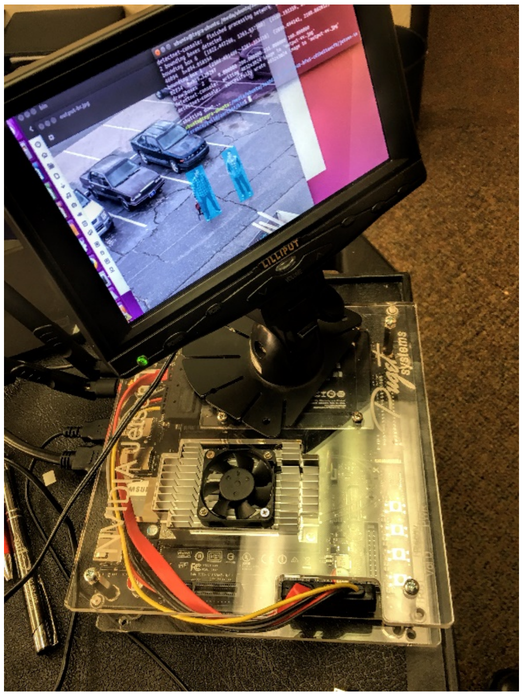


Figure 13: NVIDIA Jetson TX1 development board demonstrating pedestrian detection at COSMIAC. The Jetson board is housed within a clear protective case, and the fan and heatsink of the TX1 SoC can be seen just below the monitor stand; photo credit: Brian Zufelt.

Training a customized detectNet model to locate 3U-sized CubeSats took approximately 19 hours, and involved feeding several hundred images through the neural network. This length of time could be shortened by utilizing more powerful GPU hardware with increased VRAM. Detection of 3U-sized CubeSats can be seen in Figures 14 and 15. The localization of the two CubeSats, however, is not perfect: at greater distances, the model is not able to draw separate bounding boxes around each object, as seen in Figure 15. Correcting this would be a matter of increased training using more images, and would ideally be accomplished using the aforementioned powerful GPU with 8Gb+ of video memory in order to reduce processing time.

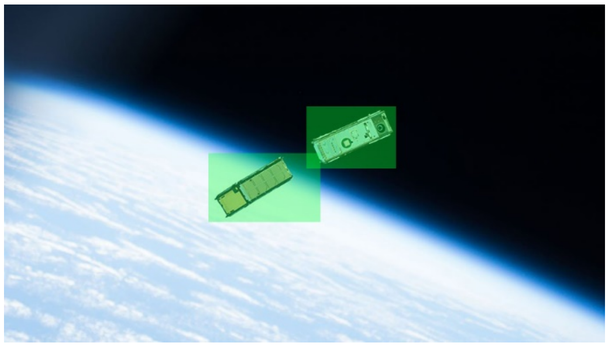


Figure 14: Close-up of two 3U-sized CubeSats following NanoRacks deployment; photo credit: NanoRacks.

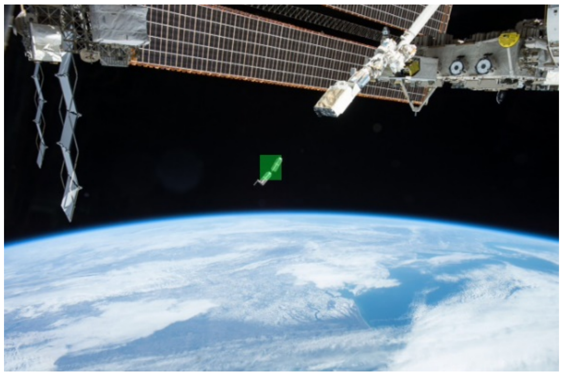


Figure 15: Two 3U-sized CubeSats just after NanoRacks deployment from the International Space Station (ISS). Notice at greater distance the model does not distinguish two separate objects; photo credit: NanoRacks.

The third and final image processing task consisted of image segmentation with segNet. The segNet package takes an image, identifies different object types based on a customizable list, and highlights objects of each type in different colors. Segmentation is similar to recognition and detection in that different objects are identified and located within a field of view. However, this classification occurs "at the pixel level as opposed to classifying entire images as with image recognition" or to locating a given set of objects within an image as with detection.[3] Segmentation, therefore, allows for the possibility of every unique object or surface within an image to be separately identified and located. This can be as simple as a flying drone separating ground from sky, or as complex as a driverless automobile safely navigating through a crowded urban environment.

As with imageNet and detectNet, the interface for training segNet models is DIGITS, via either command-line or web browser. Utilizing segNet essentially requires three data items: The first item is an image or stream of images in which to locate and separate objects. The second item is a text file containing the names of the object types being separated—these names are equivalent to the labels for object classification. The third item is another text file containing Red-Green-Blue (RGB) values to associate with the label of each object.

For example, first-person perspective video taken from a flying aerial drone will consist of a stream of images that could be said to contain two basic types of objects: land (terrain) and sky, as seen in Figure 16. A text file for labels would be created containing the lines “terrain” and “sky”. Another text file for colors would be created containing RBG color values corresponding to the labels: “0 255 0” is green for terrain and “0 0 255” is blue for sky.

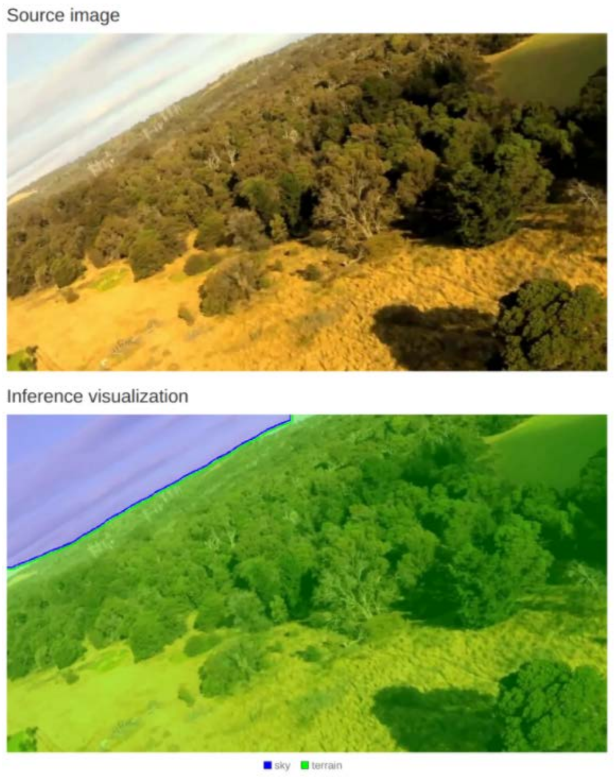


Figure 16: Example of aerial drone sky vs. ground segmentation with segNet; this example model finds “terrain” and “sky” and colors those regions green and blue, respectively.[3]

Our experiments replicated the ground and sky segmentation using the provided segNet pre-trained model to separate terrain and sky, seen in Figure 16, but with original aerial drone footage taken by COSMIAC, seen in Figure 17. Notice the segmentation is not perfect, and portions of land are being classified as sky. This can be solved with increased training using a more robust database of labelled training images.

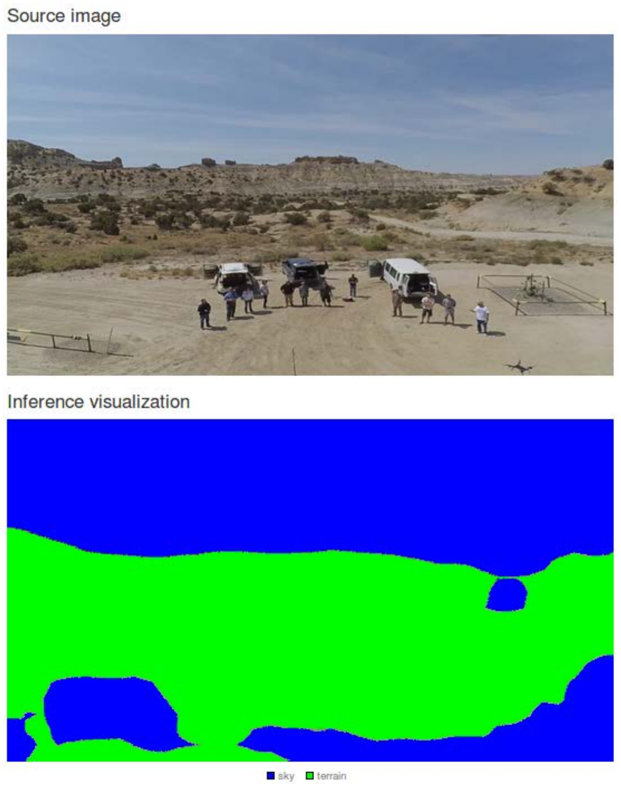


Figure 17: Aerial drone picture of COSMIAC team in New Mexico, with segmentation of sky (blue) vs. terrain (green), although some portions of ground are incorrectly classified as sky; source photo credit: COSMIAC.

Though it is quite forward-thinking, image segmentation as applied by driverless road vehicles could theoretically be applied to direct traffic for pilotless space vehicles (e.g. satellites). In an orbital environment, image segmentation could be performed to separate planets or other objects from space or from each other, as seen in Figure 18. The example in Figure 18 uses an image of the Earth and Moon in space (image created at COSMIAC) and the same classification model as the aerial drone image in Figure 17 (which segments terrain and sky). Even though the model provided by segNet is nominally trained for “terrain” and “sky”, it is still able to segment the different types of objects in Figure 18, i.e. space vs. Earth and Moon. This classification occurs with an interesting reversal, in that the Earth and Moon objects are technically classified as “sky” and space is classified as “terrain”. However, the Earth object in frame is comprised of blue ocean and white clouds—very similar in appearance to sky. Furthermore, the view of the Earth and Moon from space could be considered to be a view of their skies. In any event, the object segmentation is sound, and it is possible to solve away these errors with longer training times using more images and better-optimized neural networks, or by simply reassigning the labels and colors in the text files.

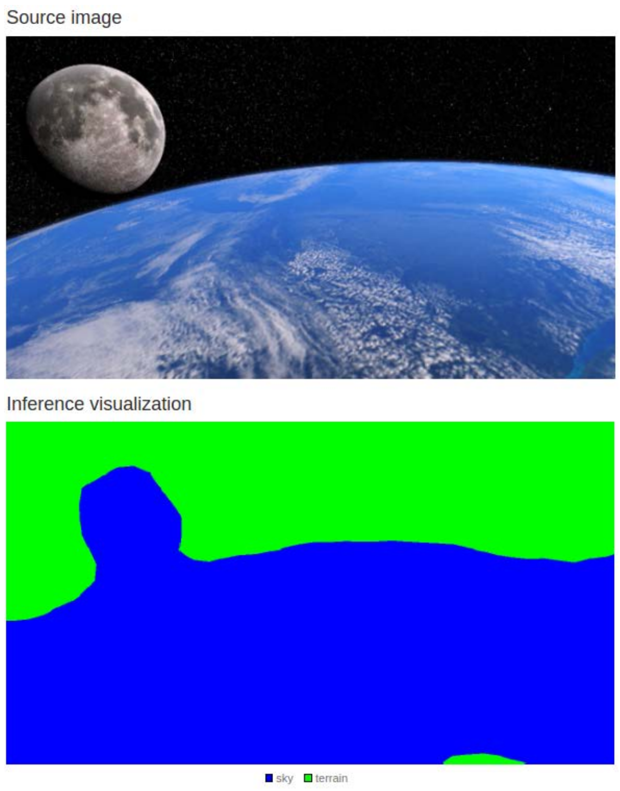


Figure 18: Segmentation of the Earth and Moon (blue), and space (green) using a source image created at COSMIAC. Notice that although terrain and sky are nominally labelled incorrectly, the segmentation of object types is rather good, and could be improved simply by editing the associated label and color text files; source photo credit: Brian Zufelt.

IV. TECHNICAL CHALLENGES AND PROGRESS

Running on the TX1 simulates potentially running on board of a small satellite. Challenges naturally include the volume of space and the distances between objects—higher resolution images are more difficult to obtain at greater distances. Objects moving very fast also makes it difficult to obtain clear high-resolution images. In addition, the lack of a large and robust image database of spacecraft that can be used for model training limits much of what can be accomplished.

The challenge to fast and accurate models that is most able to be influenced by the user is the hardware implemented for training. But even with advanced hardware, model training times can become intractable if the training database grows exponentially large—adding one item to an image database means adding at least several dozen (if not hundreds or thousands) of individual images of that object. Each individual image is more work (i.e. more tensor objects) for the convolutional neural network. The most important hardware to consider is the GPU, with the GPU's amount of VRAM being the most important specification. Maximizing processing power is difficult because GPU hardware, and especially Nvidia GPU hardware, tends to be very financially expensive for units containing more than 8Gb of VRAM. Fortunately, the proliferation of virtual reality technology, which theoretically requires twice as much VRAM to render separate image frames for each eye, is having an enabling effect on deep learning with GPUs—as VR applications become more commonplace, so does the hardware required to run them. Prices for GPUs with 8Gb of VRAM are now lowering to match GPUs that had 4Gb of VRAM just two years ago.

The accuracy of the recognition, detection, and segmentation being performed could always stand to be improved. Even with the aforementioned lengthy training times, models can still makes mistakes, e.g. imageNet could still classify images incorrectly, detectNet could still fail to correctly locate objects, and segNet could still incorrectly draw boundary lines when separating objects. However, even these mistakes are sometimes able to be interpreted, which helps with optimizing the types of images required for efficient training. For example, notice that in Figure 19, almost the entire Earth along with the majority of the moon are being classified as the same object type. The fact that this object type is nominally classified as "sky" and assigned the color "blue" is merely a preprocessing decision that can be adjusted. Similarly, while space and the darker portion of the Moon (as well as slivers of the Earth containing clouds) are being classified as "terrain" and colored green, this can be adjusted as well.

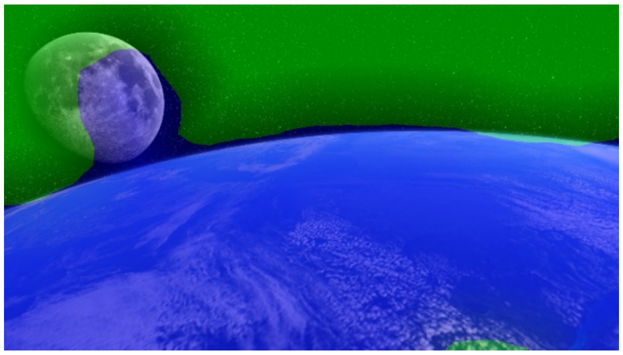


Figure 19: Layered images from Figure 18, highlighting that even imperfect segmentation is somewhat able to be interpreted; photo credit: Brian Zufelt.

Improving model accuracy could occur in two ways: First, image classification accuracy could be improved via the brute-force method of simply spending more time training with larger databases of more objects and increased numbers of labelled training images. Eventually, models would see enough examples of objects from every angle and in every lighting condition so that any possible future image of the same object type would be recognized. Well-trained models could be based on databases of images for every object in the dictionary, for example. Second, classification accuracy could be also be improved by utilizing different and better neural network models than AlexNet and GoogLeNet (such as customized or proprietary models), though this goes beyond the scope of this experiment.

The primary progress would be deploying increasingly autonomous systems on more types of satellites, and Nvidia's TX1 hardware goes to great lengths to show that capability. The TX1 is not limited to image processing, and is a fully capable system on a chip that provides both central and graphics processing. Image processing tasks lend themselves easily to GPU hardware, but really any data-intensive deep learning task is well-suited to the parallel scalability of GPU processing.

V. SUMMARY

The Nvidia Corporation is making great effort to provide data scientists and engineers with the tools required to perform efficient deep learning tasks using off-the-shelf hardware and open-source software. Powerful processing capability in a small package such as the TX1 enables this learning to occur onboard the data-collecting instruments—rather than transmitting raw data to the ground, satellites could send processed intelligence. Similarly, instead of waiting to receive an update package from the ground, a satellite could process its own data and apply corrections automatically. This level of increased space situational awareness is the goal of applying Nvidia GPU technology to data collected by satellites.

The three image processing tasks of recognition, detection, and segmentation can be applied to any type of image. On-ground experiments training models to recognize, detect, and separate two different sizes of cube satellites were successful using both images and live 3D-print models. However, this does not guarantee the process can be exactly replicated in space. Further training and testing using images obtained from space would be a beneficial step on the way to conducting actual tests in orbit.

The onboard processing capability enabled by Nvidia hardware and software can reduce data requirements for missions and expand the types of missions that small satellites and CubeSats are used for. An example space situational awareness application of onboard satellite image processing would be the ability for a spacecraft to point out areas of interest, identify and locate objects it determines may be relevant, and then execute an autonomous course of action, rather than downloading massive arrays of images for post-processing on the ground.

VI. FUTURE WORK

For future work, both COSMIAC and SGT are currently involved in nanosatellite projects with organizations such as the Air Force Research Laboratory (AFRL) and the National Aeronautics and Space Administration (NASA). This has caused teams of scientists, engineers, and students to become well-versed in a wide variety of different satellite configurations and missions. Current activities are also underway at COSMIAC and SGT in the areas of machine learning and big data analytics. The big data aspects incorporate a multitude of open source software technologies that have made data processing and mining faster and more efficient than ever before. Additionally, with cloud computing becoming increasingly prevalent and inexpensive, the capability to acquire virtual hardware for model training is almost limitless.

For long term activities, the team at COSMIAC would be interested in building and flying a payload imager upon the International Space Station for future studies of model deployment.

REFERENCES

[1] K.G. Santhanam, “The Anatomy of Deep Learning Frameworks,” in KD Nuggets Blog, February 2017. Retrieved from: http://www.kdnuggets.com/2017/02/anatomy-deep-learning-frameworks.html

[2] NVIDIA, “Jetson Embedded Platform”, NVIDIA Corporation 2017. Retrieved from: http://www.nvidia.com/object/embedded-systems-dev-kits-modules.html

[3] D. Franklin. “Guide to Deploying Deep-Learning Inference Networks and Deep Vision Primitives with TensorRT and Jetson TX1/TX2.” NVIDIA Corporation, 2017. Retrieved from: https://github.com/dusty-nv/jetson-inference

[4] NVIDIA, “DIGITS Interactive Deep Learning GPU Training System,” NVIDIA Corporation 2017. Retrieved from: https://developer.nvidia.com/digits

[5] NVIDIA, “Jetson TX1 Developer Kit,” NVIDIA Corporation 2017. Retrieved from: https://developer.nvidia.com/embedded/buy/jetson-tx1-devkit

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” in IEEE Computer Vision and Pattern Recognition (CVPR), 2009. Retrieved from: http://www.image-net.org/papers/imagenet_cvpr09.pdf

[7] A. Krizhevsky, I. Sutskever, G.E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks,” in Advances in Neural Information Processing Systems 25 (NIPS 2012). Retrieved from: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks

[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, “Going Deeper with Convolutions,” in IEEE Computer Vision and Pattern Recognition (CVPR), 2015. Retrieved from: http://www.cvfoundation.org/openaccess /content_cvpr_2015/papers/Szegedy_Going_Deeper_With_2015_CVPR_paper.pdf

[9] NVIDIA, “JetPack,” NVIDIA Corporation 2017. Retrieved from: https://developer.nvidia.com/embedded/jetpack

[10] Y. Jia, E. Shelhamer, “Caffe Deep Learning Framework,” Berkeley Vision 2014. Retrieved from: http://caffe.berkeleyvision.org/

[11] R.G. Gomez-Ol, “Deep Learning frameworks: a review before finishing 2016,” Medium.com 2016. Retrieved from: https://medium.com/@ricardo.guerrero deep-learning-frameworks-a-review-before-finishing-2016-5b3ab4010b06

[12] M. Rumanek, T. Danek, and A. Lesniak, “High Performance Image Processing of Satellite Images Using Graphics Processing Units,” in Geoscience and Remote Sensing Symposium (IGARSS), 2011 IEEE International, 2011, pp. 559-561. Retrieved from: http://ieeexplore.ieee.org/document/6049189/

[13] NVIDIA, “cuDNN GPU Accelerated Deep Learning,” NVIDIA Corporation 2017. Retrieved from: https://developer.nvidia.com/cudnn

[14] NVIDIA, “TensorRT Deep Learning Inference Optimizer and Runtime Engine,” NVIDIA Corporation 2017. Retrieved from: https://developer.nvidia.com/tensorrt

[15] G.J. Scott, K. Backus, D.T. Anderson, “A Multilevel Parallel and Scalable Single-Host GPU Cluster Framework for Large-Scale Geospatial Data Processing,” in Geoscience and Remote Sensing Symposium (IGARSS) 2014 IEEE International, pp. 2475-2478, July 2014. Retrieved from: http://ieeexplore.ieee.org/document/7325718/references?ctx=references