US Army Corps of Engineers®
Engineer Research and
Development Center
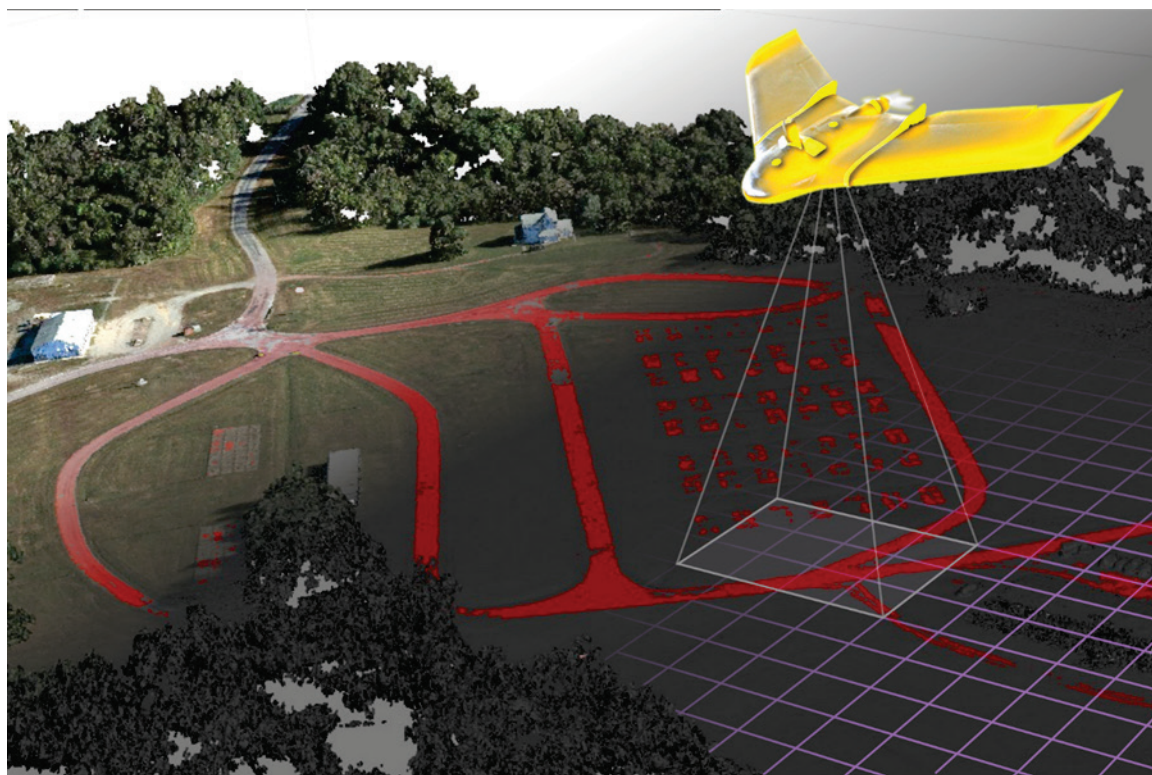
Geospatial Research Laboratory

Geointelligence – Geospatial Data Analysis and Decision Support

Use of Convolutional Neural Networks for Semantic Image Segmentation Across Different Computing Systems

Andmorgan R. Fisher, Timothy A. Middleton, Jonathan Cotugno, Elena Sava, Laura Clemente-Harding, Joseph Berger, Allistar Smith, and Teresa C. Li

March 2020

Use of Convolutional Neural Networks for Semantic Image Segmentation Across Different Computing Systems

Andmorgan R. Fisher, Timothy A. Middleton, Jonathan Cotugno,
Elena Sava, Laura Clemente-Harding, Joseph Berger, Allistar Smith,
and Teresa C. Li

Geospatial Research Laboratory
U.S. Army Engineer Research and Development Center
7701 Telegraph Road
Alexandria, VA 22315-3864

Final Report

Approved for public release; distribution is unlimited.

Abstract

The advent of powerful computing platforms coupled with deep learning architectures have resulted in novel approaches to tackle many traditional computer vision problems in order to automate the interpretation of large and complex geospatial data. Such tasks are particularly important as data are widely available and UAS are increasingly being used.

This document presents a workflow that leverages the use of CNNs and GPUs to automate pixel-wise segmentation of UAS imagery for faster image processing. GPU-based computing and parallelization is explored on multi-core GPUs to reduce development time, mitigate the need for extensive model training, and facilitate exploitation of mission critical information. VGG-16 model training times are compared among different systems (single, virtual, multi-GPUs) to investigate each platform's capabilities.

CNN results show a precision accuracy of 88% when applied to ground truth data. Coupling the VGG-16 model with GPU-accelerated processing and parallelizing across multiple GPUs decreases model training time while preserving accuracy. This signifies that GPU memory and cores available within a system are critical components in terms of preprocessing and processing speed. This workflow can be leveraged for future segmentation efforts, serve as a baseline to benchmark future CNN, and efficiently support critical image processing tasks for the Military.

Contents

Figures and Tables

Figures

Tables

Preface

This study was conducted for the Geospatial Research Laboratory (GRL) under PE 62784/ Proj 855/ Task 23, "GeoICUE/TREADSTONE." The technical monitor was Dr. Jean D. Nelson.

The work was performed under the Data and Signature Analysis Branch (TRS) of the TIG Research Division (TR), U.S. Army Engineer Research and Development Center, Geospatial Research Laboratory (ERDC-GRL). At the time of publication, Ms. Jennifer L. Smith was Chief, Data Signature and Analysis Branch; Ms. Martha Kiene was Chief, TIG Research Division; and Mr. Ritchie Rodebaugh was Director of the Technical Directorate. The Deputy Director of ERDC-GRL was Ms. Valerie L. Carney and the Director was Mr. Gary Blohm.

COL Teresa A. Schlosser was Commander of ERDC, and Dr. David W. Pittman was the Director.

1   Introduction

1.1   Background

Machine learning applications have become increasingly popular due to the wide availability of large volumes of high resolution remotely sensed data and the decreasing cost for high performance computing (HPC) infrastructure. Data driven approaches in the fields of image processing and computer vision have advanced geospatial research and can be leveraged to determine mission critical information. Yet, deriving actionable information from large sets of complex imagery in an efficient manner remains a central challenge of geospatial intelligence (GEOINT) gathering. Rapid, accurate classification of aerial imagery provides decision makers with functional information that enables operational understanding and resource allocation according to identified constraints. Imagery collection by military units using diverse sensors onboard unmanned aerial systems (UAS) have increased significantly over the last decade and is expected to continue to rise. This suggests that the number of autonomous operations, collection of high definition geolocated images, and HPC systems that can be deployed to the tactical edge will increase. As such, there has been a commensurate need for faster automated image processing both for low level tasks, such as de-noising or segmentation, and high-level tasks, such as classification (Castelluccio et al. 2015).

Semantic segmentation of predefined classes in aerial imagery involves assigning a class label to each pixel, thus, partitioning the image into meaningful segments (or sets of pixels). Semantic segmentation is a core problem in the field of computer vision and is a key task that enables complete geospatial understanding of a scene. Automated segmentation has many important real-world applications, including autonomous driving (Cordts et al. 2016), augmented reality, land cover mapping (Matikainen and Karila 2011), change detection (Tang and Zhang 2017), environmental monitoring, and urban planning (Karantzalos 2015).

Machine learning techniques, such as convolutional neural networks (CNNs), have shown enormous success when applied to image segmentation problems for red, green, blue (RGB) imagery (Marnamis et al. 2016). A central challenge hindering broad adoption of such algorithms has been hardware limitations. The introduction of massively parallel

computing architectures resulted in hardware that can now more efficiently manage workloads related to computer vision algorithms (Kyrkou et al. 2018).

This document highlights the use of CNNs and graphics processing unit (GPU) acceleration to address the need for an automated pixel-wise image segmentation approach for imagery collected by UAS. Pixel-wise image segmentation is a challenging problem because each pixel is assigned a label based on the local context surrounding the pixel, making it a computationally intensive task. This report addresses the challenge by implementing a pre-trained CNN and transfer learning to provide a series of time-resolved images with dynamic attributes. Additionally, GPU-based computing and parallelism is explored on multi-core GPUs to reduce development time, mitigate the need for extensive model training, and facilitate the exploitation of mission critical geospatial information. Computation times to train the CNN model are compared among three different systems (single GPU, virtual GPU, and multi-GPU). Thus, this work delivers a useful and flexible tool applicable to a range of geo-intelligence tasks commonly undertaken by the U.S. Army Corps of Engineers (USACE) and the intelligence community (IC), in general. Such tasks include: three-dimensional (3-D) modeling, survey and mapping, inspection, and maneuver support. The CNN model is tested and validated on imagery collected over three distinct sites and tested across different computing platforms to assess model efficiency and accuracy.

1.2   Related work

1.2.1 Artificial neural networks (ANNs)

Artificial Neural Networks (ANNs) are widely used for image classification tasks (Goodfellow et al. 2016; Kernell 2018). ANNs are inspired by biological neural networks in the human brain that have been designed to recognize patterns. A neural network consists of multiple layers of neurons, where each layer can be thought of as a single processing unit that can take multiple inputs, but has only one output (Wang and Xi 2012). A representation of a standard neural network structure is shown in Figure 1(a).

Figure 1. Representation of a standard neural network with two hidden layers is shown on the left. On the right, the same network is shown after dropout is applied. The crossed neurons have been dropped. Figure taken from Srivastava et al. 2018.

(a) Standard Neural Net   (b) After applying dropout.
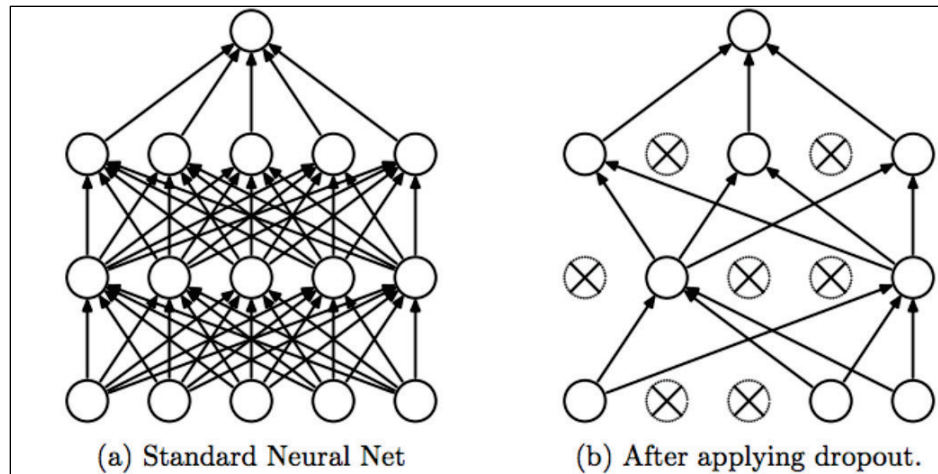
Large neural networks serve as powerful image classification and segmentation tools (Kernell 2018). Activation functions are a critical component of ANNs as they provide information on whether a neuron must be activated and if the information received is relevant or should be overlooked. Adding an activation function to a node introduces non-linearity to the network. This enables the network to learn from complex data and provide accurate predictions (Goodfellow et al. 2016).

Activation functions are categorized into two types - linear or nonlinear. The Step function represents a simple activation function that is often used with binary classifiers, and the outputs are either 1, indicating activated, or 0, indicating not activated (Goodfellow et al. 2016). In the past, common choices for neural network activation functions were the Sigmoid and hyperbolic tangent (TANH) functions as they are easily differentiable and non-linear. However, these functions result in dense activations which are computationally expensive (Kernell 2018). Modern neural networks commonly use the Rectified Linear Unit (ReLU) function because it results in sparse and efficient activations making it less computationally expensive (Parashar et al. 2017).

A graphical representation of each activation function is shown in Figure 2. Note that the Step function (a) suppresses the input data into a set range (0,1), while the Sigmoid (b) and TANH (c) functions allow the model to generalize and adapt to the input data, forming an "S" shaped curve. For example, graphs (b) and (c) show that the function is limited to the upper and lower bounds. The ReLU (d) graph shows the function

creating a positive linear slope only when values exceed zero. The flexibility of the linear behavior (continuous values) associated within each activation function allows CNN models to be optimized more easily than activation functions that exhibit stricter behavior such as the Step function (discrete values) (Kernell 2018). For instance, when ReLU function values are less than zero, they get discarded, while values greater than zero vary linearly with no upper limit bounds.

Figure 2. Graphical representation of slope for commonly used activation functions. Recreated from Kernell (2018).
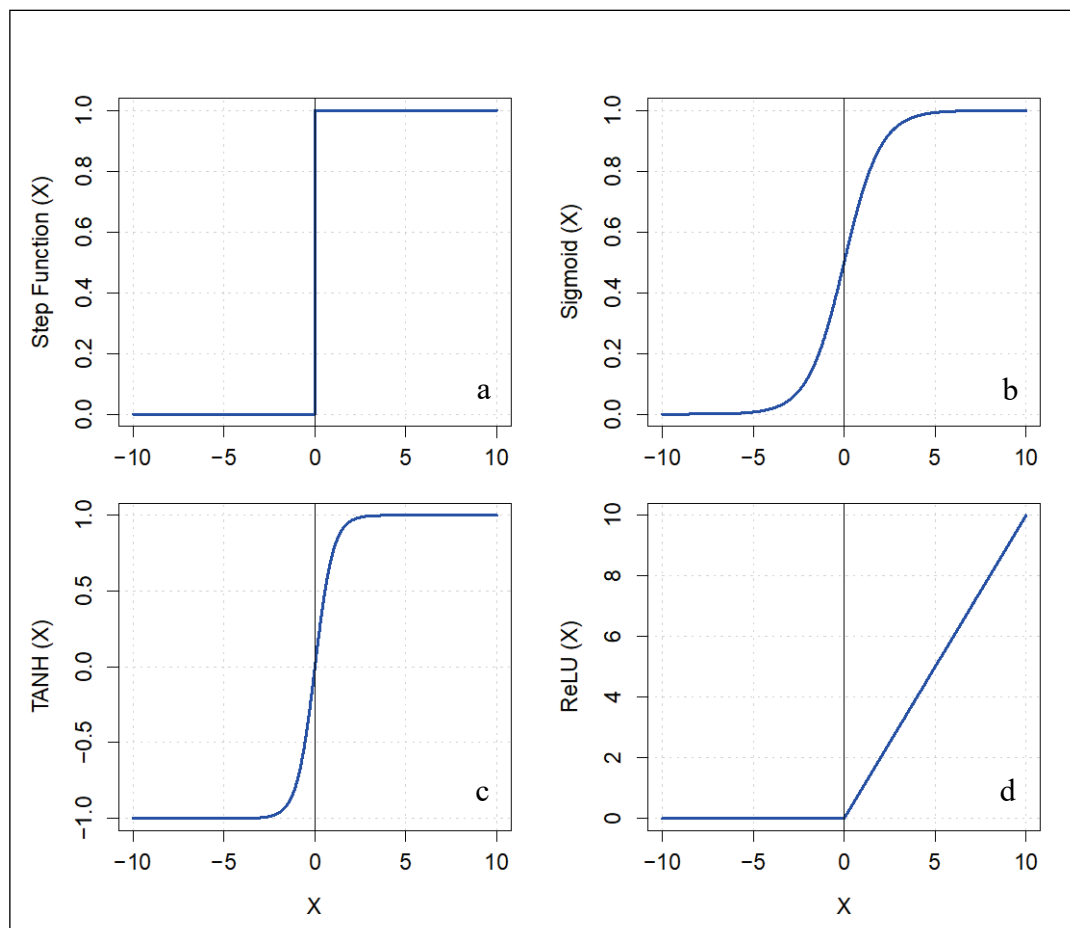
Model optimization entails finding the set of parameters, or weights, that minimize the loss function (Garcia-Garcia et al. 2017). This loss function is used to estimate how well the model is trained by calculating the difference between the model's predicted values and the observed input data values as well as determine the optimal model weights. The model weights (or parameters) are then adjusted using a gradient descent optimization algorithm which identifies the average error of the loss function in the training data. A comprehensive review of gradient descent

optimization algorithms conducted by Sebastian Ruder (2017) found the Adam optimizer to be the best overall choice of optimizing algorithms. The Adam algorithm updates model weights iteratively based on the input data and is selected for this work as it is (1) straightforward to implement; (2) computationally efficient; (3) well-suited to big data problems; and (4) requires little tuning of hyperparameters (Srivastava et al. 2014; Ruder 2017).

A common challenge when leveraging CNN's for image segmentation on large data is preventing the model from overfitting when training on the input data. Overfitting prevents the model from identifying and learning different relationships in the data, thus reducing its ability to make predictions when applied to new data. For this reason, regularization is used as a means to constrain and prevent the network from overfitting the data (Garcia-Garcia et al. 2017). The following three types of regularization are routinely used with deep learning algorithms: L1, L2, and dropout. L1 and L2 are two of the most commonly used types of model regularization. L1 penalizes the absolute value of the weights, which allows the weights to be reduced to zero and is useful when model compression is desired. L2 forces the weight values to decay approaching zero (Garcia-Garcia et al. 2017; Kernell 2018). Alternatively, dropout is a means of regularization that temporarily and randomly removes neurons, as well as all their incoming and outgoing connections, from the network as shown in Figure 1(b) (Karpathy 2018). Because the network is prevented from using the same neurons repeatedly, this method serves to improve the generalization of the network (Kernell 2018).

A deep convolutional neural network developed by the Visual Geometry Group (VGG) from University of Oxford was used to segment aerial imagery into predefined classes. The VGG-16 model architecture is comprised of 16 weighted layers and is freely available for download online through GitHub. The VGG-16 is distinguished from other CNNs by the use of many convolutional layers with progressively smaller receptive fields, as opposed to only a few convolutional layers with large fields (Parashar et al. 2017). The receptive fields in a CNN model represent the center area of focus (spatial extent) the model is looking at, as each convolution is applied. These fields are commonly thought of as the filtering size window applied in image processing tasks. The overall architecture of VGG-16 is shown in Figure 3. The network's convolutional layers are represented using white rectangles of various dimensions, followed by a pink max

pooling layer and are divided into five groups. The last two convolutional layers, represented in blue, are flattened through a fully connected layer and softmax is used to perform the image segmentation. This architecture was chosen because the model is simpler to train due to the reduced number of parameters and an increased number of nonlinearities in between the neuron layers allowing the model to efficiently learn and achieve better performance. This is important because it makes the decision function more discriminative between the different classes (Parashar et al. 2017).

The VGG-16 is fine-tuned using a loss function which gives the network the ability to penalize mistakes, thus reducing the error when making predictions (Kernell 2018). Loss functions are used to estimate the error of a set of weights in a neural network. Neural networks are typically trained using stochastic gradient descent to calculate error in model predictions. These calculations are used to update weights in the network using backpropagation of the error algorithm. Since neural networks are essentially classification algorithms which yield a probability distribution as output, cross-entropy is arguably the most commonly used loss function (Goodfellow et al. 2016; Garcia-Garcia et al 2017). This is due to the inverse relationship between the model's prediction performance and the cross-entropy function. For instance, if the cross-entropy loss value is high the model's predictive performance decreases and diverges from the actual label associated with a class.

Figure 3. Fully convolutional neural network based on the VGG-16 architecture.
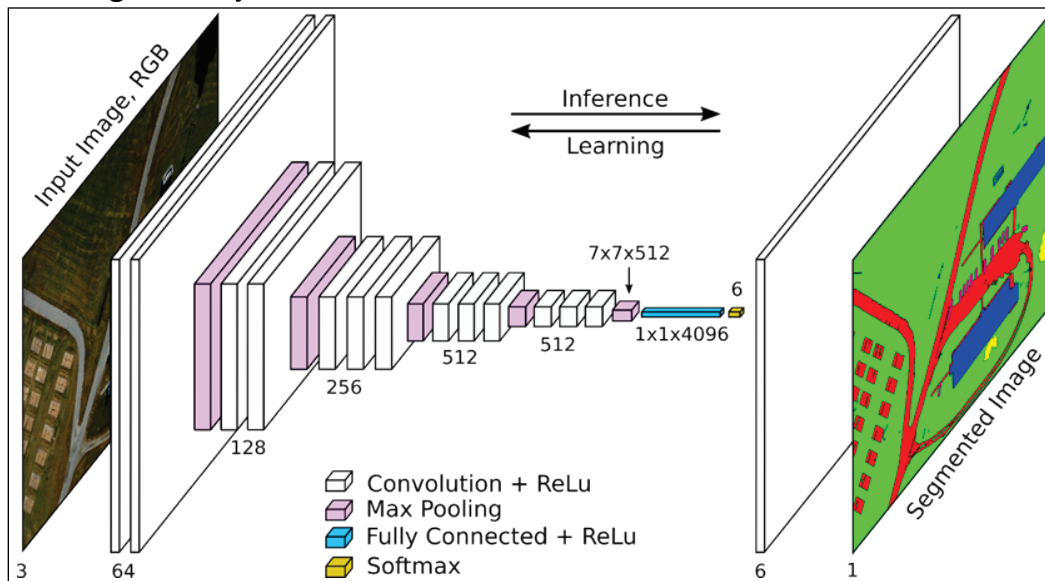
1.2.2 Semantic segmentation

The advent of powerful computing platforms coupled with deep learning architectures have recently resulted in novel approaches used to tackle many traditional computer vision problems. This progress is reflected in commercial-off-the-shelf (COTS) software and in academic research that leverages traditional and deep learning methods to automate the interpretation of geospatial information (Kingma Ba 2017; Becker et al. 2017). For example, Trimble's eCognition® software uses Google's +TensorFlow™ library to construct CNNs and automate various analytic tasks such as the segmentation of satellite and aerial imagery for land cover mapping, object detection, and point cloud classification (Becker et al. 2017).

Semantic segmentation of aerial imagery involves assigning a semantic (or object class) label to each pixel in a given image. There are several challenges that make this a difficult task. For example, imagery often contains complex objects of varying sizes, which can make simultaneous segmentation a challenge (Trimble Geospatial). Another example is that redundant object details often occur when resolution is improved, thereby increasing the difficulty of segmentation (Trimble Geospatial).

Nonetheless, neural networks have become well-known models for feature learning. Academic literature has shown an increasing number of applications of CNNs to geospatial problems using geospatial data. Many of these applications are directly applicable to GEOINT gathering including: semantic segmentation of aerial imagery (Wang et al. 2017; Chen et al. 2018), extraction of digital terrain models from laser scanning point cloud data (Volpi and Tuia 2017), scene (Hu and Yuan 2016) and land use classification (Castelluccio et al. 2015), crop yield production (Hu et al. 2015), and automated target recognition (You et al. 2017).

The performance of semantic segmentation systems can be evaluated based on metrics such as execution time, memory footprint, and accuracy (Parashar et al. 2017). Since the majority of systems are constrained by the time that can be spent on the inference pass, speed or runtime is a particularly valuable metric. Another important consideration for segmentation methods is memory usage. In fact, even GPU-accelerated neural networks are often constrained by the number of GPU cores and memory capacity (Long et al. 2014). Pixel accuracy is arguably the most important metric and generally used to gauge the performance of semantic

segmentation techniques (Parashar et al. 2017). In this work all three metrics are used to assess algorithm performance and segmentation accuracy.

2 Data, Sites and Computing Platforms Description

2.1 Data characterization

The eBee drone is a fully autonomous, commercially available fixed wing UAS used to capture aerial imagery. It is a hand-launched drone weighing approximately 1.5 lbs with a 37.8 in. wingspan. The eBee drone was purchased from senseFly, a subsidiary of Parrot Group. For this research effort, the eBee drone was equipped with a 12-megapixel Canon Powershot S110 and programmed to autonomously capture images at nadir (camera lens perpendicular to ground) from approximately 50 m above the average scene elevation. To optimize the overlap of neighboring images, an interleaving raster flight pattern was utilized. Interleaving mode performs two traces, a standard trace and retrace, followed by a second trace and retrace with interleave feedback enabled. At each site, between 10 to 17 images were collected as the image collection was limited by the battery endurance of the UAS. The collection spanned $1.25\pm0.5 \times 10^5 \, m^2$ (30.9±12.4 acres) in total aerial coverage from each site. Each image is 4000 x 3000 pixels with 2.5 cm spatial resolution collected in the RGB part of the electromagnetic spectrum. Three different research sites were used for this effort: Fort AP Hill (Virginia), Camp Cook (Louisiana), and Fort Campbell (Tennessee and Kentucky). Images were stored in the Joint Photographic Experts Group (JPEG) format, which makes use of exchangeable image file format (ExIF) tagging structure in order to store embedded metadata (i.e. georeference data). Scenes contain heterogeneous class representations including, but not limited to, buildings, roads, trees, cars, etc.

2.2 Sites description

2.2.1 Fort AP Hill

Fort A.P. Hill is a U.S. Army Installation located near Bowling Green, Virginia, approximately 65-75 miles south of GRL. The Garrison Commander was Lieutenant Colonel (LTC) Michael Gates and the Garrison Command Sergeant Major (CSM) was Joseph Reilly. Fort A.P. Hill is a Regional Collective Training Center established in June 1941 and is used to support Army, Joint, and Interagency readiness. Fort A.P. Hill spans 76,000 acres of land and is one of the largest East Coast military

installations. Imagery collection from this site contains a variety of standard land surfaces and features (i.e. grass, pavement, concrete, and vegetation), along with a number of buildings. Data collected at this site provides a range of standard samples of information.

2.2.2 Camp Cook

Camp Cook is located in Ball, Louisiana and was established in 1941. It is home to the Louisiana National Guard and the Noncommissioned Officers Academy (NCOA). The Army's Basic Leader Course (BLC) is offered at this location by the 1st Battalion NCOA. The Commandant was CSM Christopher Maxwell and Deputy Commandant was First Sergeant (1SG) Thomas Hughes. Imagery collected at Camp Cook contains similar content to the Fort A.P. Hill collection in terms of man-made features. The environment introduces additional vegetation types and different land features (i.e. creek beds) into the training dataset.

2.2.3 Fort Campbell

Fort Campbell is located at the border of Tennessee and Kentucky and spans approximately 102,000 acres. The site was identified in 1941 and base construction commenced in 1942. The Army's 160th Special Operations Aviation Regiment (SOAR), 101st Airborne Division, 5th Special Forces Group, and 19th Air Support Operation Squadron are located at Fort Campbell. The primary mission of this base is to support combat readiness for air assault training. The expanse has a variety of natural and man-made features that add further information to the training dataset.

2.3 Computing platforms description

The automated image segmentation workflow was tested on three different systems where each system represents different computing capabilities. These platforms are the HP Omen laptop (single GPU), Army Geospatial Enterprise (AGE) Node (single virtual GPU), and the Nvidia DGX station (Multi-GPU enabled). Specifications for each platform are summarized below and in Table 1. The use of varying computing platforms allows for a comprehensive comparison and summary of information in terms of processing speeds between systems relative to their embedded hardware configurations.

2.3.1 HP OMEN 17t

The HP OMEN 17t is a gaming laptop equipped with Nvidia's latest GPU hardware. Nvidia is a computer technology company specializing in parallel computing and graphics (Nvidia). Nvidia is also the inventor of the GPU technology, which has become the state-of-the-art technology for computing. The automated segmentation workflow was implemented on the HP OMEN 17t, which is a laptop system typically used for PC gaming because of the hardware. The HP OMEN 17t has an 8GB GTX 1070 graphical processing unit integrated into the system This laptop was used as a way to provide comparative metrics at different levels of computational capabilities when compared to more advanced hardware platforms.

2.3.2 Army Geospatial Enterprise Node

The automated segmentation workflow was also performed on a CentOS 7 virtual machine (VM) maintained by the AGE Node. A VM is an emulator that mimics the behavior of a separate computer and is capable of running applications as a separate computer. The AGE Node provides computing resources and maintenance to project developers where applications can be tested in the early stages of their lifecycle.[*] Nvidia GRID virtual graphical processing unit (vGPU) enables multiple virtual machines to have simultaneous, direct access to a single physical GPU, using the same Nvidia graphics drivers that are deployed on non-virtualized operating systems (Simonyan and Zisserman 2015). The benefit of this functionality is the ability to implement the automated segmentation workflow from any computer, while still utilizing the Nvidia GRID vGPU resource for processing.

2.3.3 DGX Station
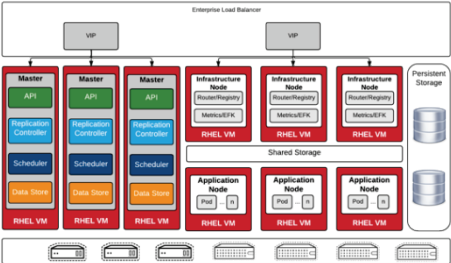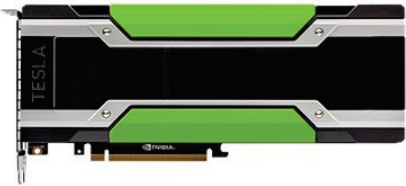
The DGX Station is one of the world's fastest workstations for artificial intelligence (AI) development (Nvidia). This workstation has been optimized for training data, which is a significant portion of the automated segmentation workflow. The DGX Station differentiates itself from the aforementioned computing platforms because of its hardware architecture. Embedded within the DGX Station are four 32 GB Tesla V100 GPUs (Nvidia DGX).[†] These GPUs combine to deliver an overall

---

[*] https://www.erdc.usace.army.mil/Media/Fact-Sheets/Fact-Sheet-Article-View/Article/712948/army-geospatial-enterprise-age-node/

[†] https://www.nvidia.com/en-us/data-center/dgx-station/

128 GBs in GPU memory. Applications can be processed in parallel since the DGX Station has four GPUs. This means that processes are evenly divided across all four GPUs instead of assigning all processes to one GPU. Implementing parallelization within a workflow is not intuitive. Additional programming is needed in order to ensure that all four GPU's communicate constantly and all workloads are assigned appropriately. The automated segmentation workflow was performed on a Nvidia DGX and system specifications are summarized in Table 1.

Table 1. System specifics categorically highlighting each computing platform used for performing the image segmentation along with associated characteristics.

Computing Platform	Category	Component	Model	
HP OMEN	Hardware	CPU RAM GPU GPU RAM Hard Disks	Intel Core i7-7700HQ (4 Cores) 32GB DDR4-2800 SDRAM GTX 1070 (8GB Total System) Data: 1TB 7200rpm SATA 512 GB PCle NVMe M.2 SSD	
	Software	OS Architecture	Windows 10 X64	
Army Geospatial Enterprise (AGE) Node	Hardware	CPU RAM GPU GPU RAM Hard Disks NIC	Intel Xeon E5-2630v4 @2.10 GHz 132 GB Nvidia Tesla P100-12Q 12 GB LSI Logic Parallel, 1: 100 GB, 2: 200 GB VMXNET	
	Host	Cores Model Hypervisor	4 cores, 2 virtual sockets Dell PowerEdge R730 VMWare ESXi 6.5.0 8294253	
	Virtual Machine	Hardware Version	ESXi 6.0 and later	
	Software	OS Kernel Architecture Virtualization	CentOS Linux 7 (Core) Linux 3.10.0-693.21.1. e17.x86_64 X86_64 VMWare	Single GPU example

Computing Platform	Category	Component	Model	
Nvidia DGX	Hardware	CPU	Intel Xeon E5-2698v4 @2.2 GHz (20-Core)	
		RAM	256 GB RDIMM DDR4	
		GPU	4x – Nvidia Tesla V100	
		GPU RAM	128 GB (Total System)	
		Hard Disks	Data: 3x – 1.92 TB SSD RAID 0	
			OS: 1x – 1.92 TB SSD	
		NIC	Dual 10GBASE-T (RJ45	
	Software	OS	Ubuntu Desktop Linux 16.04	
		Kernel	4.4.0-141-generic #167-ubuntu	
		Architecture	X86_64	

3   Methods

This document presents a workflow that leverages the use of CNNs and high-performance GPU processing to automate the pixel-wise segmentation of UAS imagery for faster image processing. Specifically, different computing platforms (HP Omen, AGE Node, Nvidia DGX) are compared to investigate each platform's advantages and limitations for tasks that often require high computing resources. Transfer learning is applied from an off the shelf, pre-trained CNN for model initialization. Accordingly, this reduces model development time and the extensive need for model training. Each image is individually segmented into seven independent classes resulting in a final segmented product. All steps necessary for acquiring aerial imagery, creating training and validation data sets, and segmenting imagery are described below.

3.1   Training data creation

Reliable training data is a foundational component in machine learning processes as these data are used by the algorithm to train and learn underlying trends. However, for this project, creating training data is labor intensive as it requires each individual pixel in an image to be manually labelled into seven predetermined classes. These classes are as follows: low vegetation, high vegetation, buildings, road surfaces, vehicles, water, and clutter. For this assessment, clutter consists of all other features present in the images that fall outside of the other six classes. The International Society for Photogrammetry and Remote Sensing (ISPRS) guidelines were used as a reference to distinguish between these classes. The process of manually labeling each pixel is known as digitizing, where a trained user traces information from an image and assigns a label associated with the predefined classes. Once the training data are assigned a label they are converted to a single standardized value. This step allows pixel values that are often unsigned integers between 0 and 255 to be represented directly, as numbers to the neural network scaled to pixel values in an array ranging from 0 to 6 standardizing the values as shown in Figure 4. Thus, each value corresponds to a class as follows: low vegetation - 0, high vegetation -1, buildings –2, roads –3, vehicles -4, water -5, clutter -6. An open source image manipulation software (Gnu Image Manipulation Program, Version 2.8) was used to manually create the training data.[*]

---

[*] https://www.gimp.org/

Figure 4. Sample image converted to training labels and represented to integers.

Low Vegetation   Building   Roads   Vehicle

0   2   3   4
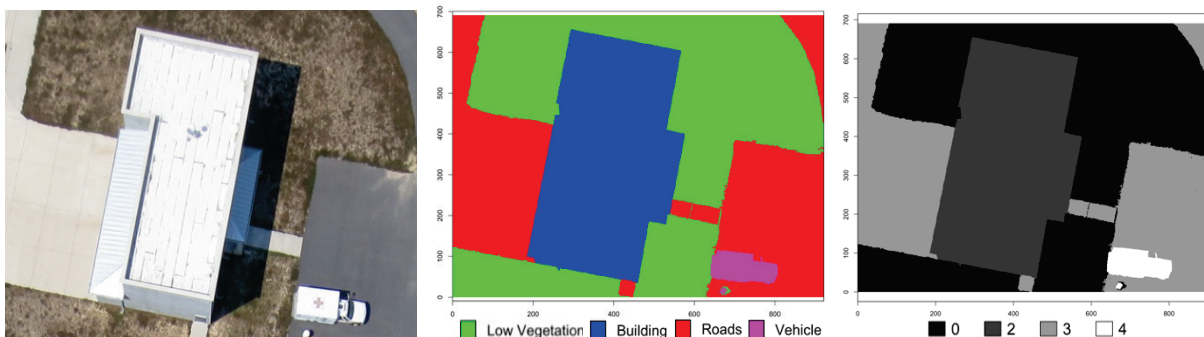
3.2   Image splitting

The dimensions of the original images collected from the eBee are 4000 X 3000 pixels. Since these dimensions are too large for the vGPU on the AGE node to process individually, all images in the dataset are split into tiles with a uniform size of 512 x 512 pixels for processing. This step is shown in Figure 6. The original eBee image is represented by Figure 5(a). This image is then split into equal tiles represented by a red grid as shown. The output results into 100 individual tiles as shown by Figure 5 (b and c). A 50% overlap is used when splitting the original image as this percentage has shown promising orthorectification results when images are mosaicked back together using a custom python script. Tiling images with no overlap often result in strong boundary artifacts created after each image is processed individually. This step was carried out throughout each computing platform for a uniform comparison across all systems. However, the Nvidia DGX station has 32 gigabytes (GB) of random-access memory (RAM) per GPU, and the images do not need to be tiled to execute the same workflow. After each original image is split into tiles, 80% of the data are divided into a training set and the remaining 20% are used for a validation set. The validation set is used to evaluate the model's performance and minimize bias in the algorithm.

Figure 5. Representation of image splitting used to crop the eBee data into multiple tiles for processing.
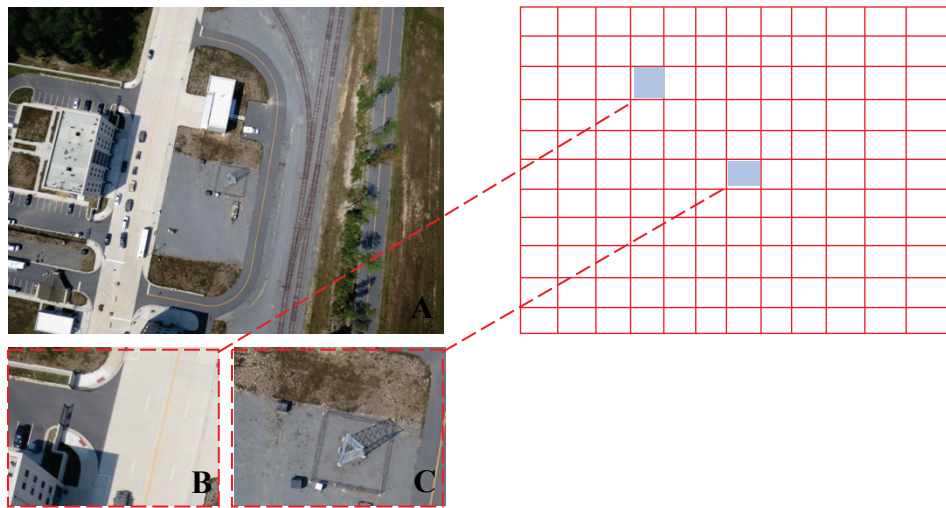
3.3   Wavelet transformation

Wavelet transformation is a mathematical function and signal processing technique that helps localize data by decomposing a signal into different frequency sub-bands. The wavelet transformation technique can be applied to a variety of fundamental signal processing tasks such as enhancing images and sounds recordings by removing noise or for data reduction by compression (e.g. jpeg compression) (Baaziz et al. 2010). When applying a wavelet transformation, data are compressed and transformed based on multi-resolution analysis (Huang and Aviyente 2008). This consists of the decomposition of an image into subimages containing different fractions of signal value which are known as wavelet coefficients, as shown in Figure 6. This preserves the relative distance between objects at different levels and allows natural clusters in the data to be easily identified. In practice, the transformation divides the data equally at each iteration (Han et al. 2006). First, it applies a data smoothing technique such as sum or weighted average. Next, it performs a weighted difference, which allows the details of features to be identified. The output results in a smoothed or low frequency version of the original data by removing the high-frequency information. This procedure occurs recursively by applying the transformation on the image resulting from the previous iteration and continues until a termination condition is met (Arivazhagan and Ganesan 2003; Amolins et al. 2007).

Figure 6. Sample image representing the wavelet decomposition process. The original image is shown in the upper left quadrant. The other three quadrants represent the original image being transformed after a horizontal, vertical, and diagonal filter are applied.
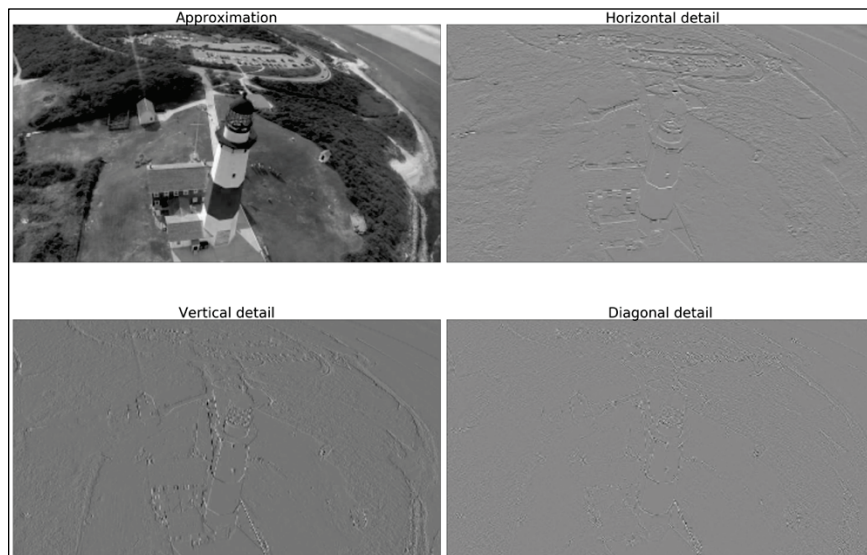
For this project, the function was built to serve as a pre-processing step and to create multi band images outside the original 3-bands (RGB). Thus, the dimensionality of the image would entail RGB along with the computed wavelet sub-images as additional bands with total bands of six. This pre-processing step would introduce spatial and textural information to the CNN along with the color information for model training. However, due to time constraints, the assessment of the model training and testing with the wavelet decomposition is not incorporated for this analysis.

3.4 Parallelization

Parallelization is the process of designing a program or system to perform a given task in parallel. Implementing an automated image segmentation workflow on a platform with a single-GPU differs greatly from implementing the same process on a system that has multiple GPUs available. Single-GPU and multi-GPU platforms have strict software dependencies. GPU hardware determines the dependencies needed within each platform. Guidelines provided by Nvidia were followed in order to seamlessly integrate libraries and toolkits with the deep learning framework, Tensorflow-GPU, needed to perform the image segmentation. The Nvidia Compute Unified Device Architecture (CUDA) toolkit was used with the deep neural network library (cuDNN). This enabled libraries, debugging, and optimization tools needed for GPU recognition and

accelerated computing with Python. The Nvidia CUDA toolkit and cuDNN are freely available for download though the Nvidia website.*

3.4.1 Single GPU versus Multi-GPU

System configuration for single-GPU acceleration has fewer scripting and dependency requirements as opposed to a multi-GPU set-up. For single GPU initiation, the working directory requires all modules listed in Table 2 to be available. Due to traditional methods for deep learning tasks being performed on CPUs, model configurations remain agnostic to the hardware available in the system and require the code to be altered in order to distinguish and utilize the GPUs from CPUs. In order to test if the GPUs are being recognized, and therefore used by Tensorflow and NVIDIA cuDNN, the following command is used: "tf.test.is gpu available" (Tensorflow). If there is only one GPU on the platform, the result is "/GPU:0." Because data initialization and parameter settings should be performed on the CPU, specifying GPU use only needs to be performed directly prior to when training starts using the command "with tf.device('/GPU:0')."

Table 2. Summary of libraries used for implementing the VGG-16 model on single and multiple GPUs.

	Single GPU		Multi-GPU	
	python 3.6	nose	python 3.6	nose
	numpy mat	mpi4py	numpy mat	mpi4py
	plotlib	sphinx	plotlib	sphinx
	scipy	m2w64	scipy	m2w64
Libraries	scikit-learn	tensorflow	scikit-learn	tensorflow
	pillow	keras	pillow	keras
	libpython		libpython	horovod
	pygpu		pygpu	MPI
	theano		theano	
	libgpuarray		libgpuarray	

The Nvidia DGX system is used as the operating platform to test single GPU vs multi-GPU implementation. In contrast to single GPUs that process natively on the platform, multi-GPU is a more complex workflow because it uses GPU-enabled Nvidia Docker containers. The Nvidia Docker containers package applicable dependencies needed to run an application

---

* https://docs.nvidia.com/cuda/cuda-installation-guide-microsoft-windows/index.html

utilizing onboard GPUs (NvidiaDocker). The Nvidia GPU Cloud hosts predefined, application specific, containers that a user can import into a working directory and use. The Nvidia container used for this work is Tensorflow 1.8.0 py3. Once the container is pulled, it is then saved with an image identification number (ID) for example, 8289b0a3b285.

The Tensorflow 1.8.0 py3 container containins all the required libraries to execute the VGG-16 model. In addition to the Docker containers, two extra libraries are required in order to run the VGG-16 model in parallel. These libraries are Horovod and message passing interface (MPI) listed in Table 2. Horovod is an open source distributed (parallelization) training framework for deep learning tasks – Github (Sergeev and Del Balso 2018). Seamless to the user, Horovod handles variable updates, data management, and appends tasks to GPUs. This library simplifies the transition to multi-GPU usage and enables the code to scale to the number of GPUs available.

MPI is the second library needed to efficiently leverage GPUs. It is an internal process communication interface that distributes data in memory and efficiently represents or "communicates" the data across processors. Essentially, it enables the user to execute a program in parallel without compiling and running the process sequentially across processors (Abernathey and Key 2017).* This library is widely used from large-scale supercomputers to cloud computing infrastructures and single nodes on systems. The Horovod and MPI libraries also work together by using the platforms' CPU to gather information from each individual GPU at each training iteration. The CPU then redistributes updated information to each GPU for the next training iteration

Although these systems have similar software components, highlighted in Table 1, each platform requires an independent configuration to perform the image segmentation workflow due to the hardware differences. For example, systems like the HP Omen and AGE Node are not able to handle UAS imagery at their full resolution, thus each image had to be split into several small ones. On the contrary, the DGX system, which is naturally designed for large data processing tasks, can handle the full resolution imagery.

---

* https://rabernat.github.io/research_computing/parallel-programming-with-mpi-for-python.html

4 Results and Discussion

4.1 Model training and image segmentation

The VGG-16 algorithm was initially pre-trained on a large benchmark dataset to distinguish general classes. This provided the model with "pre-packaged" weights representing the importance between the input values at each node. These initial weights were used to fine-tune the deeper layers of the model's network as the model was trained on our dataset (Fort AP Hill, Camp Cook, and Fort Campbell) for specific classes of interest. The training set comprised of 1,000 image tiles and was constructed as discussed in Section 3.1. Additional model optimization parameters that influenced the model training are discussed below.

The ReLU activation function is used to evaluate and shape the training of the network. This function results in sparse and efficient activations by computing true zero values, making it less computationally expensive for complex networks. Thus, the function does not saturate for large inputs and has been shown to surpass human-level performance (Wang and Xi 2012; Nvidia Corporation 2015). This makes it particularly effective for use with the VGG-16 network, which consists of a number of convolutional and max pooling layers with a fully connected layer at the end of the network.

The cross-entropy loss function is chosen to estimate error in the neural network by calculating the difference between the true and the estimated distribution. This function is represented by the equation below:

$$H(p_i, q_i) = -\sum_i p_i \ \log(qi + \varepsilon) \tag{1}$$

where

$$\varepsilon << 1 = \text{small bias introduced for numerical stability}$$
$$p = \text{the true distribution}$$
$$q = \text{the estimated distribution.}$$

The cross-entropy function, in conjunction with a softmax layer, provides probabilities for each output class. Lastly, the dropout regularization technique and the Adam optimizer were added to the networks architecture to simplify the network and improve model generalization and robustness.

The trained CNN model is applied to classify all unlabeled pixels in the remaining images. The resulting segmentation of each image is a discrete output indicating various classes of heterogeneous features present within each image. Figure 7 represents the segmented images classified by the CNN in comparison to the original images and ground truth data (training labels).

Figure 7. Qualitative results of two-dimensional (2D) segmentation showing the original images on the left, ground truth data in the middle, and segmented images as classified by the CNN.
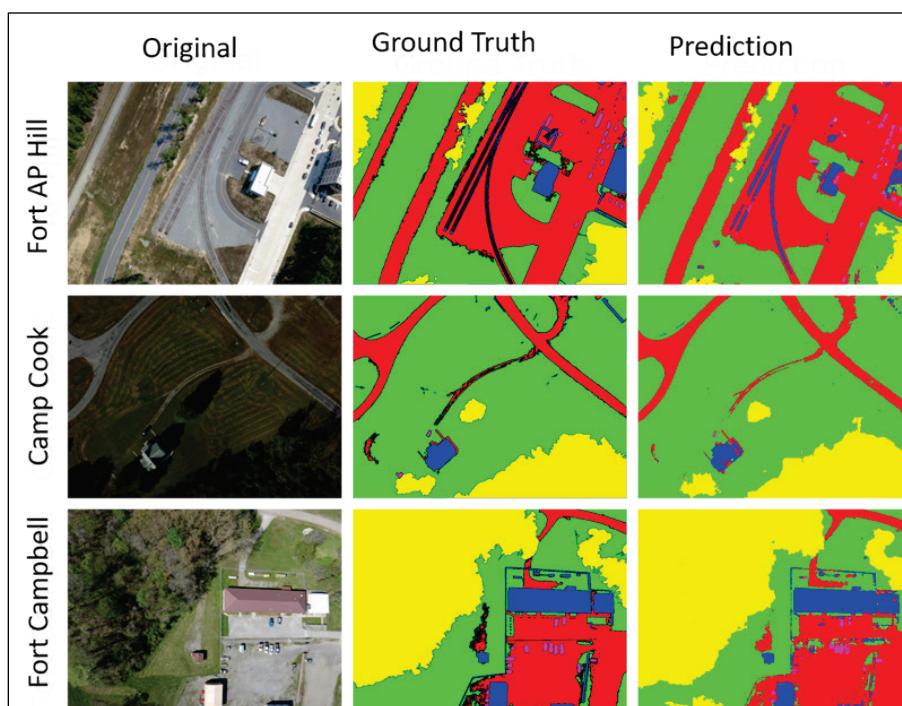
Figure 7 compares model output between study sites with different densities of urban terrain and vegetative cover. In comparison to the original images and the associated ground truth data, the predicted classes from the model are well preserved. This is true for large continuous features such as buildings, seen in Fort AP Hill, or low vegetated regions present at the Camp Cook site. This is also true for smaller features such as cars as seen in the Fort Campbell and Fort AP Hill images. It can be concluded that the model is not affected by the underrepresentation of pixels representing a class (e.g. cars) in the overall pixel count of an image. Additionally, it can be inferred that the model is resistant to overfitting the dominant classes. If class overfitting occurred, then low vegetated areas (e.g. grass) that have similar color values as high vegetated regions (trees) would be predicted as one class. Whereas in Figure 7, they are predicted as

separate classes highlighted by a green color (low vegetation) and yellow (high vegetation). Lastly, it can be seen that the model can distinguish between in-class variations with similar spectral values such as the road and building in Fort AP Hill, yet ignores variations in spectral values cause by shadows as seen in Camp Cook.

4.1.1 Evaluation metrics

Since no single measurement gauges every aspect of system performance, a number of different evaluation metrics were used to assess the accuracy of the CNN model. These metrics are accuracy, precision, recall, and F-1 score. Accuracy refers to the portion of correct classifications calculated by counting the number of times the classifier correctly predicts a class and dividing by the total number of overall predictions (Kernell 2018).

The loss function can be used to estimate the model's training performance and complement the overall accuracy metric due to the inverse relationship between the two. For instance, the higher the loss function is during training, the lower the model accuracy is, and the lower the loss function is during training, the higher the overall accuracy metric is.

Caution, however, should be used when looking at overall model accuracy because it can be inaccurate for datasets that have a skewed distribution and uneven class representation leading to high class imbalance (Kernell 2018).

$$Accuracy = \frac{Total \; \# \; of \; correct \; predictions}{Total \; \# \; of \; predictions}$$

Precision is obtained by dividing the number of true positives by the total number predicted positives. Recall is calculated by dividing the number of true positives by the number of actual positives (i.e., the sum of all true positives and false negatives). Therefore, precision is the metric of choice when the cost of false positives is high, and recall is the metric of choice when the cost of false negatives is high (He et al. 2015).

$$Precision = \frac{True \; positives}{Total \; \# \; of \; positive \; predictions} \quad Recall = \frac{True \; Positives}{Total \; \# \; of \; true \; positives}$$

The F-1 score is a function of precision and recall and reflects a balance between the two metrics. For example, a high F-1 score indicates low false positives and low false negatives. In other words, a perfect model would

have a score of 1, while a completely failed model would have a score of 0 (He et al. 2015).

$$F1 = 2\left(\frac{Precision * Recall}{Precision + Recall}\right)$$

The model's precision, recall, and mean F-1 score are calculated across GPU enabled platforms to assess how system characteristics affect the models' performance. For this assessment, the AGE Node (single virtual GPU) was used in comparison with a single GPU from the multi-GPU DGX system. Table 3 highlights the results after the accuracy calculations are derived. It can be noted that overall model precision, recall, and F-1 scores are similar and not significantly affected by the differences in systems. An F-1 score of 0.82 suggests that overall, the CNN algorithm performs relatively well at accurately predicting pixels to the correct associated classes. This is further confirmed by the precision and recall values, which are close in value. This similarity implies that there is an overall balance between correct and incorrect predictions by the algorithm.

Table 3. Comparison of precision, recall and F-1 scores for the AGE Node vGPU and a single GPU on the DGX Station.

	Precision	Recall	Mean F-1 Score	Batch Size (# of image tiles)	Processing Speed (image tiles/second)
AGE Node vGPU	0.88	0.80	0.82	5	126
DGX Station	0.87	0.81	0.83	128	459

The most noteworthy difference between the systems is the batch size and processing speed used to execute the CNN. Batch size refers to the number of images provided to the model at a time. For example, the AGE node (vGPU) only has access to a portion of the 12 GB of GPU RAM resident on the host system. For this reason, images had to be split into tiles and run in batch sizes of 5 tiles. This yields a total processing speed of 126 image tiles per second as shown in Table 3. In essence, defining the batch size allows the model to start training with the first set (batch) of images and iterate throughout the entire dataset accordingly until all samples are propagated through the nodes of the CNN.

In contrast to the AGE node, the workflow in the DGX occurs considerably faster. Because the DGX has 32 GB of GPU RAM on each video card, the batch size can be increased significantly. Table 3 lists the batch size and processing speeds on the DGX Station for a single GPU with a batch size of

128 images at a rate of 459 image tiles per second. Although it is intuitive that the system built for tasks that require large amounts of processing will perform better. This comparison shows that nearly identical results are obtained with the trade-off for processing time. Nevertheless, this processing time is critical for time sensitive military applications that require large data processing. Furthermore, as the problem is scaled, computational needs will scale accordingly.

Model training time provides an additional assessment used to evaluate the differences between the three computing platforms. This metric is selected because training CNN models is the most time-consuming stage of the workflow. However, this is also the most critical determinant to the overall model performance. For this experiment, 1,000 training images of a slightly larger tile size (921 x 691 pixels) were used to train the CNN across the HP Omen (single GPU), AGE Node (vGPU), and DGX station. The code was run in parallel in order to utilize the maximum potential of each system.

Similar to the previous experiment, the models achieved comparable accuracy across all platforms. However, parallelizing the CNN across four GPUs greatly reduced the overall model training time by an order of magnitude as shown by Table 4. Multi-core systems, such as the DGX station, which allow for large-scale, distributed model training can be leveraged by Army planning analysts needing on-the-fly, large-scale data processing capabilities.

Table 4. Comparison between single and multi-GPU computing platforms used for CNN model training and accuracy.

	HP Omen Single GPU	AGE Node Single vGPU	DGX Station Multi-GPU (4)
Batch Size	1	5	120 (30 per GPU)
Epoch	40	40	40
Training Time (hrs)	21.75	10	2.6
Accuracy (%)	83	82	84

Observing the models' loss function throughout training provides insight on overall model's performance. However, this performance may be influenced by a number of forward and backward passes, known as epochs, which the CNN conducts during the training stage. For instance, when 1,000 training images are used and the batch size is 500, then

2 iterations (1 forward and 1 backwards) are conducted to complete 1 epoch. The number of training samples also influence model performance by impacting model bias and variance. For these reasons it is important to understand where the balance is met between training size and epoch. Finding a balance between the number of epochs and training size is important to train the model efficiently and accurately. Training sizes that are too large may take too long to process and overload memory. Conversely, insufficient training sizes can potentially increase bias by not having enough data to train on in each epoch

Model learning curves are commonly used as a diagnostic tool to visualize machine learning algorithms performance over a period of time. Validation accuracy, validation loss, and training loss are good estimates used to identify potential model learning problems in the early stages of the workflow. These curves allow the end user to foresee if the model is overfitting or underfitting and provide insight if model adjustments are required. Learning curves for validation accuracy, validation loss, and training loss curves are shown in Figure 8 for the CNN for a series of epochs and iterations.

Comparing the validation and training loss respectively, both loss curves follow a similar pattern where both converge to a point of stability at approximately 40 epochs. This pattern between the two curves suggests that the model does not overfit or underfit while training. Instead it shows it has an overall good fit. Associating the validation curves with the training curve allows the end user to understand and evaluate the state of the CNN model at each stage of training. For example, this evaluation is performed while the model is training on the training dataset, as well as, during its validation stage when it is applied on a dataset that it has not seen before.

The validation accuracy curve, represented in blue, shows that the CNN converges at approximately 0.85 (85%) accuracy when the model is applied to the validation dataset. This level of accuracy is also highlighted in Table 4 when the model is applied across different computing platforms and GPUs.

Figure 8. Model learning curves for validation accuracy (blue line), validation loss (red line) and training loss (shown in pink) for the CNN for a series of epochs and iterations.
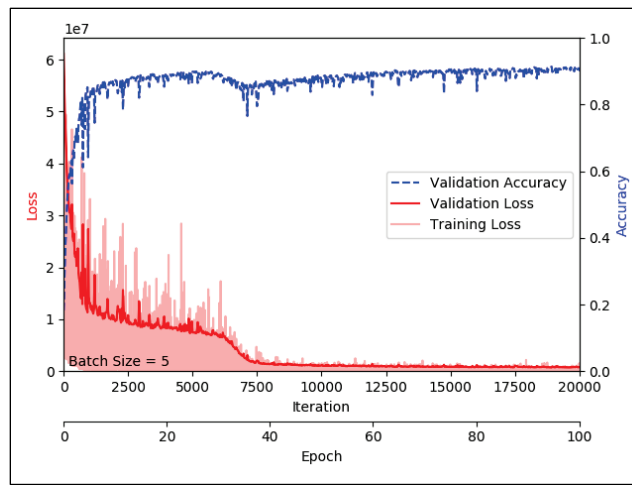
An additional graph is generated to compare how the accuracy of the model changes as more training data is introduced. This fosters an understanding of how the model is performing over the training sequence. It visualizes the minimum amount of data that the model needs to reach the best compromise between bias and variance, which in turn can minimize training time. Figure 9 depicts the model accuracy with respect to the amount of training data. After a certain point (approximately 18 images) the CNNs validation accuracy marginally increases as more training images are added. Additionally, the graph shows that the best accuracy of 0.92 is achieved with only 42 training images.

Figure 9. Comparison of model accuracy in relation to the number of training images used.
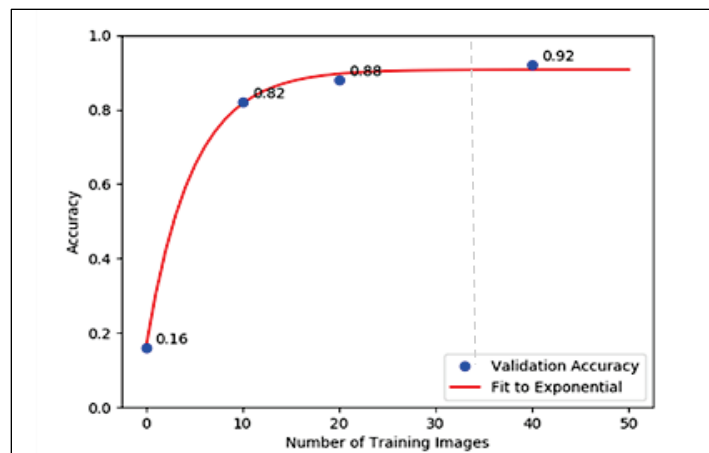
5 Conclusion

This document presents a workflow that leverages the use of CNNs and high-performance GPU processing to automate pixel-wise image segmentation of aerial imagery collected from UAS. Specifically, different computing platforms are compared to investigate each platforms advantages and limitations for tasks that often require high computing resources. Such tasks are particularly important as data becomes more widely available and as UASs are increasingly used to improve situational awareness. The ability to perform a quick and accurate analysis on aerial imagery becomes vital for agile decision making.

The results presented in the report indicate that CNNs are an effective method for image segmentation tasks and this method can be successfully applied in regions with various land cover types and complexities. This application is illustrated in Figure 7 where the classes predicted by the VGG-16 model corresponded well to the ground truth data with a precision accuracy of 88% (Table 3). The CNN model was able to accurately predict pixels that were continuous, signifying large features (e.g. buildings or sizable areas of low vegetation) as well as smaller features (e.g. cars and railroads). This shows the model is resistant to overfitting in the dominant classes and can distinguish which class a pixel belongs to regardless of neighboring pixels with similar characteristics.

The model's training performance was evaluated using a sequence of epochs and training sample sizes. The analysis suggested that the CNN algorithm converges after 40 epochs with approximately 85% accuracy. When a larger number of epochs were tested, the model's accuracy slightly increased. This suggests that as the number of epochs increases, the model weights adjust in the neural network and causes the model to begin to overfit.

When the VGG-16 model is coupled with GPU-accelerated processing and parallelized across multiple GPUs, the model training time is significantly reduced while preserving the same accuracy as summarized in Table 4. Although the workflow between the systems varies slightly in the number of epochs and batch size, approximately the same accuracy is retained across all three systems (HP Omen, AGE Node, DGX). This signifies that GPU memory and core count available within a system are critical components in terms of processing speed and preprocessing steps. For

example, due to the 128 GB of memory available in the DGX, the system is able to process large amounts of data with no preprocessing required. Whereas, with the HP Omen and AGE Node, which have single GPUs, images must be split into small tiles and processed in batches.

The VGG-16 model was initially chosen due to its simple architecture and literature support that it was more accurate and less prone to over-fitting than other algorithms such as the InceptionV3 network. Limitations of the VGG-16 model derive from the large number of parameters and weights, which require expert input to fine tune the model. Additionally, specific tasks may require many convolutions which could decrease the model's efficiency. Future work could investigate the application of the model to datasets with supplementary layers. For example, expanding the imagery to include textural information derived from wavelet decomposition. This step allows the end user to introduce additional layers and identify optimal combination to reduce workload.

Fully convolutional networks represent a rich class of models capable of performing a wide range of tasks such facial recognition, scene labeling, and image segmentation. Reliable training data is a critical component when working with such models as it is used to fit the algorithm. This research created a unique training dataset. This dataset can be leveraged for future segmentation efforts and serve as a baseline to benchmark future CNN and efficiently support critical image processing tasks for the Military.

References

Amolins, K., Y. Zhang, and P. Dare, P. 2007. Wavelet based image fusion techniques—An introduction, review and comparison. *ISPRS Journal of photogrammetry and Remote Sensing*, *62*(4):249-263.

Arivazhagan, S., and L. Ganesan. 2003. Texture classification using wavelet transform. *Pattern recognition letters*, *24*(9-10):1513-1521.

Baaziz, N., O. Abahmane, and R. Missaoui. 2010. Texture feature extraction in the spatial-frequency domain for content-based image retrieval. *arXiv preprint* arXiv:10125208

Becker, C., N. Hani, E. Rosinskaya, E. d'Angelo, and C. Strecha. 2017. Classification of aerial photogrammetric 3D point clouds. *Photographic Imaging and Remote Sensing* 84(5):297–295. https://doi.org/10.14358/PERS.84.5.287.

Castelluccio, M., G. Poggi, C. Sansone, and L. Verdoliva. 2015. Land use classification in remote sensing images by convolutional neural networks. *arXiv preprint arXiv:1508.00092.*

Chen, K., K. Fu, M. Yan, X. Gao, X. Sun, and X. Wei. 2018. Semantic segmentation of aerial images with shuffling convolutional networks. In *IEEE Geoscience and Remote Sensing Letters* 15(2):173–177. doi: 10.1109/LGRS.2017.2778181.

Cordts, M., M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. 2016. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* 3213–3223.

Garcia-Garcia, A., S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez. 2017. A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857v1, 2017.*

Goodfellow, I., Y. Bengio, and A. Courville. 2016. *Deep Learning*. Cambridge, MA: MIT Press.

Han, J., M. Kamber, and D. Mining. 2006. Southeast Asia Edition: Concepts and Techniques.

He, K., X. Zhang, S. Ren, and J. Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification.

Huang, K., and S. Aviyente. 2008. Wavelet feature selection for image classification. *IEEE Transactions on Image Processing*, *17*(9):1709-1720.

Hu, X., and Y. Yuan. 2016. Deep-learning-based classification for dtm extraction from als point cloud. *Remote Sensing* 8(9):730.

Hu, F., G. Xia, J. Hu, and L. Zhang. 2015. Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery. *Remote Sensing* 7(11):14680–14707.

Karantzalos, K. 2015. Recent Advances on 2D and 3D Change Detection in Urban Environments from Remote Sensing Data. In *Linked Activity Spaces: Embedding Social Networks in Urban Space* Springer pp. 237 - 272.

Karpathy, A. 2018. CS231n - Convolutional Neural Networks for Visual Recognition. Stanford University. (Accessed 11 March 2019) http://cs231n.github.io/.

Kernell, B. 2018. *Improving Photogrammetry Using Semantic Segmentation*. Advanced Level Thesis. Linkoping, Sweden: Linkoping University.

Kingma, D., and J. Ba. 2017. Adam: A method for stochastic optimization. *arXiv preprint arXiv: 1412.6980v9.*

Kyrkou, C., G. Plastiras, T. Theocharides, S. Venieris, and C. Bouganis. 2018. Dronet: Efficient convolutional neural network detector for real-time UAV applications. In *2018 Design, Automation and Test in Europe Conference and Exhibition*. doi: 10.23919/DATE.2018.8342149.

Long, J., E. Shelhamer, and T. Darrell. 2014. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 3431–3440.

Marmanis, D., J. D. Wegner, S. Galliani, K. Schindler, M. Datcu, and U. Stilla. 2016. Semantic segmentation of aerial images with an ensemble of CNNs. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *3*, 473.

Matikainen, L., and K. Karila. 2011. Segment-based land cover mapping of a suburban area - Comparison of high-resolution remotely sensed datasets using classification treees and test field points *Remote Sensing* 3(8):1777–1804. https://doi.org/10.3390/rs3081777.

Nvidia Corporation. 2015. GRID Virtual GPU User Guide, DU-06920-001. Santa Clara, CA. (Accessed on 20 June 2019). http://us.download.Nvidia.com/Windows/Quadro_Certified/GRID/354.80/ESXi-6.0/352.83-354.80-Nvidia-grid-vgpu-user-guide.pdf.

Parashar, A., M. Rhu, A. Mukkara, A. Puglielli, R. Venkatesan, B. Khailany, J. Emer, S. W. Keckler, and W. J. Dally. 2017. SCNN: an accelerator for compressed-sparse convolutional neural networks. In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA).* Doi: 10.1145/3079856.3080254.

Ruder, S. 2017. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv: 1609.04747v2.*

Sergeev, A., and M. Del Balso. 2018. *Horovod: Fast and easy distributed deep learning in TensorFlow.*

Simonyan, K., and A. Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. *arXiv: 1409.1556v6.*

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958.

Tang, Y., and L. Zhang. 2017. Urban change analysis with multi-sensor multispectral imagery. *Remote Sensing* 9(3):252. https://doi.org/10.3390/rs9030252.

Trimble Geospatial. nd. eCognition 9, Trimble. (Accessed 20 June 2019) http://www.ecognition.com/.

You, J., X. Li, M. Low, D. Lobell, and S. Ermon. 2017. Deep gaussian process for crop yield prediction based on remote sensing data. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence* 4559–4566.

Volpi, M., and D. Tuia. 2017. Dense semantic labeling of subdecimeter resolution images with convolutional neural networks. In *IEEE Transactions o Geoscience and Remote Sensing* 55(2):881–893.

Wang, C., and Y. Xi. 2012. Convoluional neural network for image classification., Baltimore, MD: Johns Hopkins University.

Wang, H., Y. Wang, Q. Zhang, S. Xiang, and C. Pan. 2017. Gated convolutional neural network for semantic segmentation in high resolution images. *Remote Sensing,* 9(5):446. https://doi.org/10.3390/rs9050446.

Unit Conversion Factors

Multiply	By	To Obtain
acres	4,046.873	square meters
cubic feet	0.02831685	cubic meters
cubic inches	1.6387064 E-05	cubic meters
cubic yards	0.7645549	cubic meters
feet	0.3048	meters
hectares	1.0 E+04	square meters
inches	0.0254	meters
microns	1.0 E-06	meters
miles (nautical)	1,852	meters
miles (U.S. statute)	1,609.347	meters
miles per hour	0.44704	meters per second
mils	0.0254	millimeters
pounds (mass)	0.45359237	kilograms
square feet	0.09290304	square meters
square inches	6.4516 E-04	square meters
square miles	2.589998 E+06	square meters
square yards	0.8361274	square meters
tons (2,000 pounds, mass)	907.1847	kilograms
tons (2,000 pounds, mass) per square foot	9,764.856	kilograms per square meter
yards	0.9144	meters

Acronyms and Abbreviations

AI	Artificial intelligence
AGE	Army Geospatial Enterprise
ANN	Artificial Neural Network
BLC	Basic Leader Course
CNN	Convolutional Neural Network
COTS	Commercial-Off-The-Shelf
CUDA	Compute Unified Device Architecture
cuDNN	deep neural network library
ERDC	Engineer Research and Development Center
EPEL	Enterprise Linux
ExIF	Exchangeable Image File Format
GEOINT	Geospatial Intelligence
GeoICUE	Geo-Intelligence for Complex Urban Environments
GB	Gigabytes
GRL	Geospatial Research Laboratory
HPC	High Performance Computing
IC	Intelligence Community
ISPRS	International Society for Photogrammetry and Remote Sensing
JPEG	Joint Photographic Experts Group
MPI	message passing interface
NCOA	Noncommissioned Officers Academy
RAM	random access memory
RGB	red, green, blue
ReLU	Rectified Linear Unit
SOAR	Special Operations Aviation Regiment
TANH	hyperbolic tangent
UAS	Unmanned Aerial Systems
USACE	U.S. Army Corps of Engineers
VGG	Visual Geometry Group
vGPU	virtual graphical processing unit

VM	virtual machine
2-D	two-dimensional
3-D	three-dimensional

REPORT DOCUMENTATION PAGE

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
March 2020	Final report	

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Use of Convolutional Neural Networks for Semantic Image Segmentation Across Different Computing Systems	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER 62784

6. AUTHOR(S)	
Andmorgan R. Fisher, Timothy A. Middleton, Jonathan Cotugno, Elena Sava, Laura Clemente-Harding, Joseph Berger, Allistar Smith, and Teresa C. Li	5d. PROJECT NUMBER 855
	5e. TASK NUMBER 23
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Geospatial Research Laboratory U.S. Army Engineer Research and Development Center 7701 Telegraph Road Alexandria, VA 22315-3864	ERDC/GRL TR-20-7

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
Headquarters, U.S. Army Corps of Engineers Washington, DC 20314-1000	
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The advent of powerful computing platforms coupled with deep learning architectures have resulted in novel approaches to tackle many traditional computer vision problems in order to automate the interpretation of large and complex geospatial data. Such tasks are particularly important as data are widely available and UAS are increasingly being used.

This document presents a workflow that leverages the use of CNNs and GPUs to automate pixel-wise segmentation of UAS imagery for faster image processing. GPU-based computing and parallelization is explored on multi-core GPUs to reduce development time, mitigate the need for extensive model training, and facilitate exploitation of mission critical information. VGG-16 model training times are compared among different systems (single, virtual, multi-GPUs) to investigate each platform's capabilities.

CNN results show a precision accuracy of 88% when applied to ground truth data. Coupling the VGG-16 model with GPU-accelerated processing and parallelizing across multiple GPUs decreases model training time while preserving accuracy. This signifies that GPU memory and cores available within a system are critical components in terms of preprocessing and processing speed. This workflow can be leveraged for future segmentation efforts, serve as a baseline to benchmark future CNN, and efficiently support critical image processing tasks for the Military.

15. SUBJECT TERMS		
Aerial photography Drone aircraft Geospatial data – Computer processing	Remote-sensing images Image processing Neural networks (Computer science)	Computer vision

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER (include area code)
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR	42	