Autonomous runway damage detection through deep learning

John S. Harchanko*, Michael Wellfare, and David Forester of
Polaris Sensor Technologies, 200 Westside Square, Huntsville, AL, USA 35801

ABSTRACT

Detection and classification of damage and debris on runways is important to ensure safe operation of aircraft. Manual or human-in-the-loop monitoring of such large areas can be difficult and unsafe. Autonomous operation is possible by regular overflights of Small Unmanned Aerial Systems (SUAS) as the basis for change detection algorithms augmented by deep learning. A series of tests were conducted at Tyndall AFB, Florida in which an undisturbed runway was imaged from an SUAS. Subsequent post-damage flights imaged craters and debris. We present a change detection approach coupled with Convolutional Neural Networks (CNNs) for classification of the damage. Once trained and validated, the network successfully classified the image patches as one of four types of objects (crater, camouflet, debris, or null) with an accuracy of 98.5%. Furthermore, the location of the objects was predicted within 10-16% and the radius of the debris fields was predicted to better than 7%.

Keywords: change detection, neural networks, artificial intelligence, CNN, UXO

1. INTRODUCTION

As the United States Air Force (USAF) moves towards expeditionary environments, where airbases are increasingly exposed to enemy attacks, it becomes more important to be able to respond quickly to attacks that damage the airfield by restoring the airfield to its minimum operational state. A Rapid Airfield Damage Assessment System (RADAS) to automatically locate, classify, and measure runway damage is highly desirable because it will reduce the timeline required to assess airfield damage and restore airfield operations. RADAS would also reduce risk to personnel from unexploded ordnance by providing a stand-off capability for assessment. Currently this assessment is performed manually.

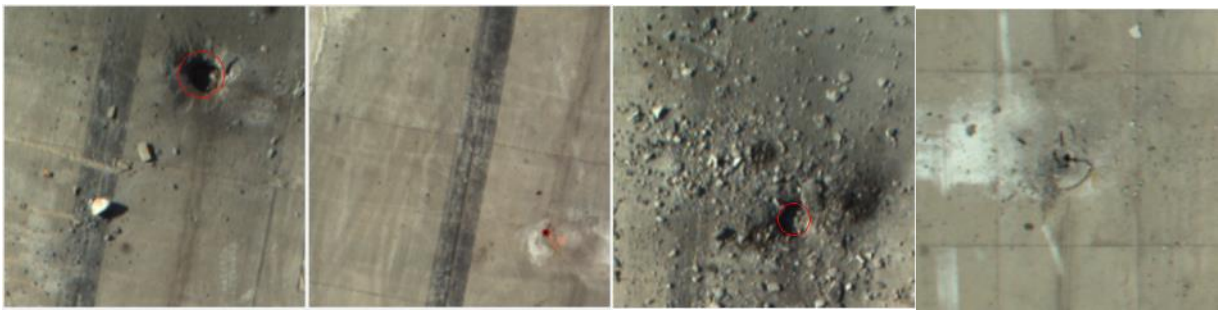

Figure 1 From left to right, a crater with large chunks of debris, a camouflet, three craters surrounded by debris, and a camouflet with large chunks of debris from a nearby crater.

Due to the questionable usability of the possibly damaged paved surfaces, operations using Small Unmanned Aerial Systems (SUAS) have the advantage of getting to the damage site and providing a good perspective for damage assessment sensors. The main limitation of SUAS technology is the maximum payload combined with the amount of runway (time aloft) and data bandwidth (computational load). We have been working to demonstrate how the optimal combination of sensors and algorithms can address these challenges. Some sample images of runway damage are shown in **Figure 1**.

Polaris Sensor Technologies, Inc., is widely recognized as an industry leader in the design, development, building, and test of polarimetric imaging sensors. Imaging polarimetric sensors require a broad skillset including optics, mechanics, electronics, image processing, and some system engineering. This skillset is combined with industry

partners and team members to provide a suite of sensors and associated software that is purpose built for an SUAS. The next two sections provide some background.

1.1 Damage Detection System Approach

Polaris Sensor Technologies, Inc., developed a system approach for runway damage detection that utilizes two SUAS: a "Scout" and a "Surveyor", see **Figure 2**. The Scout SUAS mission is to cover as much ground as quickly as possible while locating the damage sites. The Scout imagery is radioed to a "Home Base" computer that indexes the geo-location of each image with a baseline image from the same geo-location. The baseline imagery contains no damage at that geo-location. Image patches of the areas of change are passed to a neural network that provides preliminary damage classification of craters, camouflets (the enemy's munition sometimes creates large cavity under the runway marked only by a small hole in the runway surface), debris (from craters and may also contain UXO), or null (a class created for cases in which the change area does not contain anything of interest). In the case of craters and camouflets, the software also measures the location and diameter.

Figure 2. Research and development Scout SUAS

Based on these findings, a Surveyor SUAS may be dispatched to the area while the Scout continues its search path. The Surveyor flies much lower than the Scout and carry a payload different from but complementary to the Scout's sensor payload. The Surveyor mission is to confirm the findings of the Scout (geo-location and metrology of crater, camouflet, and debris) and to provide additional information such as UXO class (A-F).

Home Base provides the interface to a common database. Other systems and personnel monitor that database so that informed decisions can be made regarding the order of repair operations. Using the Scout, Surveyor, Home Base approach requires command, control, and deconfliction of multiple SUAS. In exchange for some complexity, the RADAS mission is accomplished in the quickest possible time with the best possible accuracy.

1.2 General Algorithm Approach

Our algorithm, outlined in **Figure 3**, starts with change detection between some baseline image and a current image. To detect the change, we register the two images and identify which areas or patches likely contain change. Those patches are then sent to a neural network which classifies the image patch and provides the location and size of the damage (in pixels). This damage location and size are then processed with the SUAS meta-data to provide location and size in real-world units.

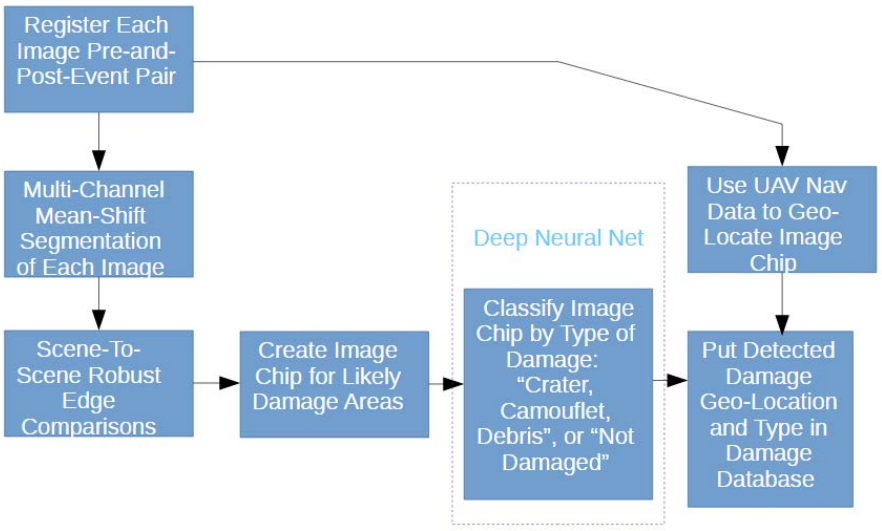

Figure 3 We combine change detection, a neural network, and geo-location to locate, classify, and measure runway damage.

2 DATA COLLECTION

The system for data collection, research, and development is shown in **Figure 2**. While the airframe is a DJI S900, the DJI flight controller has been replaced with a PixHawk and a custom stabilized gimbal has been mounted under the airframe which provides a stable platform for a machine vision camera. An onboard SolidPC provides the camera and iris control and also feeds waypoints to the PixHawk. An automated algorithm running on the SolidPC optimizes the lens iris and the camera's integration time. The SUAS mission parameters are chosen to manage the data bandwidth within the downlink speeds of 30 Mbps maximum.

A data collection at Silver Flag, Tyndall Air Force Base, was conducted on August 2, 2016. The setup is shown in **Figure 4**. The SUAS Command and Control was located generally within the space marked "C/C". The SUAS system took off and landed within the space marked "VTOL". A white and a black calibration target where located just to the North of the VTOL area so that they were included in the collected imagery and could be utilized for post-collect camera calibration and inclusion in the change-detection algorithm. The areas that are numbered roughly correspond to the sections of runway that were analyzed. The extent of each numbered section roughly corresponds to the field of view of the sensor. The sections are slightly off center of the runway but are centered on the damaged areas to ensure that the entire blast was included in the camera's field of view.

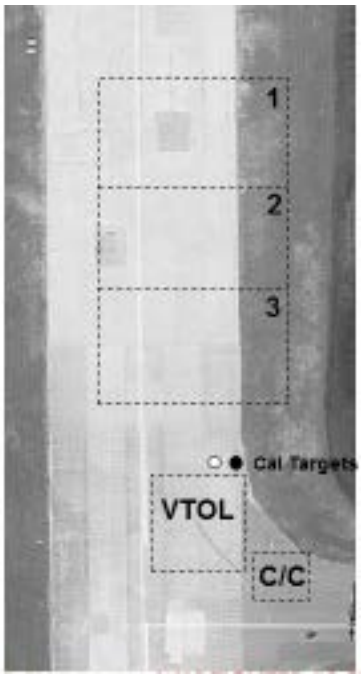

Figure 4. Test setup at Silver Flag.

Imagery of the runway was collected both before and after the blast. The SUAS started at the VTOL, flew North through sections 3, 2, and 1, then turned around and flew through the same sections. Upon returning to the VTOL area it hovered at altitude while the camera settings were monitored and tuned from the C/C area. The SUAS then flew the same path again with the new camera settings.

Some ground truth imagery is shown below in **Figure 5**. There are three spalls on the right which are not visible since they are covered by debris from the adjacent crater. (A spall is an area in which the runway surface is partially damaged/removed but the damage does not penetrate the runway.) The crater is approximately 4' in diameter and 1' deep. The camouflet is 6" in diameter. Note that prior to the blasting, a hole was drilled into the concrete and the explosives were implanted. This is obviously different from a live munition but was required for safety reasons.

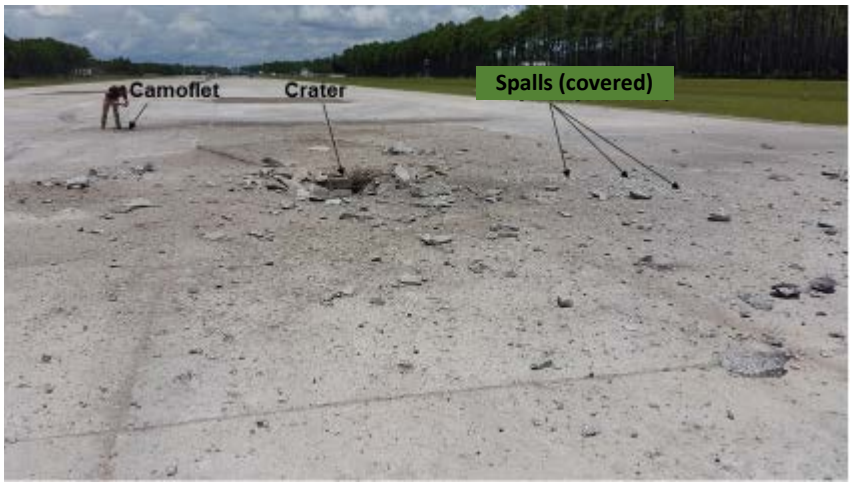

Figure 5. Post-blast camouflet, crater, and spalls (covered by debris from the crater).

3 CHANGE DETECTION

A sample of the resulting images from the camera on board the UAV is shown in **Figure 6**. These before and after images were taken over section 1 of the runway (see **Figure 4**). In the upper portion of **Figure 6**, the new holes that have been drilled for the explosives are visible. Also visible are some drilled holes used during previous tests as well as some old damage that has been partially cleared away (crater, spalls, etc.).

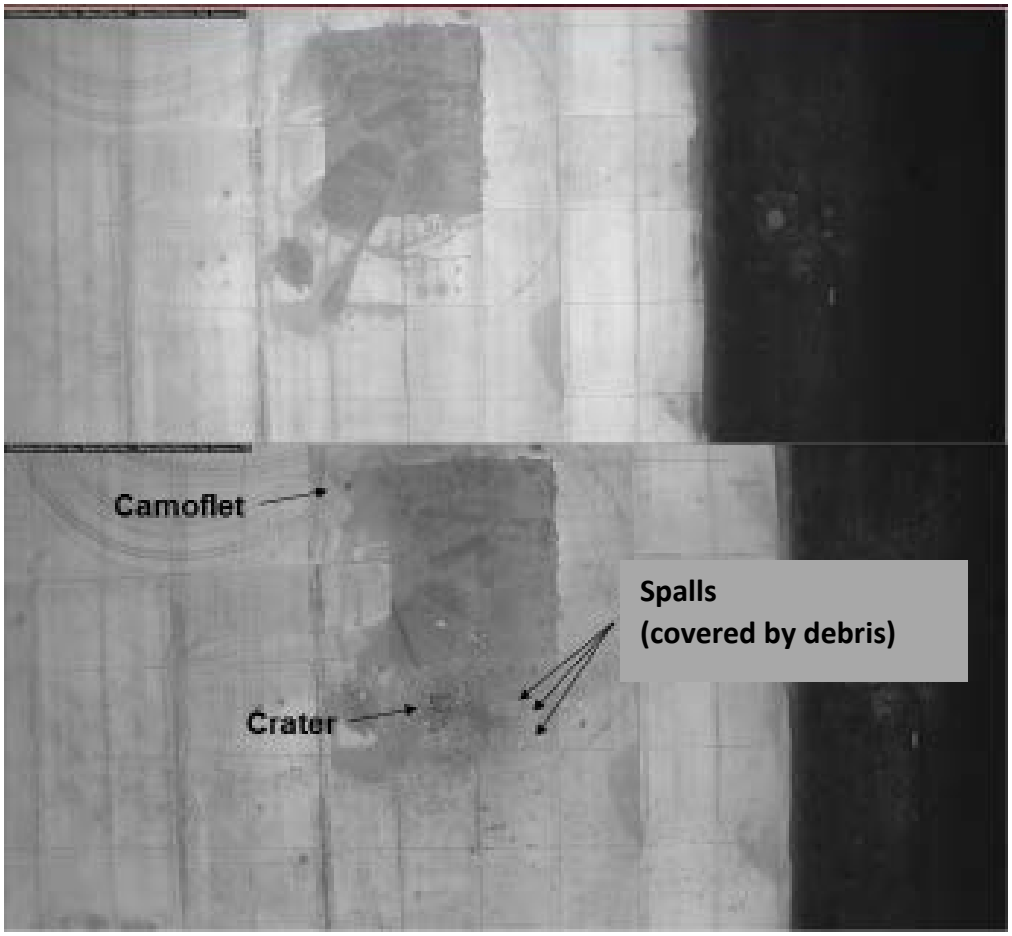

Figure 6. Runway section 1 before (top) and after (bottom). Image is grayscale.

Our change detection algorithm includes edge finding methods in the value (i.e., grayscale) image, see **Figure 7**. The strength of the edge (i.e., the gradient) is reflected in the brightness/intensity of the edge image. Thus, while a number of edges are found, when the two edge images are subtracted and a threshold is applied the result will include only the sharpest edges (i.e., strongest gradients), see **Figure 8**. On the left in this figure is the change in edges resulting from the subtraction of the before and after edge images of **Figure 7**. Some of these edges made it past the threshold due solely to the fact that the lighting changed and the edge appeared stronger; these edges are present in the before edge image and are not a product of any event or explosion. To remove these false alarms, we simply look to see if any of the thresholded edges are close to an edge in the baseline image. The downside is that part of an edge can sometimes be filtered out using this method but if the object is large enough, the remainder of the object's edges are enough to make it through this filtering step. The result of this filter is shown in the right side of **Figure 8**.

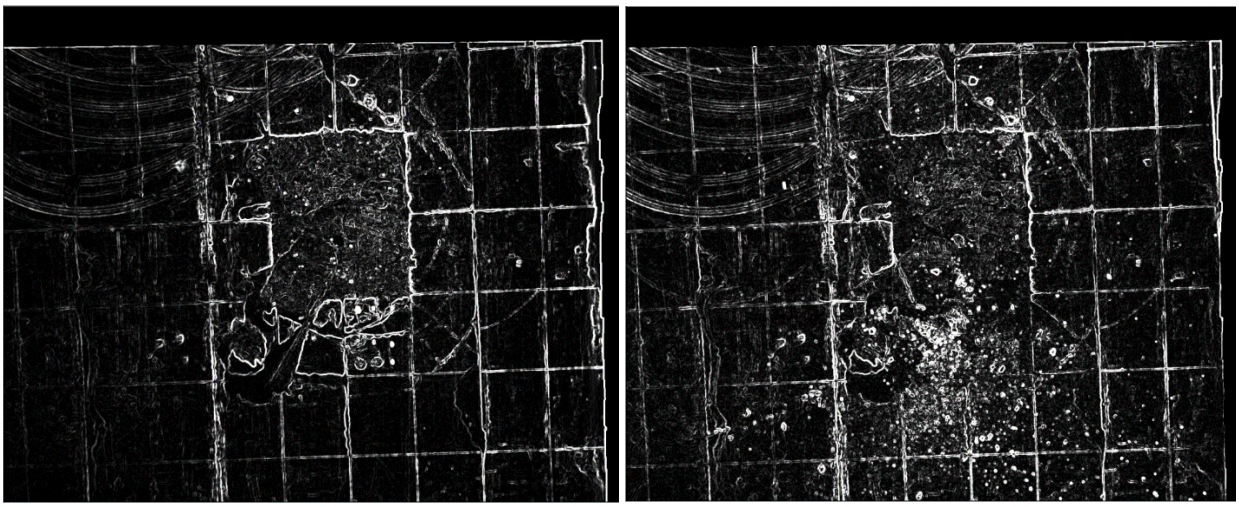

Figure 7. Runway section 1. The edge image before (left) and after (right). Brightness of the pixel corresponds to the sharpness of the edge.

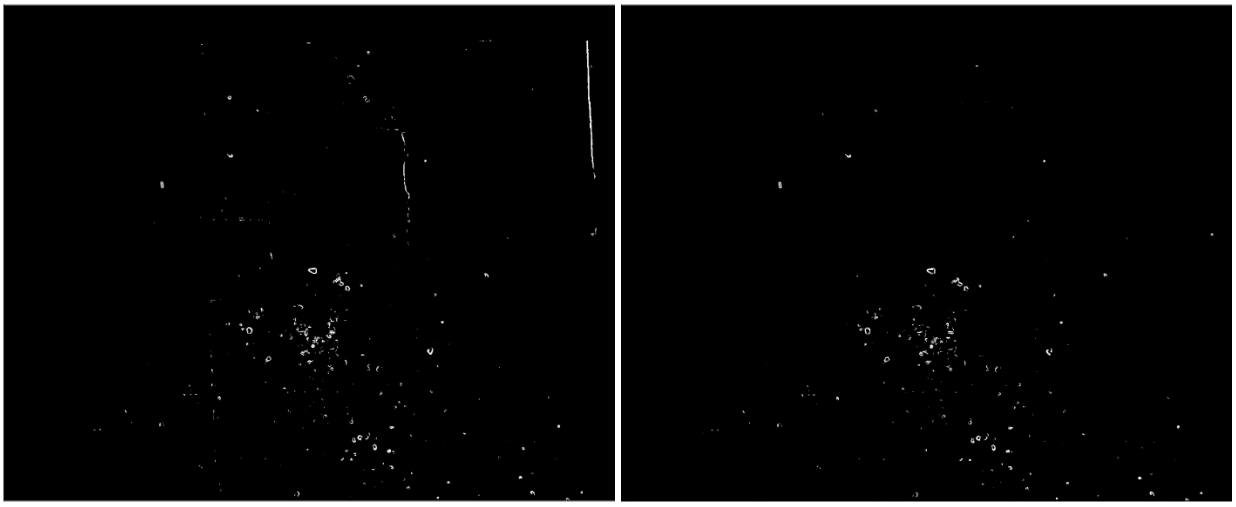

Figure 8. Runway section 1. Strongest gradients of the change in edges (left). False alarms filtered out (right) showing only debris.

4 CLASSIFICATION

Once the false alarms are removed, what is left may be either a crater, a camouflet, a piece of debris, an UXO, or a null/false alarm (not every image processing algorithm is perfect so we try to catch these false alarms at the component and system level; e.g., Scout and Surveyor). Our latest work employs neural networks to make this classification but other, early attempts were also successful. We describe both methods starting with the earlier attempt first.

To perform this classification without using neural networks, we necessarily limited the classes to two: crater/camouflet or debris. To choose between the two, we utilize the average value provided by the black calibration target (see **Figure 4**) and search within a rectangular box that bounds the edges of each piece of debris displayed in the right side of **Figure 8**. If that black calibration target value is found within the bounding box of a piece of debris, then that bounding box is classified as containing a crater/camouflet. The result of this operation is shown on the left in **Figure 9**. The final image is shown on the right in the same figure in which the crater and camouflet are boxed in red while the entire debris field is boxed in green. An alternate representation is to color the final image with red crater pixels (see left side of **Figure 9**) and green debris pixels (see right side of **Figure 8**). Also note that the customized algorithm is now able to find the edges of the runway automatically as indicated by the gray border.

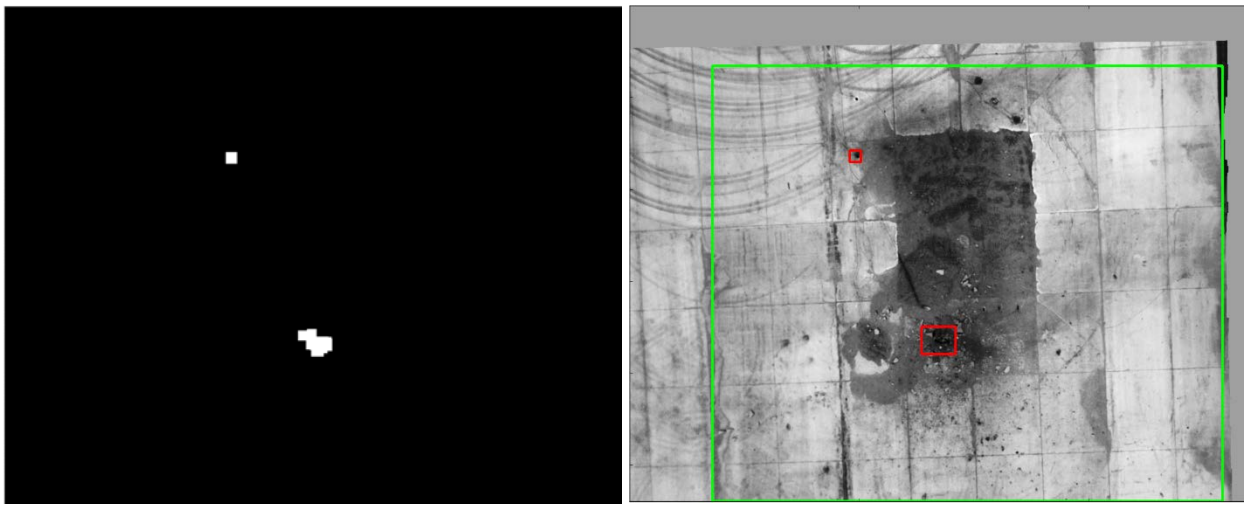

Figure 9. Runway section 1. Crater image (left) and final image (right) with red boxes showing a crater and a camouflet and the green box showing the extent of the debris field.

This approach, while rather simplistic worked fairly well but did not lend itself at all to being extensible to additional classes. The best method for classifying the changed areas involves the training and evaluation of neural networks, i.e. "Artificial Intelligence". Neural networks (a network of "neurons") are simply a clever collection of linear equations with an output function. In its simplest form, a neuron consists of the inputs to a linear equation where the inputs are weighted and the output function is chosen so that the neuron "fires" when the linear equation result reaches a threshold. More complex neurons are typically utilized in which the output function is smooth and not a step function. Arranging the neurons into layers and providing for a "dense connection" between layers (each neuron in each layer is connected to every neuron in the preceding and subsequent layer) is what gives the neural network the power to perform calculations that, if employed with traditional IF-THEN-ELSE structure, would quickly drive the researcher to reconsider their career choice due to the dizzying and often incomprehensible result. In other words, neural networks are good at addressing things that are hard/impossible to program step by step. Often, the problem sets in which neural networks are applied is easily or routinely solved by a human as in this case, "Draw a red box around the crater".

The weights of each neuron's input is calculated by comparing the actual result of the network's output against the desired result. The difference is back propagated through the layers using a supervisory control routine so that the new weights better approximate the desired result. The drawback of neural networks is that they need a lot of examples to convergence to a solution and the examples must be carefully managed so the network doesn't just memorize the answer.

Here we note that it is impossible to avoid the imprecise language of "training", "memorizing", "learning", "behavior", etc. In reality, the network does no such thing but only appears to share some similar traits with their biological inspiration; however, to invent words to make this distinction would be somewhat pedantic and unnecessary.

To get the data ready for input to the neural network, we created a "truthing tool" for all the original image patches. Due to the limited number of damage sites available, it became necessary to expand the original image patches to a set of 5,000 total images by varying the brightness, contrast, color, shift, rotation, scale, etc. of the original images. The truthing tool organizes the original image patches and asks the user to classify each image. If a crater or camouflet is present in the image, the user is asked to locate it and provide the radius of the damage. This truthing data is then attached to the original as well as the corresponding expanded imagery. Note that in some cases, even a human could not classify 100% of the original image patches.

To avoid memorization, we divide the truthed data into three subsets: training, validation, and test. Of the 5,000 images, we chose 80% for the training set, 17% for the validation set, and the remainder for test. The training set is used to calculate a score proportional to the difference between the desired and actual network output and to set the weights. Since a lower score is better, it is generally referred to as "loss". The validation image set is withheld until the entire training set has been utilized the first time. To ensure the network is learning, the validation set is then

evaluated (without setting the weights). The training set is then shuffled and the process is repeated. Plotting the loss for the training and validation set, we should see an asymptotic approach to some minimum value which is largely determined by the architecture of the network (how the layers are constructed) and the quality of the data. Once the researcher has determined enough training has taken place (or they get impatient and want to see the result!) the training process is stopped and the researcher can then utilize the test image set. These provide the researcher with a qualitative understanding of how well the network has been trained; by exposing the network to entirely new imagery, one can get a sense of how well the network might work in the real world.

As mentioned before, the results obtained from a neural network are not only related to the quality and organization of the data but also the network architecture. Rather than reinvent the wheel, there are many generalized network architectures for object recognition that are publicly available (like VGG16, for instance). We implement one of these networks and freeze those weights. Onto the output of the generalized object classifier, we implement a flattening layer followed by a dense layer followed by four dense output layers, or heads, see **Figure 10**. One of the output heads indicates which the four classes the image patch belongs to while the other three outputs provide the row location, column location, and the damage radius (all in pixels). Using the same network for all four outputs simplifies the training and implementation of the neural network.

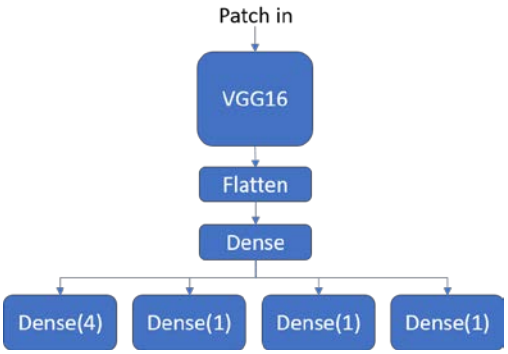

Figure 10. One example of a RADAS classifier network.

The dataset was split into three subsets as explained above. The following is cut and pasted from the Jupyter notebook to illustrate the dimensionality:

reading the previously split & saved testing, training, and validation sets...
x_train: (4000, 128, 128, 3) **y_train**: (4000, 7)
x_val: (1000, 128, 128, 3) **y_val**: (1000, 7)
x_test: (40, 128, 128, 3) **y_test**.shape: (40, 7)

The model was then trained over 30 epochs. An epoch is a training cycle in which the network is exposed to all the training images in order to set the weights; it is subsequently evaluated on all the validation images. The images are reshuffled and the epoch begins again. The loss history (recall that lower is better) is shown in **Figure 11**. Note in this figure that the shape of all the curves asymptotically approaches some minimum value (i.e. the second derivative of the curve does not change sign). If the network were memorizing the answer, we would tend to see inflection points in the curves. This generally shows that the network has successfully learned as well as it can given the structure of the network and the limitations of the image dataset. The results are quite good given the limited number of images in the dataset. The classification results are summarized as follows using the validation dataset:

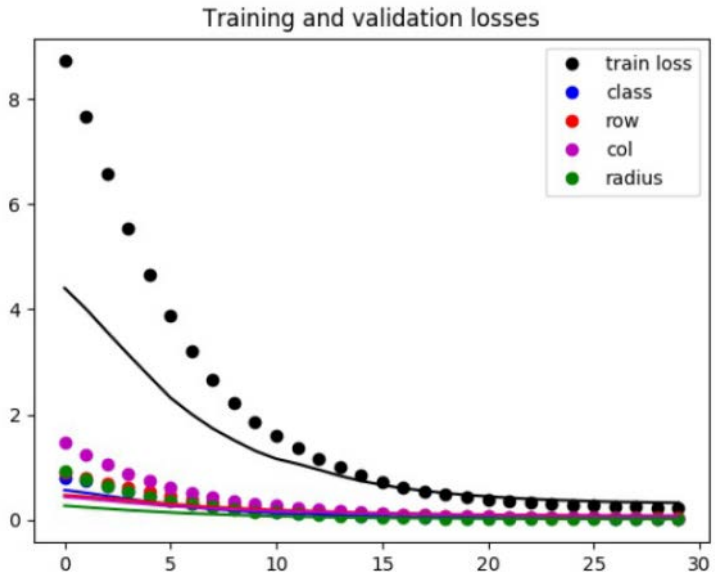

Figure 11. Training and validation score for 30 epochs for the 4 network outputs: class, row, col, and radius. Training score is shown as circles, validation loss

- Of 267 Crater images, 0 were mis-classed (**100%** accurate).
- Of 161 Camouflet images, 1 were mis-classed as Debris (**99.4%** accurate).
- Of 276 Null images, 14 were mis-classed as Debris (**95%** accurate).
- Of 296 Debris images, 0 were mis-classed (**100%** accurate).

The location of the damage in the image patch (row, col) and the radius of the damage (in pixels) for the crater and the camouflet are nearly precisely the same, see **Table 1**. We use the *median* percent absolute prediction error due to the presence of a very small fraction of the predictions being off by an extreme amount which can skew the mean.

Table 1. Quantitative assessment of neural network performance to perform damage metrology.

Damage Type	Predicted Row Err	Predicted Col Err	Predicted Radius Err
Crater	14.9%	10.5%	5.2%
Camouflet	16.0%	10.9%	6.7%

5 CONCLUSION

The network achieved very good results. The ability to provide not only classification and location but also initial measurements of damage size from the Scout imagery provides strong capability from a relatively simple sensor. Continued development and improvement to ensure capability across a wide range of operating parameters requires continued data collection of actual runway damage.

Future work will focus on taking advantage of the latest advancements in neural networks. Many networks are now publicly available that provide the same or better results, faster. Furthermore, they achieve this using either less computational resources or computational resources that are well suited for this type of work such as a Graphics Processing Unit (GPU).

The authors would like to thank everyone at the Air Force Civil Engineering Center for all their hard work, diligence, and can-do attitude.