



Identification of Partially Resolved Objects in Space Imagery with Convolutional Neural Networks

Christopher A. Ertl¹ · John A. Christian¹

Published online: 3 June 2020
© American Astronautical Society 2020

Abstract

The identification of partially resolved space objects in an image and matching it to a database of known objects is useful for many scenarios. Machine learning can identify objects at various distances, relative attitudes, and phase angles. In this study, a convolutional neural network is constructed and evaluated. Performance as a function of phase angle is used to evaluate the capabilities of the neural network. The network's ability to correctly identify objects blurred to various degrees is also assessed. A method for novelty detection is explored. Numerical results suggest that the use of deep learning may be a viable option for the identification of partially resolved objects.

Keywords Deep learning · Partially resolved objects · Convolutional neural network · Space imagery

Introduction

The autonomous identification of objects in space imagery is an enabling technology for a variety of space exploration missions. Objects in images can range from point sources that illuminate only a few pixels (which we refer to as *unresolved objects*) to large extended bodies that span many pixels (which we refer to as *resolved objects*). Currently, there are established techniques that are used to identify objects at both ends of this spectrum. In some situations, however, the object that we observe is of

An earlier version was presented as a conference paper, AAS 18-412, at the AAS/AIAA Astrodynamics Specialists Conference, Snowbird, UT, August 19-23, 2018

✉ Christopher A. Ertl
ertlc@rpi.edu

John A. Christian
chrisj9@rpi.edu

¹ Rensselaer Polytechnic Institute, Troy, NY 12180, USA

intermediate size (which we refer to as *partially resolved*), and identification in this regime is especially challenging.

Deep learning using neural networks is one way of addressing the classification problem. Recently, deep learning has gained popularity within the computer vision community [63]. As neural networks have improved, the computational cost has diminished while the accuracy has improved [65]. With applications such as object recognition [36], object tracking [70], and many others [6, 62], it is clear that there is a potential for deep learning in the space domain.

While a wide variety of neural network architectures exist, this study considers neural networks designed with both convolutional and fully-connected layers. This neural network architecture is trained using a large database of images of known space objects and then tested using a separate set of test images. Section “[Rendering of Training and Test Images](#)” discusses the rendering of the database of images. Section “[Neural Network Design](#)” gives a basic background of neural networks and discuss the architecture of the neural network we have developed. Section “[Network Training](#)” examines the training of our network. Lastly, Section “[Results](#)” discusses the results of the crisp images, blurred images, and novelty detection.

Background

Identification of Unresolved and Fully-Resolved Objects

Objects within space imagery may appear at varying resolutions, ranging from unresolved to fully-resolved. Unresolved objects can be well approximated as a point source and only illuminate a few pixels. A fully-resolved object appears large enough in the image that individual features on the object may be discerned.

The prevailing technique for the characterization and classification of unresolved objects is lightcurve inversion. The use of lightcurves to synthesize information of object shapes has been discussed as far back as 1906 [57] and gained popularity in recent decades with techniques developed by Kaasalainen and Torppa [29] and Kaasalainen et al. [30]. Lightcurve inversion is a mathematical technique in which light intensity over time is used to determine the rotational period and provide insight into the shape model of an object. Past work has shown success in model development and classification for both asteroids [29, 30, 45, 69] and artificial objects [15, 17, 27, 41, 42, 46].

The higher spatial resolution of fully resolved objects permits the use of classical machine vision techniques for object recognition, including: histogram of oriented gradients (HOG) [9, 72], scale invariant feature transform (SIFT) clustering [5, 43, 49], speeded up robust features (SURF) [3, 32, 59], features from accelerated segment test (FAST) [34, 55], oriented FAST and rotated BRIEF (ORB) [50, 71], and others. Many analysts, however, have recently moved away from hand-crafted features in favor of deep learning techniques for object identification [1, 47].

Between the two extremes of object resolution discussed above is the category of partially-resolved objects. These objects are no longer point sources of light, nor do they have the spatial resolution of fully-resolved objects. Partially-resolved objects

have a discernible overall shape and generally span 5–30 pixels in their longest direction for a well focused camera. Camera defocus may cause larger objects (in terms of pixel extent) to still appear “partially resolved”, as this term is used to describe situations where the objects overall shape is apparent but individual surface features are not. Thus far, there has been little work involving the identification of partially resolved objects [53]. This work provides initial steps for one solution to this problem.

Classification with Neural Networks

Neural networks are a powerful tool for pattern recognition. One of the first successful instances of neural networks comes from Widrow and Hoff’s “Adaline” adaptive linear, pattern classification machine [73]. This network was developed to recognize a pattern of binaries and output a resulting binary.

The first convolutional neural network was developed by Fukushima in 1980 [16]. The neocognitron was inspired by the work of Widrow and Hoff and introduced two types of layers: convolutional layers and downsampling layers. Shortly after this, the method of back-propagation was deployed to the problem of machine learning [56]. This method iteratively changes the weights and biases within the neural network in order to minimize the output error, and is still used by most deep learning algorithms.

One of the first computer vision challenges met by neural networks was to classify images in the Modified National Institute of Standards and Technology (MNIST) database [39]. The database includes thousands of handwritten letters and numbers. Early neural networks used to solve this problem were relatively “shallow” by today’s standards such as LeNet [38], a network with two convolutional and two fully-connected layers. LeNet was highly successful at this classification task when compared to the conventional computer vision methods being used at the time.

As the images requiring classification became progressively complex, so did the convolutional neural network architectures. Leading to deeper neural networks. One of the first “deep” neural networks, AlexNet [36], was trained to classify objects within the ImageNet database [8]. The ImageNet database contains thousands of labeled images of thousands of different objects. AlexNet’s architecture includes five convolutional layers and three fully-connected layers. To reduce over-fitting, where a neural network is not able to generalize outside of its training data, the network utilized random dropout [22]. The network performance was also improved by using activation function Rectified Linear Units (ReLU) [7], which removes negative activations.

After AlexNet, the depth of convolutional neural networks increased further with notable examples being developed by industrial research laboratories, such as Microsoft’s ResNet [21] and Google’s DeepDream [64]. These networks include millions of parameters and continue to grow with newer iterations. A majority of the current literature for object recognition using deep learning makes use of a variation of convolutional neural networks [31, 40, 60]. Our network architecture utilizes both convolutional and fully-connected layers, which will be discussed in greater detail in Section “[Neural Network Design](#)”.

Of special relevance here, convolutional neural networks have been used by astronomers to classify a variety of space objects in images [2, 10, 26, 33, 51].

Other Machine Learning Classification Methods

Although neural networks are one of the most widely used machine learning methods for classification, there exists other classification methods. Random decision forests were developed in [23] and use decision trees for classification. The method has been used for both terrestrial [4, 61] and space [48, 54] image classification.

The k-nearest neighbors algorithm, developed in 1951 [13], is a non-parametric classification algorithm. Similar to random forest, k-nearest neighbors has also been used for image classification [14, 66, 74].

Rendering of Training and Test Images

Neural networks require a large and diverse set of images for both training and testing. We accomplish this by rendering synthetic images of 14 objects, seven spacecraft and seven asteroids, using 3D triangular mesh models obtained from the NASA 3D Resources repository¹ and the NASA Planetary Data System² (PDS). The seven spacecraft include Cassini, Far Ultraviolet Spectroscopic Explorer (FUSE), Galileo, Hubble Space Telescope, International Space Station (ISS), Maven, and Voyager. The seven asteroids include 101955 Bennu, 6489 Golevka, (8567) 1996 HW1, 25143 Itokawa, 216 Kleopatra, 4486 Mithra, and 4 Vesta. These 14 objects are known to the network, which attempts to classify any observed object using one of these 14 labels. Since we may often encounter novel objects that are unknown to the network and for which no training has been performed, we render an additional ten objects from other repositories. These additional ten objects are used to evaluate how our network behaves when challenged with an object outside its training set, and, if we can, reliably identify such a scenario.

All rendering is performed using the open-source Blender software package.³ The simulation environment is configured to take images of the objects from varying relative attitudes, distances, and phase angles. Test and training images are 8-bit monochrome with dimensions of 30×30 pixels. The objects are placed such that they span 5 to 30 pixels in their longest direction within these images. In real imagery, where the full image is substantially larger, we would perform classification on a 30×30 window centered on the observed object. Figure 1 demonstrates this process.

The images for each object were rendered using Blender's application programming interface (API). For each object we rendered 60,000 training images and a separate set of 5,000 test images. Both the training and test images have a uniform distribution in range, relative attitude, and phase angle. The range spans the distance

¹<https://nasa3d.arc.nasa.gov/models>

²<https://pds.nasa.gov/>

³<https://www.blender.org>

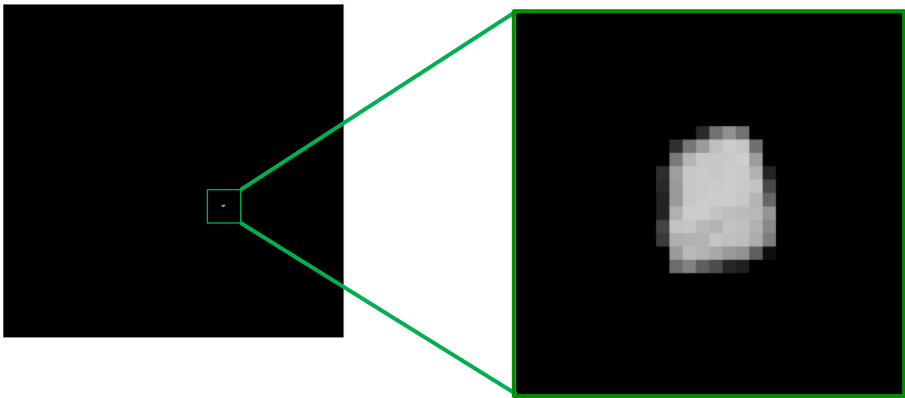


Fig. 1 Example of converting from a full image to 30×30 window centered on the observed object

where the object appears to be 5 to 30 pixels in width and the phase angle is from 0 to 138 deg. Once a range is chosen for an image, we randomly sample a location on a sphere with an origin at the centroid of the object and a radius being the chosen range. We then adjust the camera's attitude such that the object is always at the center of the 30×30 patch. This allows us to render each object from a uniform distribution of relative attitudes. The illumination direction is then calculated using the known phase angle and the camera quaternion.

The object ID, relative attitude, and range are all unknown to the network when challenged with a test (or operational) image. In contrast, a good estimate of the phase angle, g , is generally known since the Sun is presumed to be much farther away than the distance between the camera and observed object. Therefore, taking advantage of this knowledge, the images for each object are separated into five overlapping bins based on the known phase angle. This results in 12,000 training images and 1,000 test images per bin, per object. Figure 2 provides a visual representation of the phase angle ranges used by the five training bins. Although the training data is uniformly sampled over phase angle, the trained networks may not be fully developed at their boundaries. The operating bins reduce the phase angle range by 8 degrees ensuring that test images conform to the constraints of the trained networks. Table 1 give the exact values for the phase angle ranges of the training and operating bins. Examples of the rendered images are provided in Fig. 3. In previous work, it is shown that convolutional neural networks perform poorly when trained on the entire phase angle range as compared to when they are trained on a portion of the phase angle range [12]. For this reason each of the five overlapping phase angle bins will train a separate convolutional neural network of the same architecture.

Additionally, to better understand the effect of defocus on performance, we also develop sets of training and test images that are blurred to varying amounts. These sets of training and test images are versions of the crisp test images already rendered with defocus being simulated by convolution of each image with a Gaussian kernel having the appropriate standard deviation. The Gaussian kernel was selected since

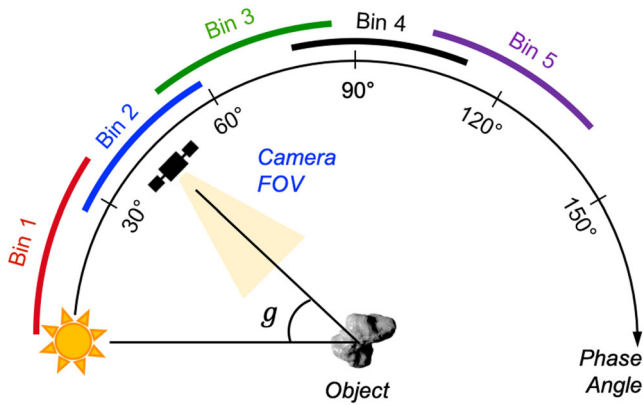


Fig. 2 A visualization of the five overlapping ranges of phase angles (g) used for training and operating bins. As the angle between the sun and the observer increases the visibility of the object degrades

this is known to be a good approximation of blur from defocus for well-built cameras [28, 52]. Thus blur is simulated according to,

$$I_{blur} = G \otimes I \tag{1}$$

where \otimes is the 2D convolutional operator, I is the original image, I_{blur} is the blurred image, and G is the Gaussian kernel.

Neural Network Design

Background of Neural Networks

Although there are a multitude of applications for neural networks ranging from function approximation [25] to natural language processing [6], this work focuses exclusively on classification and pattern recognition in digital images. Originally, the

Table 1 Phase angle ranges for both training and operating bins

Bin	Training Phase Angle Range (deg)	Operational Phase Angle Range (deg)
1	0 – 34	0 – 30
2	26 – 60	30 – 56
3	52 – 86	56 – 82
4	78 – 112	82 – 108
5	104 – 138	108 – 134

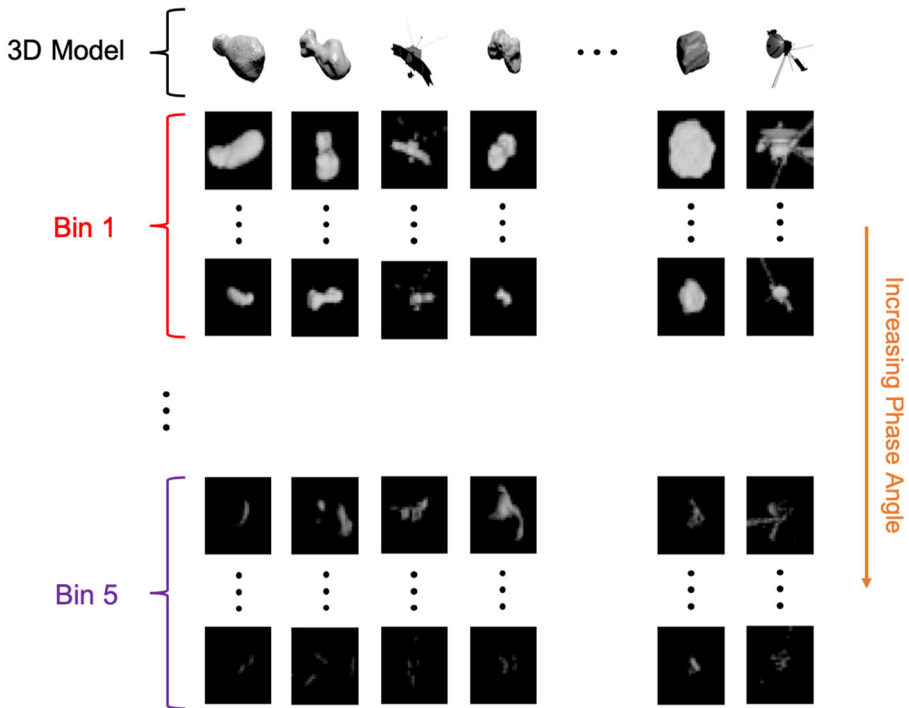


Fig. 3 Images are rendered using a 3D model at varying attitudes, ranges, and phase angles. These images are then sorted into 5 overlapping bins based on phase angle

neural networks used for these problems were made of only fully-connected layers. The typical architecture for these fully-connected neural networks included an input layer consisting of a flattened array of the image. Each pixel value of the flattened array connects to every neuron in the next layer. The layers of neurons between the input layer and the output layer are called the hidden layers of the network. Each of the hidden layers are fully connected to both the previous layer and the next layer. The neurons of the output layer are used to determine how the network has classified the image. This is illustrated in Fig. 4.

The output of a neuron in a fully-connected network is calculated as

$$x^{(n+1)} = \omega^{(n+1)}x^{(n)} + b^{(n+1)} \quad (2)$$

where x is a vector of the neuron activations, ω is the matrix of weights associated with the connection of neurons between two subsequent layers, and b is the bias vector. The superscript defines the layer.

Convolutional neural networks were developed to improve upon the successes of the fully-connected network for computer vision applications [20]. Unlike the fully-connected neural network, a convolutional neural network allows for shared weights among pixels and neurons rather than a single weight being assigned to every pixel

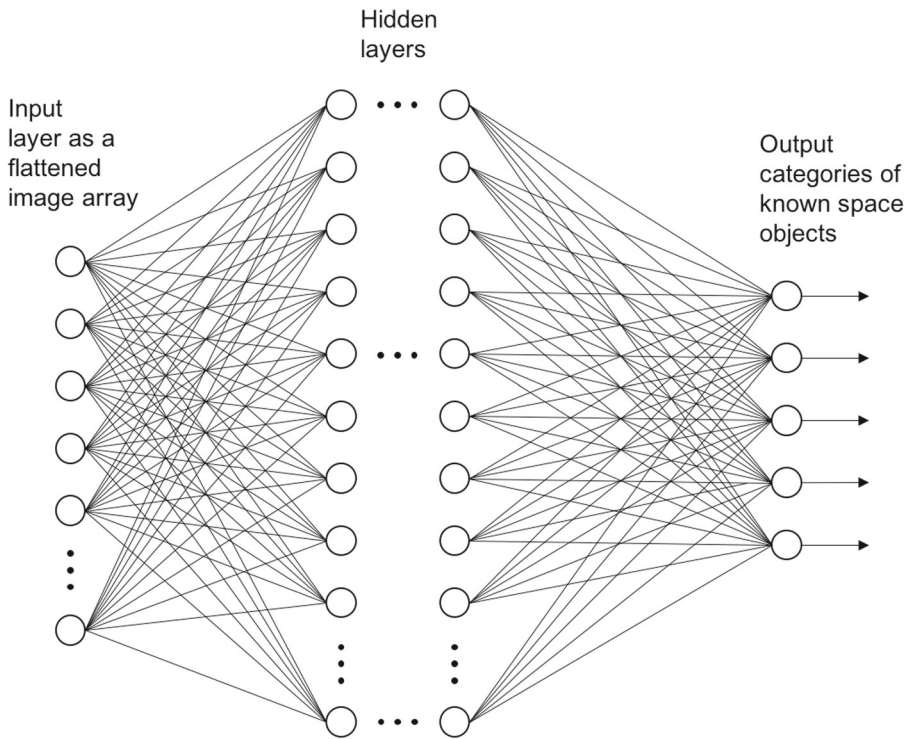


Fig. 4 A basic visualization of a fully-connected network showing the three major components of the network

and neuron in the network. Similar to the 2D convolutions used extensively in classical computer vision, a convolutional layer takes a filter and convolves it with an image to create a feature map [44]. The output pixels comprising the feature map are calculated as

$$x_l^{(n+1)} = \sum_k \omega_{k,l} \otimes x_k^{(n)} + b_l^{(n+1)} \tag{3}$$

where the superscript defines the layer, the l is the feature map for the current layer, and k is the input channel. Since convolutional layers may output multiple feature maps, there are an equal number of kernels, w , for the current feature map to the number of feature maps in the previous convolutional layer. These kernels make up a filter and the number of filters equal the number feature maps desired for the current convolutional layer. Each filter has a bias, b , associated with it. The filters move across the image building up the feature maps.

Both the fully-connected and convolutional neural network include parameters that can be changed to improve the performance of the networks. These parameters include their weights, either from connections between fully-connected layers or fil-

ters in a convolutional layer, and each neuron or feature map's associated bias. The way these parameters are optimized in training will be discussed in Section “[Network Training](#)”.

A more detailed discussion of fully-connected and convolutional neural networks can be found in most texts on deep learning, e.g. [19].

Convolutional Neural Network Architecture

The neural network architecture developed for this work was implemented using the Pytorch machine learning library⁴ and includes convolutional layers, max pool layers, and fully-connected layers. Following a convolutional layer, there can be a max pooling layer which is used to reduce the dimension of feature maps by reducing the elements in a filter to the max element. The final feature map array is then flattened and connected to a fully-connected neural network.

The new network we designed for the present task includes five convolutional layers with each layer using a 3×3 kernel. Input images and features maps are zero padded, such that each convolutional layer produces feature maps of the same size as its input. The network also includes two max pool layers and three fully-connected layers. It should be noted that ReLU is used as the activation function for each convolutional and fully-connected layer. This convolutional neural network architecture was selected after testing multiple architectures and finding the approach presented here to produce the best results. A visualization of our convolutional neural network architecture can be seen in Fig. 5. Five separate networks of this architecture are then trained on the five phase angle bins.

Network Training

Training of a neural network is divided into two pieces: learning and generalization. Learning is the process of adaptively understanding the fundamental aspects of the training data in order to correctly classify it. Generalization is being able to correctly classify data outside of the training data [75]. To train a neural network, a training set of images is used along with a loss function and an optimization method to adjust the parameters in the network.

A network begins its learning process by splitting the training data into batches. Our network uses cross-entropy loss as its loss function. The loss function, L , for each batch of training images is calculated as⁵

$$L = -\ln \left\{ \frac{\exp [x^{(k)} (class)]}{\sum_i \exp [x^{(k)} (i)]} \right\} = -x^{(k)} (class) + \ln \left\{ \sum_i \exp [x^{(k)} (i)] \right\} \quad (4)$$

where the superscripts define the neuron layer, which in this case is the output layer k . Thus, $x^{(k)} (i)$ is the activation of the i^{th} neuron in the output layer. The activation,

⁴<https://pytorch.org/docs/stable/nn.html>

⁵<https://pytorch.org/docs/stable/nn.html>, under CrossEntropyLoss section

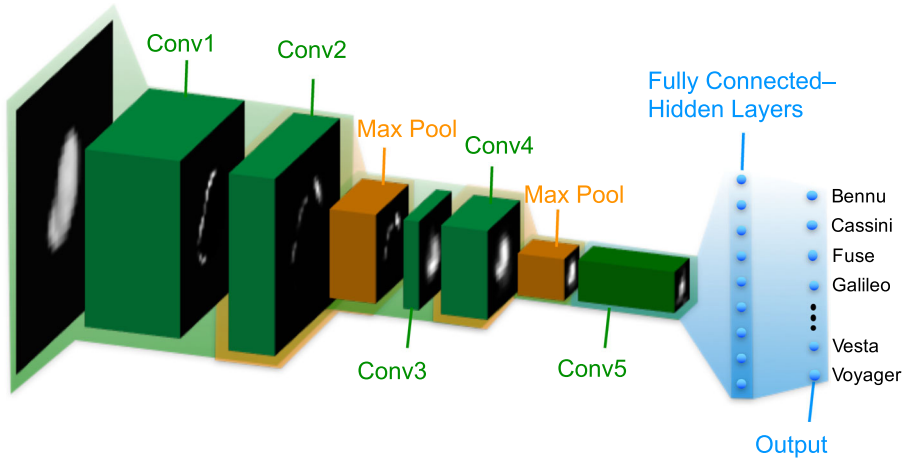


Fig. 5 Our convolutional neural network architecture includes five convolutional layers separated by a max pool layer after every two convolutional layers and three fully-connected hidden layers at the end of the network

x (*class*), is the activation of the classifier that correctly classifies the object in the image. The calculated loss is then utilized by the optimizer.

Using Eq. 4 for a batch of images, the optimizer modifies the parameters of the network through backward propagation with a Stochastic Gradient Descent (SGD) optimizer. We found a momentum of 0 and a static learning rate of 0.0001 to produce the best results for this network. The SGD optimizer is typically used for neural network training and is used to optimize the weights and bias parameters [24].

This process of calculating loss for a batch of images and then optimizing the weights and bias parameters is repeated until there are no more batches left. The images are then rearranged and split up into new batches. This is called an epoch. We trained our network with 250 epochs and batch sizes of 10.

Each of our networks, both crisp and blurred, experience a minor amount of overfitting. Overfitting can be observed by monitoring the training and validation loss at each epoch while the network is training. The network is said to be “overfit” if the loss of the validation set (a small set of training images not used for training) remains higher than the loss of the training set. In general, overfit networks are less able to generalize. Both Figs. 6 and 7 show that there is only a minor amount of overfitting, with the network trained on crisp images experiencing less overfitting than the network trained on blurred images.

Results

Crisp Image Results

The convolutional neural networks were trained on overlapping ranges of phase angles as shown in Table 1. Each bin included 12,000 images per object for training

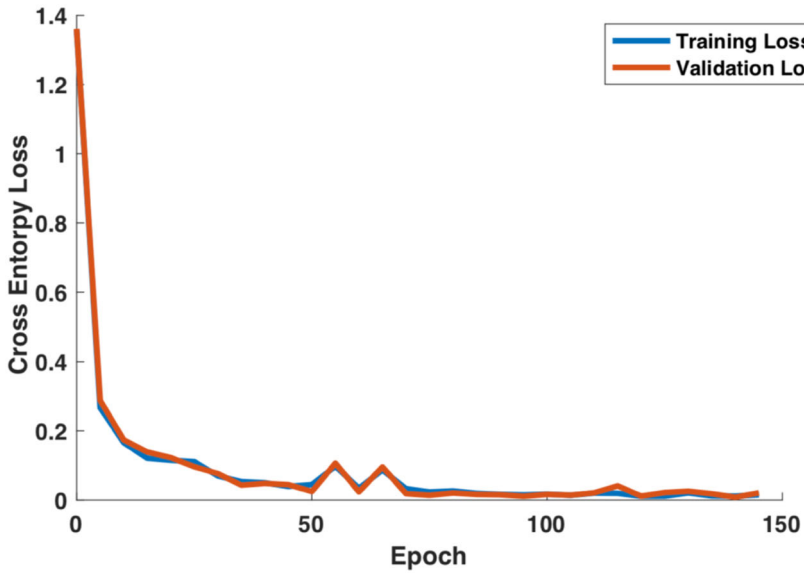


Fig. 6 Training and validation cross entropy loss of a convolutional neural network trained on crisp images. There is a minor amount of overfitting, but in general the validation loss matches the training loss

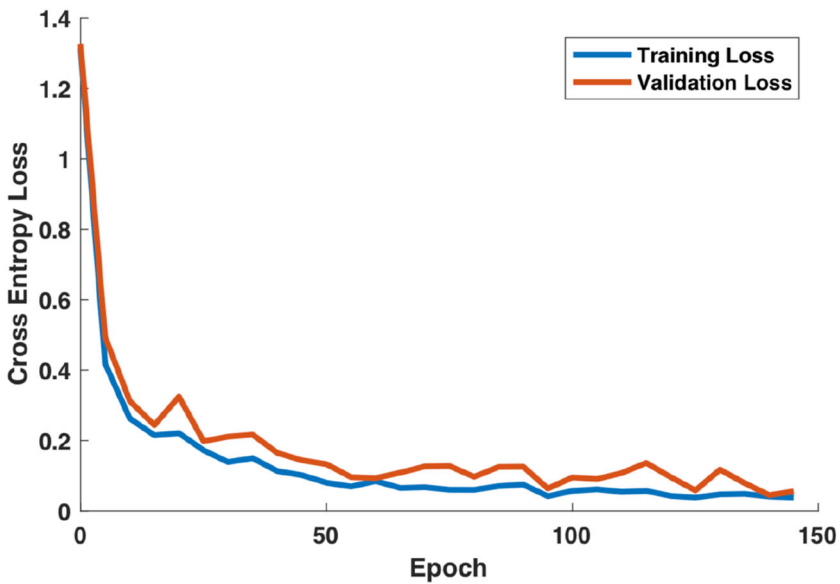


Fig. 7 Training and validation cross entropy loss of a convolutional neural network trained on Gaussian blurred images with a standard deviation of two pixels. Comparing to Fig. 6 we can see there is more overfitting, but still relatively minor

and 1,000 images per object for testing. Each image used in this assessment contains no blurring or distortion. Further, the object in each image is placed at a uniformly distributed attitude and at a uniformly distributed range (making it appear 5–30 pixels in length) — with both the relative attitude and range being unknown to the neural network. A convolutional neural network was trained and assessed for each bin, with success rates shown in Table 2.

As can be seen in Table 2, the success rates stay above 98% in bin 1 and above 86% across all five bins. The success rates tend to decrease at higher phase angles, but this is expected since object visibility decreases at higher phase angles. Thus, the success rate of each object degrades as we move from bin 1 to bin 5.

The confusion matrix for the bin 1 convolutional neural network can be found in Table 3. The data show that a majority of our classifications for the bin 1 convolutional neural network are correct, with the largest confusion coming from our network incorrectly classifying Bennu as Vesta in about 2% of the test cases.

Examples of images with objects classified by the convolutional neural networks can be found in Fig. 8.

Blurred Image Results

The results of Section “Crisp Image Results” summarizes the success rates of the network when trained with crisp images. In actuality, images captured during a space-flight mission may be blurred due to defocus, jitter, or other effects. The network must be assessed on its ability to generalize not only with varying ranges and phase angles, but also varying blur. Two networks are assessed, one is a neural network trained with only crisp images and the other is a neural network trained the images

Table 2 Success rate (%) of networks trained and tested on only crisp images. Success rates of each object degrade at higher bin numbers

Object	Bin1	Bin2	Bin3	Bin4	Bin5
Bennu	98.1	98.8	98.8	95.0	91.5
Cassini	99.5	99.4	99.4	96.9	91.2
FUSE	99.2	99.9	99.5	97.2	87.8
Galileo	99.9	99.8	99.1	97.8	94.8
Golevka	98.8	99.7	99.6	97.8	91.6
Hubble	99.9	99.7	99.8	97.4	94.6
HW1	98.5	97.2	99.1	98.2	95.2
ISS	98.7	98.8	97.6	94.2	86.3
Itokawa	99.3	98.9	98.1	98.4	96.5
Kleopatra	99.4	99.4	98.2	98.0	96.4
Maven	99.8	99.9	99.0	97.2	91.2
Mithra	99.7	99.4	98.9	96.9	94.8
Vesta	99.6	99.2	96.4	97.6	94.7
Voyager	99.4	98.8	98.6	97.3	93.6

Table 3 Confusion matrix for the bin 1 convolutional neural network. The confusion matrix provides information on the correct classifications (diagonal of the matrix) and the incorrect classifications (off diagonals)

		Predicted													
		Bennu	Cassini	FUSE	Galileo	Golevka	Hubble	HW1	ISS	Itokawa	Kleopatra	Maven	Mithra	Vesta	Voyager
Actual	Bennu	981	0	0	0	0	0	0	0	0	0	0	0	19	0
	Cassini	0	995	0	0	1	0	0	0	1	0	2	0	0	1
	FUSE	0	0	992	0	0	0	0	2	4	1	0	1	0	0
	Galileo	0	0	0	999	0	1	0	0	0	0	0	0	0	0
	Golevka	0	0	0	0	988	3	0	0	1	0	0	8	0	0
	Hubble	0	0	0	1	0	999	0	0	0	0	0	0	0	0
	HW1	2	0	1	0	1	0	985	0	2	4	0	5	0	0
	ISS	0	2	8	0	1	0	1	987	0	0	0	0	0	1
	Itokawa	2	0	0	0	2	0	3	0	993	0	0	0	0	0
	Kleopatra	0	0	0	0	2	0	2	0	0	994	0	2	0	0
	Maven	0	0	0	0	0	0	0	0	0	0	998	0	0	2
	Mithra	0	0	0	0	1	0	2	0	0	0	0	997	0	0
	Vesta	4	0	0	0	0	0	0	0	0	0	0	0	996	0
	Voyager	0	1	0	0	1	0	0	0	0	0	4	0	0	994

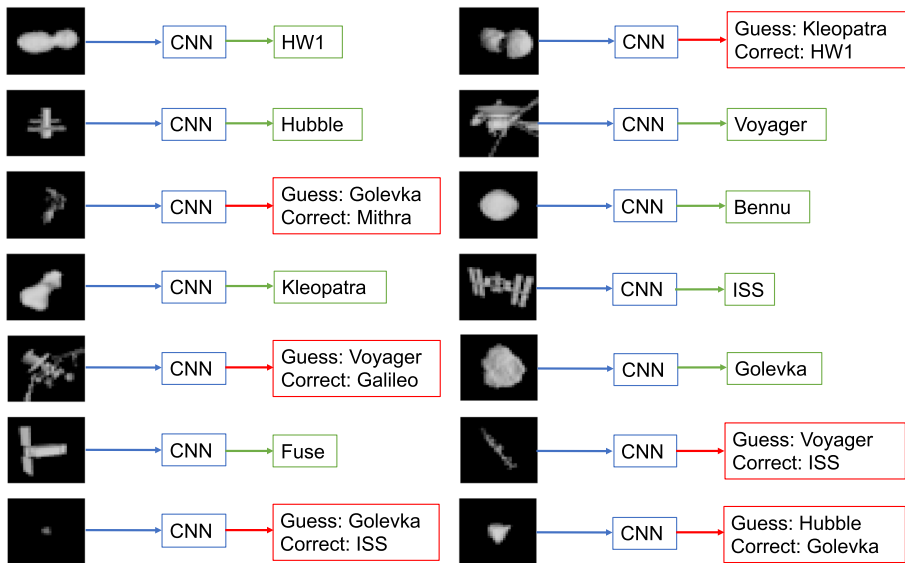


Fig. 8 Examples of images and the classification results produced by our convolutional neural network from Fig. 5. Correct classifications are outlined in green. Incorrect classifications are outlined in red, along with details of the incorrect assignment

blurred to varying amounts. Examples of the images used to test the networks can be seen in Fig. 9.

Figure 10 shows how the bin 1 convolutional neural network responds to increased blur while being trained on both crisp images (left) and images that are blurred between a standard deviation of 0 to 2 pixels (right). In all cases, classification performance decreases with increased blur, which makes sense since information is being lost (the blur is effectively a 2D low-pass filter). What is striking, however, is how quickly the performance of the network trained on crisp images deteriorates for blurs above 0.5 pixels. This suggests the network generalizes poorly with unexpected image blur or defocus. When trained with images of appropriate blur, however, the network shows reasonable performance over the entire range of blurs investigated.

We then trained and tested five of our convolutional neural networks on bin 1 images with fixed degrees of Gaussian blur. The standard deviation of Gaussian blur ranges from zero pixels to two pixels. Each network is trained and tested on the same number of images specified in Section “Crisp Image Results”, but these networks were trained with 150 epochs. As we can see from Table 4, the success rates stay above 90% and there is no significant degradation to the success rates as the training and test images are blurred to an increasing extent. A confusion matrix is provided for the fifth convolutional neural network trained and tested on bin 1 images with a fixed blur width of 2 pixels in Table 5.

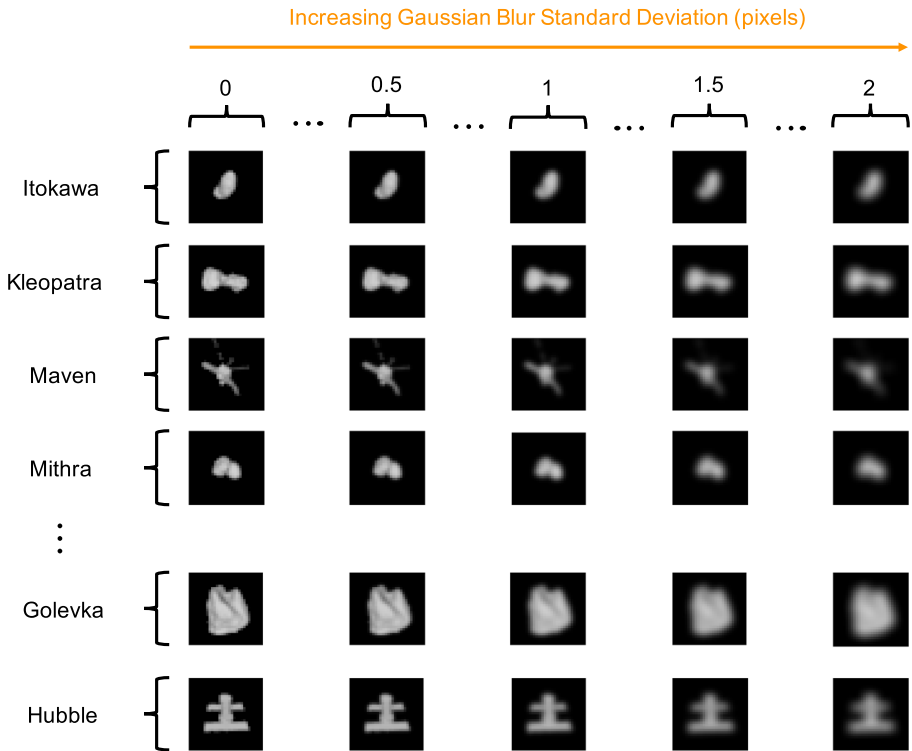


Fig. 9 Examples of test images that are blurred using a Gaussian kernel. Each crisp test image for all objects are blurred to varying degrees

Novelty Detection Results

A significant problem with conventional neural network architectures (including our own) is that any object within an image will be classified by the neural network even if the network was never trained with that particular object. The network will give its “best guess” and classify the image incorrectly with the label from the training set that produced the best match. It is unreasonable to train a classifier for all objects in the known universe, therefore a method must be developed for our neural networks to detect and classify novel objects in imagery. This is where the distinction between generalization and novelty detection becomes important. A network’s ability to generalize comes from its ability to correctly classify a known object in an image it was not trained on and under different operating conditions, while novelty detection is the ability to classify an object outside of the group of known objects as unknown.

In order to assess the network’s ability to detect novel objects, 2,000 images of ten random objects were rendered in the bin 1 phase angle range.

Here, we explore the use of classifier activations as a means of novelty detection similar to the method explored by LeCun et al. [37]. This approach classifies objects as unknown by using the activation of the largest classifier and the activation gap

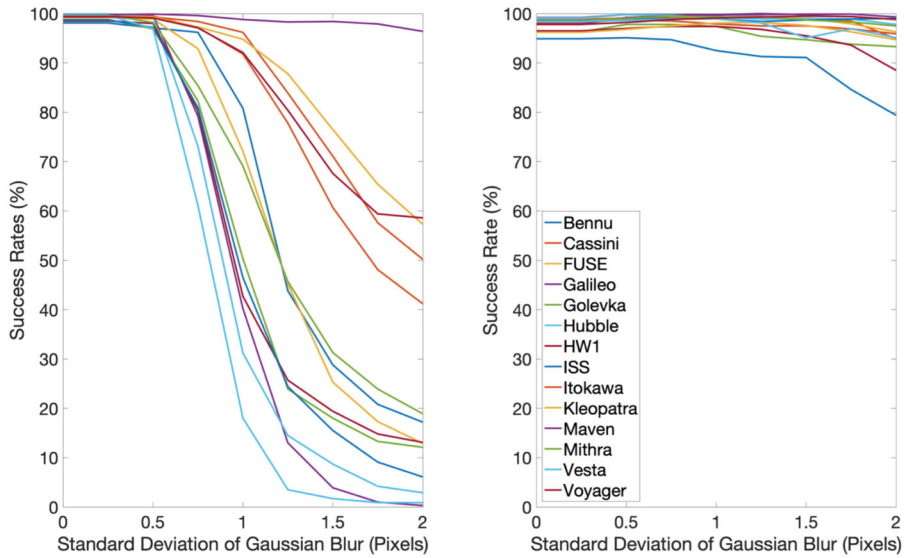


Fig. 10 Object classification performance for two convolutional neural networks tested with increasingly blurred images. One was trained with only crisp images for bin 1 (left) and the other was trained with a range of blurred images for bin 1 (right)

Table 4 Success rate (%) of networks trained and tested on bin 1 images with fixed blur widths

Object	Blur Width (px)				
	0	0.5	1	1.5	2
Bennu	99.5	97.9	93.8	96.5	95.2
Cassini	98.8	99.2	99.1	99.4	98.4
FUSE	100	99.2	99.2	99.1	97.9
Galileo	99.6	99.7	99.7	99.8	99.4
Golevka	98.6	98.7	98.2	97.4	98.4
Hubble	99.9	99.7	99.7	99.5	100
HW1	98.7	98.8	96.9	98.6	95.6
ISS	98.7	98.2	98.9	99.5	99.9
Itokawa	98.4	98.7	98.9	98.3	97.2
Kleopatra	99.2	99.0	99.0	99.1	99.2
Maven	99.5	99.4	99.8	100	99.8
Mithra	99.2	98.9	98.0	98.1	98.2
Vesta	97.8	97.1	99.6	94.5	92.0
Voyager	99.6	99.5	99.2	99.8	99.8

Table 5 Confusion matrix for the bin 1 convolutional neural network both trained and tested on images with a fixed gaussian blur width of 2 pixels

		Predicted													
		Bennu	Cassini	FUSE	Galileo	Golevka	Hubble	HW1	ISS	Itokawa	Kleopatra	Maven	Mithra	Vesta	Voyager
Actual	Bennu	952	0	0	0	10	0	0	0	1	0	0	0	37	0
	Cassini	0	984	1	3	4	0	0	0	2	0	0	0	0	6
	FUSE	0	0	979	0	0	0	14	2	2	1	2	0	0	0
	Galileo	0	0	1	994	0	0	0	1	0	0	2	0	0	2
	Golevka	7	1	0	0	984	1	3	0	0	1	0	3	0	0
	Hubble	0	0	0	0	0	1000	0	0	0	0	0	0	0	0
	HW1	3	0	0	0	2	0	956	0	17	14	0	7	1	0
	ISS	0	0	1	0	0	0	0	999	0	0	0	0	0	0
	Itokawa	3	0	0	0	14	0	6	0	972	1	0	4	0	0
	Kleopatra	0	0	0	0	3	0	2	0	1	992	0	2	0	0
	Maven	0	0	0	2	0	0	0	0	0	998	0	0	0	0
	Mithra	1	0	0	0	8	0	5	0	3	1	0	982	0	0
	Vesta	80	0	0	0	0	0	0	0	0	0	0	0	920	0
	Voyager	0	1	0	1	0	0	0	0	0	0	0	0	0	998

between the largest and second largest classifier. Specifically, using the set of test images, we counted the number of cases observed for each combination of top classifier score and classifier gap as a percentage of top classifier score. Example results are shown as a heat map in Fig. 11 for a few of the known objects and for all of the unknown objects. The aim is to determine if an inequality constraint exists (red

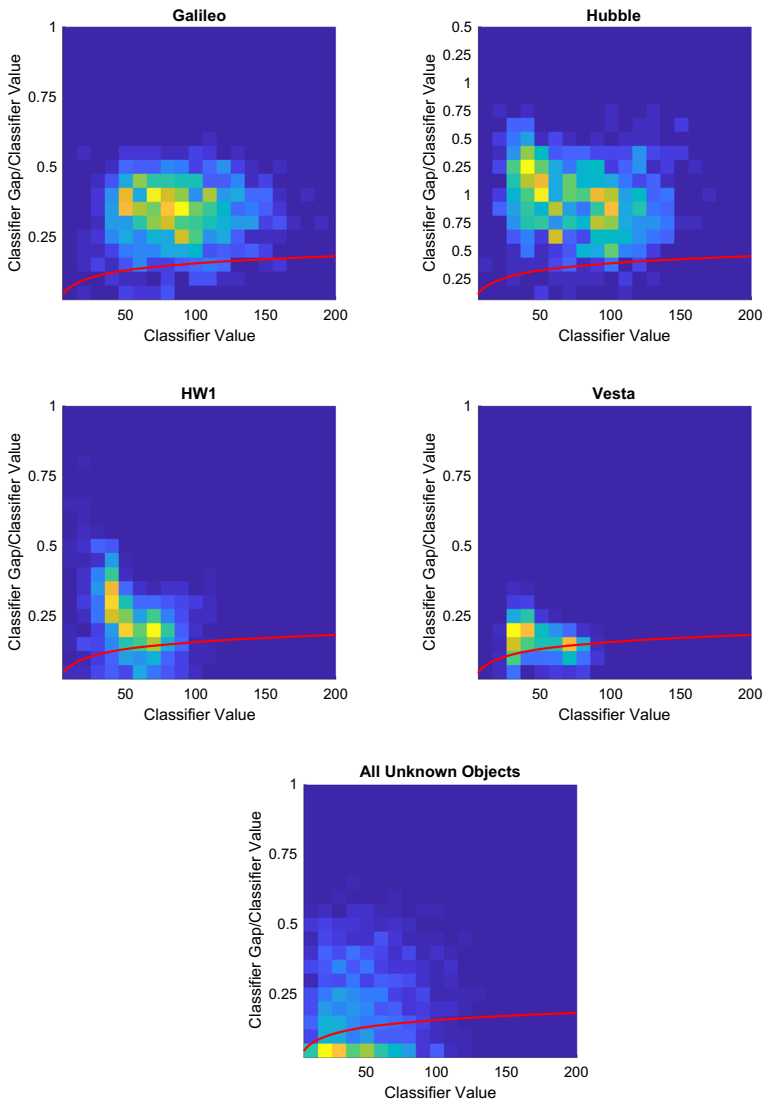


Fig. 11 Heat maps showing the number of cases observed for each combination of top classifier score and classifier gap as a percentage of top classifier score for four known objects and all of the unknown objects combined. The red line is the inequality constraint with everything below it being classified as unknown. The top two plots show known objects that are not exceedingly affected by this method of classifying unknown objects and the middle two plots show two known objects that are affected by this method

line in Fig. 11) where the scenarios falling above the constraint are known objects and scenarios falling below the constraint are unknown objects. It is immediately evident, however, that substantial overlap of the classifier activation behavior between the known objects and the unknown objects preclude the effective use of such an approach.

Regardless of these drawbacks, this method was implemented for our bin 1 convolutional neural network and the results can be seen in Table 6. Using this method most of the known object success rates stayed above 90% with one object dropping to 86%. This method correctly labeled only 30.7% of the unknown objects as unknown (with the rest of the cases incorrectly labeling the unknown objects as one of the catalog objects). Moving the inequality constraint to capture more images in the unknown object heat map may increase the success rate of the classifying unknown objects, but it also reduces the success rates for the known objects. The top classifier value and gap of an image will be different for every network. Therefore, each trained network would need to be assessed and a new constraint would need to be developed. This method's results and the inability for convolutional neural networks to provide a confidence in its classifications proves that alternative methods must be explored.

There are a variety of emerging techniques for novelty detection in deep learning that will be explored in future work, such as developing an autoencoder [11,

Table 6 Success rate (%) of network after unknown object method is implemented. This novelty detection method slightly drops the success rate of the known objects while only successfully classifying 30.7% of the unknown objects as unknown

Known object	Successful identification (%)	Unknown object	Successful classification as unknown (%)
Bennu ^c	92.4	Armadillo ^a [35]	32.5
Cassini ^c	94.6	Astronaut ^c	32.5
FUSE ^c	95.2	Bunny ^a [67]	19.0
Galileo ^c	97.8	Dawn ^c	47.0
Golevka ^c	96.6	Guitar ^b	35.0
Hubble ^c	99.3	Space Invader ^b	23.5
HW1 ^c	86.7	Apollo Lunar Module ^c	35.5
ISS ^c	96.5	Airplane ^b	20.0
Itokawa [18]	91.2	Space Shuttle ^c	28.5
Kleopatra ^c	90.6	TIE Fighter ^b	33.5
Maven ^c	93.5		
Mithra ^c	95.9		
Vesta ^c	88.7		
Voyager ^c	90.9		

^a<http://graphics.stanford.edu/data/3Dscanrep/>

^b<https://www.thingiverse.com/>

^c<https://nasa3d.arc.nasa.gov/models>

68] or generative adversarial network (GAN) [58]. Both methods require an entirely new architecture and method of training since both are considered unsupervised learning. Both the autoencoder and the GAN are used as generative models with the autoencoder detecting novelties by how poorly the network recreates the novel data. For the GAN, the adversarial segment of the network is able to detect if an object in the image is unknown to the network. These methods prove to be more successful than simply evaluating classifier activations, but a separate model must be trained to discern between novel or known objects.

Conclusion

Using a convolutional neural network architecture, we developed a technique for the identification of partially resolved space objects in images at varying distances, relative attitudes, phase angles, and defocus/blur. Three-dimensional triangular mesh models of both asteroids and spacecraft were used to develop a synthetic data sets to both train and evaluate our network architecture. The results involving crisp image training and testing showed that using multiple convolutional neural networks trained on overlapping phase angle bins provided high success rates in classifying objects within an image. The results involving blurred images showed that reasonable performance is maintained, so long as the network is trained with a mix of crisp and blurred images. A method for novelty detection was implemented with results showing a need for further exploration of the topic.

Acknowledgements This work was made possible by the U.S. Air Force Office of Scientific Research (AFOSR) Young Investigator Program (YIP) through grant FA9550-15-1-0215.

References

1. Ali, H., Awan, A.A., Khan, S., Shafique, O., ur Rahman, A., Khan, S.: Supervised classification for object identification in urban areas using satellite imagery. In: 2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), pp. 1–4. IEEE (2018). <https://doi.org/10.1109/ICOMET.2018.8346383>
2. Andreon, S., Gargiulo, G., Longo, G., Tagliaferri, R., Capuano, N.: Wide field imaging—I. Applications of neural networks to object detection and star/galaxy classification. *Monthly Notices of the Royal Astronomical Society* **319**(3), 700–716 (2000). <https://doi.org/10.1046/j.1365-8711.2000.03700.x>
3. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (surf). *Computer Vision and Image Understanding* **110**(3), 346–359 (2008). <https://doi.org/10.1016/j.cviu.2007.09.014>
4. Bosch, A., Zisserman, A., Munoz, X.: Image classification using random forests and ferns. In: 2007 IEEE 11th International Conference on Computer Vision, pp. 1–8. IEEE (2007). <https://doi.org/10.1109/ICCV.2007.4409066>
5. Chang, L., Duarte, M.M., Sucar, L.E., Morales, E.F.: A bayesian approach for object classification based on clusters of sift local features. *Expert Systems With Applications* **39**(2), 1679–1686 (2012). <https://doi.org/10.1016/j.eswa.2011.06.059>

6. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. *J. Mach. Learn. Res.* **12**, 2493–2537 (2011)
7. Dahl, G.E., Sainath, T.N., Hinton, G.E.: Improving deep neural networks for lvcsvr using rectified linear units and dropout. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 8609–8613 (2013). <https://doi.org/10.1109/ICASSP.2013.6639346>
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: a large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009, pp. 248–255. IEEE (2009). <https://doi.org/10.1109/CVPR.2009.5206848>
9. Déniz, O., Bueno, G., Salido, J., De la Torre, F.: Face recognition using histograms of oriented gradients. *Pattern Recogn. Lett.* **32**(12), 1598–1603 (2011). <https://doi.org/10.1016/j.patrec.2011.01.004>
10. Dieleman, S., Willett, K.W., Dambre, J.: Rotation-invariant convolutional neural networks for galaxy morphology prediction. *Mon. Not. R. Astron. Soc.* **450**(2), 1441–1459 (2015). <https://doi.org/10.1093/mnras/stv632>
11. Domingues, R., Michiardi, P., Zouaoui, J., Filippone, M.: Deep gaussian process autoencoders for novelty detection. *Mach. Learn.* **107**(8), 1363–1383 (2018). <https://doi.org/10.1007/s10994-018-5723-3>
12. Ertl, C., Christian, J.: Identification of partially resolved objects in space imagery with neural networks. In: AAS/AIAA Astrodynamics Specialist Conference. Snowbird, UT (2018)
13. Fix, E., Hodges, J.L. Jr.: Discriminatory analysis-nonparametric discrimination: consistency properties. Tech. rep., California Univ Berkeley (1951)
14. Franco-Lopez, H., Ek, A.R., Bauer, M.E.: Estimation and mapping of forest stand density, volume, and cover type using the k-nearest neighbors method. *Remote Sens. Environ.* **77**(3), 251–274 (2001). [https://doi.org/10.1016/S0034-4257\(01\)00209-7](https://doi.org/10.1016/S0034-4257(01)00209-7)
15. Friedman, A.M., Fan, S., Frueh, C.: Light curve inversion observability analysis. In: 2019 AAS Astrodynamics Specialist Conference (2019)
16. Fukushima, K.: Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics* **36**(4), 193–202 (1980). <https://doi.org/10.1007/BF00344251>
17. Furfaro, R., Linares, R., Reddy, V.: Space objects classification via light-curve measurements: deep convolutional neural networks and model-based transfer learning. In: AMOS Technologies Conference, Maui Economic Development Board, Kihei, Maui, HI (2018)
18. Gaskell, R., Saito, J., Ishiguro, M., Kubota, T., Hashimoto, T., Hirata, N., Abe, S., Barnouin-Jha, O., Scheeres, D.: Gaskell itokawa shape model v1.0. NASA Planetary Data System **92**, HAY-A-AMICA-5-ITOKAWASHAPE-V1.0 (2008)
19. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016). <http://www.deeplearningbook.org>
20. Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. In: Advances in Neural Information Processing Systems 28. Montreal, Canada (2015)
21. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016). <https://doi.org/10.1109/CVPR.2016.90>
22. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. arXiv:1207.0580 (2012)
23. Ho, T.K.: Random decision forests. In: Proceedings of 3rd International Conference on Document Analysis and Recognition, vol. 1, pp. 278–282. IEEE (1995). <https://doi.org/10.1109/ICDAR.1995.598994>
24. Hoffer, E., Hubara, I., Soudry, D.: Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In: Advances in Neural Information Processing Systems 30. Longbeach, CA (2017)
25. Huang, G.B., Saratchandran, P., Sundararajan, N.: A generalized growing and pruning rbf (ggap-rbf) neural network for function approximation. *IEEE Trans. Neural Netw.* **16**(1), 57–67 (2005). <https://doi.org/10.1109/TNN.2004.836241>
26. Jacobs, C., Glazebrook, K., Collett, T., More, A., McCarthy, C.: Finding strong lenses in CFHTLS using convolutional neural networks. *Mon. Not. R. Astron. Soc.* **471**(1), 167–181 (2017). <https://doi.org/10.1093/mnras/stx1492>

27. Jia, B., Pham, K.D., Blasch, E., Wang, Z., Shen, D., Chen, G.: Space object classification using deep neural networks. In: 2018 IEEE Aerospace Conference, pp. 1–8. IEEE (2018). <https://doi.org/10.1109/AERO.2018.8396567>
28. Jones, R.C.: On the point and line spread functions of photographic images. *J. Opt. Soc. Am.* **48**(12), 934–937 (1958). <https://doi.org/10.1364/JOSA.48.000934>
29. Kaasalainen, M., Torppa, J.: Optimization methods for asteroid lightcurve inversion: i. shape determination. *Icarus* **153**(1), 24–36 (2001). <https://doi.org/10.1006/icar.2001.6673>
30. Kaasalainen, M., Torppa, J., Muinonen, K.: Optimization methods for asteroid lightcurve inversion: ii. the complete inverse problem. *Icarus* **153**(1), 37–51 (2001). <https://doi.org/10.1006/icar.2001.6674>
31. Kheradpisheh, S.R., Ganjtabesh, M., Thorpe, S.J., Masquelier, T.: Stdp-based spiking deep convolutional neural networks for object recognition. *Neural Networks* **99**, 56–67 (2018). <https://doi.org/10.1016/j.neunet.2017.12.005>
32. Kim, D., Dahyot, R.: Face components detection using surf descriptors and svms. In: 2008 International Machine Vision and Image Processing Conference, pp. 51–56. IEEE (2008). <https://doi.org/10.1109/IMVIP.2008.15>
33. Kim, E.J., Brunner, R.J.: Star-galaxy classification using deep convolutional neural networks. *Mon. Not. R. Astron. Soc.*, 4463–4475. <https://doi.org/10.1093/mnras/stw2672> (2016)
34. Korytkowski, M., Rutkowski, L., Scherer, R.: Fast image classification by boosting fuzzy classifiers. *Inform. Sci.* **327**, 175–182 (2016). <https://doi.org/10.1016/j.ins.2015.08.030>
35. Krishnamurthy, V., Levoy, M.: Fitting smooth surfaces to dense polygon meshes. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '96, pp. 313–324. ACM, New York (1996). <https://doi.org/10.1145/237170.237270>
36. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) *Advances in Neural Information Processing Systems 25*, pp 1097–1105. Curran Associates, Inc. (2012)
37. LeCun, Y., Boser, B.E., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W.E., Jackel, L.D.: Handwritten digit recognition with a back-propagation network. In: *Advances in Neural Information Processing Systems*, pp. 396–404 (1990)
38. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. *Proc. IEEE* **86**(11), 2278–2324 (1998). <https://doi.org/10.1109/5.726791>
39. LeCun, Y., Cortes, C., Burges, C.: Mnist handwritten digit database. AT&T Labs [Online]. Available: <http://yann.lecun.com/exdb/mnist> **2** (2010)
40. Lin, T., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 1–1. <https://doi.org/10.1109/TPAMI.2018.2858826> (2018)
41. Linares, R., Furfaro, R.: Space object classification using deep convolutional neural networks. In: 2016 19th International Conference on Information Fusion (FUSION), pp. 1140–1146. IEEE (2016). <https://doi.org/10.1109/AERO.2016.8396567>
42. Linares, R., Jah, M.K., Crassidis, J.L., Nebelecky, C.K.: Space object shape characterization and tracking using light curve and angles data. *Journal of Guidance, Control, and Dynamics* **37**(1), 13–25 (2013). <https://doi.org/10.2514/1.62986>
43. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision* **60**(2), 91–110 (2004). <https://doi.org/10.1023/B:VISI.0000029664.99615.94>
44. Lu, H., Li, Y., Chen, M., Kim, H., Serikawa, S.: Brain intelligence: go beyond artificial intelligence. *Mobile Networks and Applications*, 1–8. <https://doi.org/10.1007/s11036-017-0932-8> (2017)
45. Lu, X.P., Huang, X.J., Ip, W.H., Hsia, C.H.: Lebedev acceleration and comparison of different photometric models in the inversion of lightcurves for asteroids. *Planet. Space Sci.* **153**, 1–10 (2018)
46. McMahon, J.W., Scheeres, D.J.: Shape estimation from lightcurves including constraints from orbit determination. In: *Advanced Maui Optical and Space Surveillance Technologies Conference* (2016)
47. Mou, L., Zhu, X.X.: RiFCN: recurrent network in fully convolutional network for semantic segmentation of high resolution remote sensing images. *CoRR arXiv:1805.02091* (2018)
48. Pal, M.: Random forest classifier for remote sensing classification. *Int. J. Remote Sens.* **26**(1), 217–222 (2005). <https://doi.org/10.1080/01431160412331269698>
49. Piccinini, P., Prati, A., Cucchiara, R.: Real-time object detection and localization with sift-based clustering. *Image Vis. Comput.* **30**(8), 573–587 (2012). <https://doi.org/10.1016/j.imavis.2012.06.004>
50. Pillai, S., Leonard, J.: Monocular slam supported object recognition. *arXiv:1506.01732* (2015)
51. Pourrahmani, M., Nayyeri, H., Cooray, A.: Lensflow: a convolutional neural network in search of strong gravitational lenses. *Astrophys. J.* **856**(1), 68 (2018). <https://doi.org/10.3847/1538-4357/aaae6a>

52. Rao, U.G., Jain, V.: Gaussian and exponential approximations of the modulation transfer function. *JOSA* **57**(9), 1159–1160 (1967)
53. Rhodes, A., Christian, J.A.: Constructing a 3D scale space from implicit surfaces for vision-based spacecraft relative navigation. In: 41st Annual AAS Guidance, Navigation, and Control Conference. Breckenridge, CO (2018)
54. Rodriguez-Galiano, V.F., Ghimire, B., Rogan, J., Chica-Olmo, M., Rigol-Sanchez, J.P.: An assessment of the effectiveness of a random forest classifier for land-cover classification. *ISPRS J. Photogramm. Remote. Sens.* **67**, 93–104 (2012). <https://doi.org/10.1016/j.isprsjprs.2011.11.002>
55. Rosten, E., Drummond, T.: Fusing points and lines for high performance tracking. In: Tenth IEEE International Conference on Computer Vision (ICCV'05), vol. 2, pp. 1508–1515. Citeseer (2005). <https://doi.org/10.1109/ICCV.2005.104>
56. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. *Nature* **323**(6088), 533–536 (1986). <https://doi.org/10.1038/323533a0>
57. Russell, H.: On the light variations of asteroids and satellites. *Astrophys. J.* **24**, 1–18 (1906). <https://doi.org/10.1086/141361>
58. Sabokrou, M., Khalooei, M., Fathy, M., Adeli, E.: Adversarially learned one-class classifier for novelty detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3379–3388 (2018). <https://doi.org/10.1109/CVPR.2018.00356>
59. Schmitt, D., McCoy, N.: Object classification and localization using surf descriptors. *CS* **229**, 1–5 (2011)
60. Schneider, S., Taylor, G.W., Kremer, S.C.: Deep learning object detection methods for ecological camera trap data. arXiv:1803.10842 (2018)
61. Schroff, F., Criminisi, A., Zisserman, A.: Object class segmentation using random forests. In: *BMVC*, pp. 1–10 (2008)
62. Sutton, R.S., Barto, A.G., Williams, R.J.: Reinforcement learning is direct adaptive optimal control. *IEEE Control. Syst. Mag.* **12**(2), 19–22 (1992). <https://doi.org/10.1109/37.126844>
63. Sze, V., Chen, Y.H., Yang, T.J., Emer, J.S.: Efficient processing of deep neural networks: a tutorial and survey. *Proc. IEEE* **105**(12), 2295–2329 (2017). <https://doi.org/10.1109/JPROC.2017.2761740>
64. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015). <https://doi.org/10.1109/CVPR.2015.7298594>
65. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016). <https://doi.org/10.1109/CVPR.2016.308>
66. Thanh Noi, P., Kappas, M.: Comparison of random forest, k-nearest neighbor, and support vector machine classifiers for land cover classification using sentinel-2 imagery. *Sensors* **18**(1), 18 (2018). <https://doi.org/10.3390/s18010018>
67. Turk, G., Levoy, M.: Zipped polygon meshes from range images. In: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '94, pp. 311–318. ACM, New York (1994). <https://doi.org/10.1145/192161.192241>
68. Vasilev, A., Golkov, V., Meissner, M., Lipp, I., Sgarlata, E., Tomassini, V., Jones, D.K., Cremers, D.: q-space novelty detection with variational autoencoders. arXiv:1806.02997 (2018)
69. Durech, J., Hanuš, J., Brož, M., Lehký, M., Behrend, R., Antonini, P., Charbonnel, S., Crippa, R., Dubreuil, P., Farroni, G., Kober, G., Lopez, A., Manzini, F., Oey, J., Poncy, R., Rinner, C., Roy, R.: Shape models of asteroids based on lightcurve observations with blueeye600 robotic observatory. *Icarus* **304**, 101–109 (2018). <https://doi.org/10.1016/j.icarus.2017.07.005>
70. Wang, N., Yeung, D.Y.: Learning a deep compact image representation for visual tracking. In: Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q. (eds.) *Advances in Neural Information Processing Systems 26*, pp 809–817. Curran Associates, Inc. (2013)
71. Wang, R., Xia, Y., Wang, G., Tian, J.: License plate localization in complex scenes based on oriented fast and rotated brief feature. *Journal of Electronic Imaging* **24**(5), 053011 (2015). <https://doi.org/10.1117/1.JEI.24.5.053011>
72. Wang, X., Han, T.X., Yan, S.: An HOG-LBP human detector with partial occlusion handling. In: 2009 IEEE 12th International Conference on Computer Vision, pp. 32–39. IEEE (2009). <https://doi.org/10.1109/ICCV.2009.5459207>
73. Widrow, B., Hoff, M.E.: Adaptive switching circuits. Tech. rep., Stanford Univ Ca Stanford Electronics Labs (1960)

74. Yu, S., De Backer, S., Scheunders, P.: Genetic feature selection combined with composite fuzzy nearest neighbor classifiers for hyperspectral satellite imagery. *Pattern Recogn. Lett.* **23**(1-3), 183–190 (2002). [https://doi.org/10.1016/S0167-8655\(01\)00118-0](https://doi.org/10.1016/S0167-8655(01)00118-0)
75. Zhang, G.P.: Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **30**(4), 451–462 (2000). <https://doi.org/10.1109/5326.897072>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.