Journal of
Applied Remote Sensing

RemoteSensing.SPIEDigitalLibrary.org

Comprehensive survey of deep learning in remote sensing: theories, tools, and challenges for the community

John E. Ball
Derek T. Anderson
Chee Seng Chan

SPIE.

John E. Ball, Derek T. Anderson, Chee Seng Chan, "Comprehensive survey of deep learning in remote sensing: theories, tools, and challenges for the community," *J. Appl. Remote Sens.* **11**(4), 042609 (2017), doi: 10.1117/1.JRS.11.042609.

REVIEW

Comprehensive survey of deep learning in remote sensing: theories, tools, and challenges for the community

John E. Ball,[a,*] Derek T. Anderson,[a] and Chee Seng Chan[b]
[a]Mississippi State University, Department of Electrical and Computer Engineering, Mississippi State, Mississippi, United States
[b]University of Malaya, Faculty of Computer Science and Information Technology, Kuala Lumpur, Malaysia

Abstract. In recent years, deep learning (DL), a rebranding of neural networks (NNs), has risen to the top in numerous areas, namely computer vision (CV), speech recognition, and natural language processing. Whereas remote sensing (RS) possesses a number of unique challenges, primarily related to sensors and applications, inevitably RS draws from many of the same theories as CV, e.g., statistics, fusion, and machine learning, to name a few. This means that the RS community should not only be aware of advancements such as DL, but also be leading researchers in this area. Herein, we provide the most comprehensive survey of state-of-the-art RS DL research. We also review recent new developments in the DL field that can be used in DL for RS. Namely, we focus on theories, tools, and challenges for the RS community. Specifically, we focus on unsolved challenges and opportunities as they relate to (i) inadequate data sets, (ii) human-understandable solutions for modeling physical phenomena, (iii) big data, (iv) nontraditional heterogeneous data sources, (v) DL architectures and learning algorithms for spectral, spatial, and temporal data, (vi) transfer learning, (vii) an improved theoretical understanding of DL systems, (viii) high barriers to entry, and (ix) training and optimizing the DL. [DOI: 10.1117/1.JRS.11.042609]

Keywords: remote sensing; deep learning; hyperspectral; multispectral; big data; computer vision.

Paper 170464SS received Jun. 3, 2017; accepted for publication Aug. 16, 2017; published online Sep. 23, 2017.

1 Introduction

In recent years, deep learning (DL) has led to leaps, versus incremental gain, in fields such as computer vision (CV), speech recognition, and natural language processing, to name a few. The irony is that DL, a surrogate for neural networks (NNs), is an age-old branch of artificial intelligence that has been resurrected due to factors such as algorithmic advancements, high-performance computing, and big data. The idea of DL is simple: the machine learns the features and is usually very good at decision making (classification) versus a human manually designing the system. The RS field draws from core theories such as physics, statistics, fusion, and machine learning, to name a few. Therefore, the RS community should be aware of and at the leading edge of advancements such as DL. The aim of this paper is to provide resources with respect to theory, tools, and challenges for the RS community. Specifically, we focus on unsolved challenges and opportunities as they relate to (i) inadequate data sets, (ii) human-understandable solutions for modeling physical phenomena, (iii) big data, (iv) nontraditional heterogeneous data sources, (v) DL architectures and learning algorithms for spectral, spatial, and temporal data, (vi) transfer learning, (vii) an improved theoretical understanding of DL systems, (viii) high barriers to entry, and (ix) training and optimizing the DL.

*Address all correspondence to: John E. Ball, E-mail: jeball@ece.msstate.edu

Herein, RS is a technological challenge where objects or scenes are analyzed by remote means. This definition includes the traditional remote sensing (RS) areas, such as satellite-based and aerial imaging. However, RS also includes nontraditional areas, such as unmanned aerial vehicles (UAVs), crowdsourcing (phone imagery, tweets, etc.), and advanced driver-assistance systems (ADAS). These types of RS offer different types of data and have different processing needs, and thus also come with new challenges to algorithms that analyze the data. The contributions of this paper are as follows:

1. *Thorough list of challenges and open problems in DL RS*. We focus on unsolved challenges and opportunities as they relate to (i) inadequate data sets, (ii) human-understandable solutions for modeling physical phenomena, (iii) big data, (iv) nontraditional heterogeneous data sources, (v) DL architectures and learning algorithms for spectral, spatial, and temporal data, (vi) transfer learning, (vii) an improved theoretical understanding of DL systems, (viii) high barriers to entry, and (ix) training and optimizing the DL. These observations are based on surveying RS DL and feature learning (FL) literature, as well as numerous RS survey papers. This topic is the majority of our paper and is discussed in Sec. 4.
2. *Thorough literature survey*. Herein, we review 205 RS application papers and 57 survey papers in RS and DL. In addition, many relevant DL papers are cited. Our work extends the previous DL survey papers[1–3] to be more comprehensive. We also cluster DL approaches into different application areas and provide detailed discussions of many relevant papers in these areas in Sec. 3.
3. *Detailed discussions of modifying DL architectures to tackle RS problems*. We highlight approaches in DL for RS, including new architectures, tools, and DL components that current RS researchers have implemented in DL. This is discussed in Sec. 4.5.
4. *Overview of DL*. For RS researchers not familiar with DL, Sec. 2 provides a high-level overview of DL and lists many good references for interested readers to pursue.
5. *DL tool list*. Tools are a major enabler of DL, and we review the more popular DL tools. We also list pros and cons of several of the most popular toolsets and provide a table summarizing the tools, with references and links (refer to Table 1). For more details, see Sec. 2.3.5.
6. *Online summaries of RS data sets and DL RS papers reviewed*. First, an extensive online table with details about each DL RS paper we reviewed: sensor modalities, a compilation of the data sets used, a summary of the main contribution, and references. Second, a data set summary for all the DL RS papers analyzed in this paper is provided online. It contains the data set name, a description, a URL (if one is available), and a list of references. Since the literature review for this paper was so extensive, these tables are too large to put in the main paper but are provided online for the readers' benefit. These tables are located at http://cs-chan.com/source/FADL/Online_Paper_Summary_Table.pdf, and http://cs-chan.com/source/FADL/Online_Dataset_Summary_Table.pdf.

This paper is organized as follows. Section 2 discusses related work in CV. This section contrasts deep and "shallow" learning, and summarizes DL architectures. The main reasons for success of DL are also discussed in this section. Section 3 provides an overview of DL in RS, highlighting DL approaches in many disparate areas of RS. Section 4 discusses the unique challenges and open issues in applying DL to RS. Conclusions and recommendations are listed in Sec. 5.

2 Related Work in CV

CV is a field of study that aims to achieve visual understanding through computer analysis of imagery. Traditional (aka, classical) approaches are sometimes referred to as "shallow" nowadays because there are typically only a few processing stages, e.g., image denoising or enhancement followed by feature extraction then classification, that connect the raw data to our final decisions. Examples of "shallow learners" include support vector machines (SVMs), Gaussian mixtures models, hidden Markov models, and conditional random fields. In contrast, DL usually

has many layers—the exact demarcation between "shallow" and "deep" learning is not a set number (akin to multi- and hyperspectral signals)—which allows a rich variety of complex, nonlinear, and hierarchical features to be learned from the data. The following sections contrast deep and shallow learning, discuss DL approaches and DL enablers, and finally discuss DL success in domains outside RS. Overall, the challenge of human-engineered solutions is the manual or experimental discovery of which feature(s) and classifier satisfy the task at hand. The challenge of DL is to define the appropriate network topology and subsequently optimizing its hyperparameters.

2.1 *Traditional Feature Learning Methods*

Traditional methods of feature extraction involve hand-coded transforms that extract information based on spatial, spectral, textural, morphological, and other cues. Examples are discussed in detail in the following references; we do not give extensive algorithmic details herein.

Cheng et al.[1] discuss traditional hand-crafted features such as the histogram of ordered gradients (HOG), which is a feature of the scale-invariant feature transform (SIFT), color histograms, local binary patterns (LBP), etc. They also discuss unsupervised FL methods, such as *k*-means clustering and sparse coding. Other good survey papers discuss hyperspectral image (HSI) feature analysis,[4] kernel-based methods,[5] statistical learning methods in HSI,[6] spectral distance functions,[7] pedestrian detection,[8] multiclassifier systems,[9] spectral–spatial classification,[10] change detection,[11,12] machine learning in RS,[13] manifold learning,[14] endmember extraction,[15] and spectral unmixing.[16–20]

Traditional FL methods can work quite well, but (1) they require a high level of expertise and very specific domain knowledge to create the hand-crafted features, (2) sometimes the proposed solutions are fragile, that is, they work well with the data being analyzed but do not perform well on data, (3) sophisticated methods may be required to properly handle irregular or complicated decision surfaces, and (4) shallow systems that learn hierarchical features can become very complex.

In contrast, DL approaches (1) learn from the data itself, which means the expertise for feature engineering is replaced (partially or completely) by the DL, (2) DL has state-of-the-art results in many domains (and these results are usually significantly better then shallow approaches), and (3) DL in some instances can outperform humans and human-coded features.

However, there are also considerations when adopting DL approaches: (1) many DL systems have a large number of parameters, and require a significant amount of training data; (2) choosing the optimal architecture and training it optimally are still open questions in the DL community; (3) there is still a steep learning curve if one wants to really understand the math and operations of the DL systems; (4) it is hard to comprehend what is going on "under the hood" of DL systems, (5) adapting very successful DL architectures to fit RS imagery analysis can be challenging.

2.2 *DL Approaches*

To date, the autoencoder (AE), convolutional neural network (CNN), deep belief networks (DBNs), and recurrent NN (RNN) have been the four mainstream DL architectures. Of these architectures, the CNN is the most popular and most published to date. The deconvolutional NN (DeconvNet)[21,22] is a relative newcomer to the DL community. The following sections discuss each of these architectures at a high level. Many references are provided for the interested reader.

2.2.1 *Autoencoder*

An AE is an NN that is used for unsupervised learning of efficient codings (from unlabeled data). The AE's codings often reveal useful features from unsupervised data. One of the first AE applications was dimensionality reduction, which is required in many RS applications. One advantage of using an AE with RS data is that the data do not need to be labeled. In an AE, reducing the size

of the adjacent layers forces the AE to learn a compact representation of the data. The AE maps the input through an encoder function f to generate an internal (latent) representation, or code, h. The AE also has a decoder function, g that maps h to the output $\hat{\mathbf{x}}$. Let an input vector to an AE be $\mathbf{x} \in \mathbb{R}^d$. In a simple one hidden layer case, the function $h = f(\mathbf{x}) = g(\mathbf{W}\mathbf{x} + \mathbf{b})$, where $\mathbf{W}$ is the learned weight matrix and $\mathbf{b}$ is a bias vector. A decoder then maps the latent representation to a reconstruction or approximation of the input via $\mathbf{x}' = f'(\mathbf{W}'\mathbf{x} + \mathbf{b}')$, where $\mathbf{W}'$ and $\mathbf{b}'$ are the decoding weight and bias, respectively. Usually, the encoding and decoding weight matrices are tied (shared), so that $\mathbf{W}' = \mathbf{W}^T$, where T is the matrix transpose operator.

In general, the AE is constrained, either through its architecture, or through a sparsity constraint (or both), to learn a useful mapping (but not the trivial identity mapping). A loss function $\mathcal{L}$ measures how close the AE can reconstruct the output: $\mathcal{L}$ is a function of $\mathbf{x}$ and $\hat{\mathbf{x}} = f'[f(\mathbf{x})]$. For example, a commonly used loss function is the mean squared error, which penalizes the approximated output from being different from the input, $\mathcal{L}(x, x') = \|\mathbf{x} - \mathbf{x}'\|^2$.

A regularization function $\Omega(h)$ can also be added to the loss function to force a more sparse solution. The regularization function can involve penalty terms for model complexity, model prior information, penalizing based on derivatives, or penalties based on some other criteria such as supervised classification results (reference Sec. 14.2 of Ref. 23). Regularization is typically used with a deep encoder, not a shallow one. Regularization encourages the AE to have other properties than just reconstructing the input, such as making the representation sparse, robust to noise, or constraining derivatives in the representation. Arpit et al.[24] show that denoising and contractive AEs that also contain activations such as sigmoid or rectified linear units (ReLUs), which are commonly found in many AEs, satisfy conditions sufficient to encourage sparsity. A common practice is to add a weighted regularizing term to the optimization function, such as the l_1 norm, $\|\mathbf{h}\|_1 = \sum_{m=1}^{K} |h[i]|$, where K is the dimensionality of $\mathbf{h}$ and λ is a term which controls how much effect the regularization has on the optimization process. This optimization can be solved using alternating optimization over $\mathbf{W}$ and $\mathbf{h}$.

A denoising autoencoder (DAE) is an AE designed to remove noise from a signal or an image. Chen et al.[25] developed an efficient DAE, which marginalizes the noise and has a computationally efficient closed-form solution. To provide robustness, the system is trained using additive Gaussian noise or binary masking noise (force some percentage of inputs to zero). Many RS applications use an AE for denoising. Figure 1(a) shows an example of an AE. The diabolo shape results in dimensionality reduction.

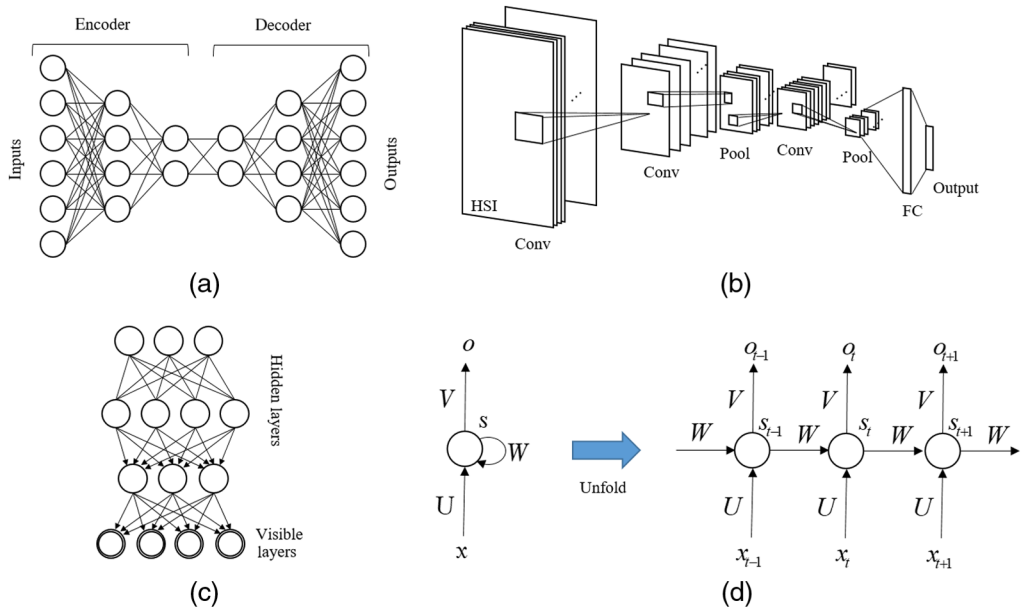

Fig. 1 Block diagrams of DL architectures: (a) AE, (b) CNN, (c) DBN, and (d) RNN.

2.2.2 *Convolutional neural network*

A CNN is a network that is loosely inspired by the human visual cortex. A typical CNN consists of multiple layers of operations such as convolution, pooling, nonlinear activation functions, and normalization, to name a few. The first part of the CNN is typically the so-called feature extractor and the latter portion of the CNN is usually a multilayer perceptron (MLP), which assigns class labels or computes probabilities of a given class being present in the input.

CNNs have dominated many perceptual tasks. Following Ujjwalkarn,[26] the image recognition community has shown keen interest in CNNs. Starting in the 1990s, LeNet was developed by LeCun et al.,[27] and was designed for reading zip codes. It generated great interest in the image processing community. In 2012, Krizhevsky et al.[28] introduced AlexNet, a deep CNN. It won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 by a significant margin. In 2013, Zeiler and Fergus[22] created ZFNet, which was AlexNet with tweaked parameters and won ILSVRC. Szegedy et al.[29] won ILSVRC with GoogLeNet in 2014, which used a much smaller number of parameters (4 million) than AlexNet (60 million). In 2015, ResNets were developed by He et al.,[30] which allowed CNNs to have very deep networks. In 2016, Huang et al.[31] published DenseNet, where each layer is directly connected to every other layer in a feedforward fashion. This architecture also eliminates the vanishing-gradient problem, allowing very deep networks. The aforementioned examples are only a few examples of CNNs.

Most layers in the CNN have parameters to estimate, e.g., the convolution filter parameters, or the scales and shifts in batch normalization. For example, a CNN that analyzes grayscale imagery employs two-dimensional (2-D) convolution filters, whereas a CNN that uses red–green–blue (RGB) imagery uses three-dimensional (3-D) filters. In general, it is easy, in theory, to support any N-dimensional signal, such as an HSI, as it really only affects the dimensionality of the first layers of filters. Through training, these filters learn to extract hierarchical features directly from the data, versus traditional machine learning approaches, that use "hand-crafted" features. Figure 1(b) shows an example CNN for HSI processing. Note that there are two convolution and pooling layers, followed by two fully connected (FC) layers.

The number of convolution filters and the filter sizes, the pooling sizes and strides, and all other layer operators and associated parameters need to be learned. In general, it has been demonstrated that the lower layers of a CNN typically learn basic features, and as one traverses the depths of the network, the features become more complex and are built up hierarchically. The FC layers are usually near the end of the CNN network, and allow complex nonlinear functions to be learned from the hierarchical outputs of the previous layers. These final layers typically output class labels, estimates of the posterior probabilities of the class label, or some other quantities such as softmax normalized values. Convolution layers perform filtering, e.g., enhancement, denoising, detection, etc., on the outputs of the previous layers. Consider convolution on a single grayscale image using a $m \times m$ mask. If the input data have dimensions of $K \times K$, then the convolution output will be $(K - m + 1) \times (K - m + 1)$, assuming a stride of one and no padding. Often, CNNs start with smaller mask sizes such as 3×3 or 5×5 and may have different sizes per layer. More information on convolution can be found in Secs. 9.1 and 9.2 of Ref. 23.

Most CNNs then apply a nonlinear activation function σ to the linear convolution result. Early CNNs used functions such as the hyperbolic tangent (tanh) or sigmoid function, but researchers later discovered that using a ReLU produced good results and was far more computationally efficient. Using the nonlinear function gives the CNN convolution layers highly expressive power. In most CNNs, there are ReLUs or parametric ReLUs (PReLUs) following the convolution layers. Given the input x, the ReLU computes the output as $\mathrm{ReLU}(\mathbf{x}) = \max(0, \mathbf{x})$. ReLU units are often used after an affine transformation of the data, such as $\mathrm{ReLU}(\mathbf{x}) = \max(0, \mathbf{w}^T\mathbf{x} + \mathbf{b})$, where $\mathbf{w}$ is a weight vector, $\mathbf{0}$ is the all zero vector, and $\mathbf{b}$ is a bias vector. For reference refer Sec. 6.3 of Goodfellow et al.[23] for more information.

For RGB imagery, 3-D convolution is used. Likewise, for N-dimensional data, a N-dimensional convolution is used. Padding can also be applied to keep information around the edge pixels in the input. Zero padding is a common practice in image processing. Another parameter for the convolution masks is the stride, or how many units are shifted in each direction between applications of the convolution mask. For a stride of one, the convolution mask shifts over

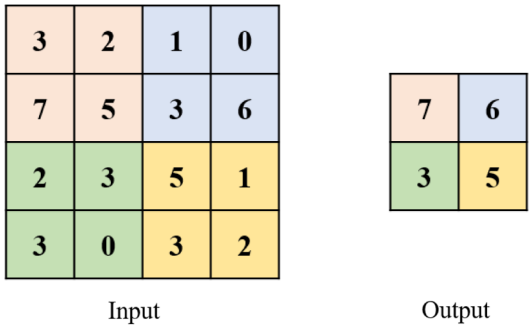

Fig. 2 CNN max pooling example with 2×2 pooling and stride of two.

one unit in each direction. A smaller stride produces larger results because more outputs are produced.

Data can be reduced by pooling, and a commonly used approach is max pooling. Consider an input with a max pooling layer with a 2×2 pooling mask and a stride of two. Figure 2 shows an example input and output. In this instance, the first output is the max of $\{3,2,7,5\}$, which is 7. The other outputs are computed in a similar manner. The max pooling layer keeps the strongest responses but destroys information on where these responses occurred in the earlier layers. Some other pooling options are average and l_2 norm pooling, which is the square root of the sum of squares of the inputs. More information on pooling can be found in Sec. 9.3 of Ref. 23.

A legitimate question is what hyperparameters to choose? Unfortunately, there is no easy answer to date, just rules of thumb and knowing from the literature what has worked for other researchers. Often experimentation and knowledge of the problem at hand are required. Some good tips can be found in Sec. 11 of Goodfellow et al.[23]

For those unfamiliar with CNNs, some good beginner tutorial and explanations can be found in Refs. 32 and 33. For those who want to use TensorFlow, helpful tutorials can be found in Ref. 32. Furthermore, a useful beginning guide to convolutional arithmetic for CNNs is Ref. 34.

2.2.3 *Deep belief network and deep Boltzmann machine*

A deep belief network (DBN) is a type (generative) of probabilistic graphical model (PGM), the marriage of probability and graph theory. Specifically, a DBN is a "deep" (large) directed acyclic graph (DAG). A deep Boltzmann machine (DBM), on the other hand, is a similar graph, but is undirected. Thus in a DBM, data can flow both ways, and training can be bottom-up or top-down. A number of well-known algorithms exist for exact and approximate inference [infer the states of unobserved (hidden) variables] and learning (learn the interactions between variables) in PGMs. A DBN can also be thought of as a type of deep NN. In Ref. 35, Hinton showed that a DBN can be viewed and trained (in a greedy manner) as a stack of simple unsupervised networks, namely restricted Boltzmann machines (RBMs), or generative AEs. To date, CNNs have demonstrated better performance on various benchmark CV data sets. However, in theory DBNs are arguably superior. CNNs possess generally a lot more "constraints." The DBN versus CNN topic is likely subject to change as better algorithms are proposed for DBN learning. Figure 1(c) depicts a DBN, which is made up of RBM layers and a visible layer.

Consider a DBM with D visible and P hidden units. The DBM energy function is $E(\mathbf{v},\mathbf{h};\theta) = -\frac{1}{2}\mathbf{v}^T\mathbf{L}\mathbf{v} - \frac{1}{2}\mathbf{h}^T\mathbf{J}\mathbf{h} - \mathbf{v}^T\mathbf{W}\mathbf{h}$, where the visible units are represented by a binary vector $\mathbf{v} \in \{0,1\}^D$, the hidden units are represented by a binary vector $\mathbf{h} \in \{0,1\}^P$, and the DBM network parameters are in $\theta = \{\mathbf{L},\mathbf{J},\mathbf{W}\}$, and $\mathbf{J}$ and $\mathbf{L}$ have zero diagonals.[36] The conditional probabilities based on the visible and hidden input states can be evaluated via

$$p(\mathbf{v},\mathbf{h};\theta) = \frac{\sum_{\mathbf{h}} \exp[-E(\mathbf{v},\mathbf{h};\theta)]}{\sum_{\mathbf{v}}\sum_{\mathbf{h}} \exp[-E(\mathbf{v},\mathbf{h};\theta)]}.$$

The numerator is an unnormalized probability and the denominator is the partition function. A DBM is typically trained layer by layer. The details of this training procedure and evaluating the DBN outputs are given in Ref. 36.

2.2.4 *Recurrent neural network*

The RNN is a network where connections form directed cycles. The RNN is primarily used for analyzing nonstationary processes such as speech and time-series analysis. The RNN has memory, so the RNN has persistence, which the AE and CNN do not possess. An RNN can be unrolled and analyzed as a series of interconnected networks that process time-series data. A major breakthrough for RNNs was the seminal work of Hochreiter and Schmidhuber,[37] the long short-term memory (LSTM) unit, which allows information to be written to a cell, output from the cell, and stored in the cell. The LSTM allows information to flow and helps counteract the vanishing/exploding gradient problems in very deep networks. Figure 1(d) shows an RNN and its unfolded version. Unlike a traditional NN, the RNN shares the network parameters $(\mathbf{U}, \mathbf{V}, \mathbf{W})$ across all steps, which makes the number of parameters to learn much smaller. In RS, RNNs have been mainly used to generate semantic image descriptions.

From Fig. 1(d), and following Ref. 38, we let $\mathbf{x}_t$ be the input to the RNN at time t and $\mathbf{s}_t$ is the hidden state at time t (e.g., the memory of the network). The hidden state is a function of the current input and the immediate past hidden state, e.g., a simple Ellman and Jordan network, $s_t = f(\mathbf{U}\mathbf{s}_t + \mathbf{W}\mathbf{s}_{t-1} + \mathbf{b})$, where the function f is usually an activation function such as tanh or a ReLU, $\mathbf{U}$ and $\mathbf{V}$ are matrices containing the network parameters, and $\mathbf{b}$ is the vector bias term. A common practice is to set the initial memory state s_0 to the zero vector. The output of our simple RNN, $\mathbf{o}_t$, is calculated as $\mathbf{o}_t = \text{softmax}(\mathbf{V}\mathbf{s}_t)$, where $\mathbf{W}$ is a matrix containing the output parameters. The network parameters $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$, and $\mathbf{b}$ must be learned by the network. Some good tutorials on RNNs are provided in Refs. 38–41.

2.2.5 *Deconvolutional neural network*

CNNs are often used for tasks, such as classification. However, a wealth of questions exists beyond classification, e.g., what are our filters really learning, how diverse are our filters, how transferable are these filters, what filters are the most active in a given (set of) image (s), where is a filter active at in a (set of) image(s), or more holistically, where in the image is our object(s) of interest (soft or hard segmentation)? In general, we want tools to help us (maybe visually) understand the networks we are working with for purposes such as "explainable AI" and/or network tuning. To this end, researchers have recently begun to explore topics such as deconvolutional CNNs.[21,22,42,43] The idea is to more-or-less "reverse" the flow of information. In most works, authors use the concept of "unpooling"—the inverse of pooling. Unpooling makes use of switch variables, which help us keep track of and place activation in layer l back to its original pooled location in layer $l-1$. Unpooling results in an enlarged, be it sparse, activation map that is then fed to some other operations.

Noh et al.[43] adapted the visual geometry group (VGG) 16-layer CNN into a deep deconvolutional network. First, they took the already trained VGG network (trained on non-RS RGB imagery) and they removed the classification layer. Next, they constructed a mirrored CNN using unpooling versus pooling—thus, the data are expanding versus shrinking in a standard pooled CNN. The first half of the network is a standard CNN and the second half is performing deconvolution. The network is then trained and it was ultimately used for semantic image segmentation.

In a different approach,[21,22] Zeiler and Fergus put forth a deconvolution CNN approach, DeconvNet, which should really be called transpose convolution, for visualizing and understanding CNNs. The crux of their approach is to use the CNN "as is." Their idea is to use the filters that were learned by the CNN to help us (visually) understand how it is working. The mathematics is relatively straight forward. First, one applies unpooling. Next, a ReLU function is applied. Last, transpose filtering (using the filter learned by the CNN) is applied. For a given image, let $\mathbf{M_i}$ be the i'th feature map at layer l. Furthermore, let $r(\hat{\mathbf{M}}_\mathbf{i})$ be the unpooled result $(\hat{\mathbf{M}}_\mathbf{i})$ that has had ReLU (r) applied to it. Last, let $\mathbf{F_k}$ be the filter that we wish to deconvolve with. The operation, technically transpose filtering, is $r(\hat{\mathbf{M}}_\mathbf{i}) * \mathbf{F_k}$, where * is the convolution operator. Zeiler used this idea to accomplish the following. First, he randomly picked filters in different layers and he identified the top nine activations (and thus images) across a validation data set. He took the location in the image (so the patch grows in size as we go deeper in the network) and he extracted its corresponding subimage so we could see what the filters are firing on. He also

plotted the transpose convolution filter result. This was an interesting visual odyssey and exploration of what hierarchical features the network was learning. However, Zeiler also used his approach to observe that some filters were dwarfed in response to others and if one normalizes filters then the behavior is less prevalent and the CNNs train (converge) faster.

In summary, interesting non-—but clearly related—RS CNN deconvolution research has been put forth to date. However, there is no systematic way yet to select and/or train the network. Furthermore, as the authors remark, their approaches respond to "parts" (the responses of what filters were learned). However, if the goal is segmentation, we may not ensure a desirable result. For example, if the CNN learns to cue off of wheels and decals on the vehicles, this will not help us to segment the entire car. Furthermore, when there are multiple instances of an object in a scene, it is not yet clear how one can associate which evidence with which instance. Regardless, this is an area of interest to the RS community.

2.3 *DL Meets the Real World*

It is important to understand the different "factors" related to the rise and success of DL. This section discusses these factors: graphical processing units (GPUs), DL NN expressiveness, big data, and tools.

2.3.1 *Graphical processing units*

GPUs are hardware devices that are optimized for fast parallel processing. GPUs enable DL by offloading computations from the computer's main processor (which is basically optimized for serial tasks) and efficiently performing the matrix-based computations at the heart of many DL algorithms. The DL community can leverage the personal computer gaming industry, which demands relatively inexpensive and powerful GPUs. A major driver of the research interest in CNNs is the ImageNet contest, which has over 1 million training images and 1000 classes.[44] DNNs are inherently parallel, use matrix operations, and use a large number of floating point operations per second. GPUs are a match because they have the same characteristics.[45] GPU speedups have been measured from 8.5 to 9[45] and even higher depending on the GPU and the code being optimized. The CNN convolution, pooling, and activation calculation operations are readily portable to GPUs.

2.3.2 *DL NN expressiveness*

Cybenko[46] proved that MLPs are universal function approximators. Specifically, Cybenko showed that a feedforward network with a single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets of $\Re^n$, with respect to relatively minimalistic assumptions regarding the activation function. However, Cybenko's proof is an existence theorem, meaning it tells us a solution exists, but it does not tell us how to design or learn such a network. The point is, NNs have an intriguing mathematical foundation that makes them attractive with respect to machine learning. Furthermore, in a theoretical work, Telgarsky[47] has shown that for NN with ReLU (1) functions with few oscillations poorly approximate functions with many oscillations and (2) functions computed by NN with few (many) layers have few (many) oscillations. Basically, a deep network allows decision functions with high oscillations. This gives evidence to show why DL performs well in classification tasks, and that shallower networks have limitations with highly oscillatory functions. Sharir and Shashua[48] showed that having overlapping local receptive fields (RFs) and more broadly denser connectivity gives an exponential increase in the expressive capacity of the NN. Liang and Srikant[49] showed that shallow networks require exponentially more neurons than a deep network to achieve the level of accuracy for function approximation.

2.3.3 *Training a DL system*

Training a DL system is not a trivial task, because: (1) many DL systems have thousands or millions of parameters, (2) RS data are limited and may not have labeled instances readily

available, (3) RS data, especially hyperspectral data, are a very large data cube and many successful DL algorithms are tuned for small RGB image patches, (4) RS data gathered via light detection and ranging (LiDAR) have insufficient DL literature (data are point clouds and not images), and (5) the best architecture is usually unknown *a priori* which means a gridded search (which can be very time consuming) or random methods such as those discussed in Ref. 50 are required for optimization. Chapter 8 of Ref. 23 discusses optimization techniques for training DL models. A thorough discussion of these techniques is beyond the scope of this paper; however, we list some common methods typically used to train DL systems.

Goodfellow et al.[23] in Sec. 8.5.4 point out that there is no current consensus on the best training/optimization algorithm. For the interested reader, the survey paper of Schaul et al.[51] provides results for many optimization algorithms over a large variety of tasks. CNNs are typically trained using stochastic gradient descent (SGD), SGD with momentum,[52] AdaGrad,[53] RMSProp,[54] and ADAM.[55] For details on the pros and cons of these algorithms, refer to Secs. 8.3 and 8.5 of Ref. 23. There are also second-order methods, and these are discussed in Sec. 8.6 of Ref. 23. A good history of DL is provided in Ref. 56, and training is discussed in Sec. 5.24. Further discussions in this paper can be found in open questions 7 (Sec. 4.7), 8 (Sec. 4.8), and 9 (Sec. 4.9).

AEs can be trained with optimization algorithms similar to a CNN. Some special AEs, such as the marginalized DAE in Ref. 25, have closed-form solutions. DBNs can be trained using greed-layer wise training, as shown in Hinton et al.[57] and Bengio et al.[58] and Salakhutdinov and Hinton[59] developed an improved pretraining method for DBNs and DBMs, by doubling or halving the weights (see the paper for more details).

Computation comparisons are complex and highly dependent on factors such as the training architecture, computer system (and GPUs), the way the architectures get data onto and off of the GPUs, the particular settings of the optimization algorithm (e.g., the mini-batch size), the learning rate, etc., and, of course, on the data itself. It is very difficult to know *a priori* what the complexities will be. This question is currently unanswered in current DL knowledge.

The survey paper by Shi et al.[60] investigates tools performance to help users of common DL tools (see Sec. 2.3.5) such as Caffe and TensorFlow to understand these tools' speed, capabilities, and limitations. They discovered that GPUs are critical to speeding up DL algorithms, whereas multicore systems do not scale linearly after about 8 cores. The GTX1080 (and now 1080Ti) GPUs performed the best among the GPUs they tested.

RNNs can be difficult to train due to the exploding gradient problem.[61] To overcome this issue, Pascanu et al.[62] developed a gradient-clipping strategy to more effectively train RNNs. Martens and Sutskever[63] developed a Hessian-free with dampening scheme RNN optimization and tested it on very challenging data sets.

Last, the survey paper of Deng[2] discusses DL architectures and gives many references for training DL systems. The survey paper of Bengio et al.[64] on unsupervised FL also discusses various learning and optimization strategies.

2.3.4 *Big data*

Every day, approximately 350 million images are uploaded to Facebook,[45] Wal-Mart collects approximately 2.5 petabytes of data per day,[45] and National Aeronautics and Space Administration (NASA) is actively streaming 1.73 gigabytes of spacecraft borne observation data for active missions alone.[65] IBM reports that 2.5 quintillion bytes of data are now generated every data, which means that "90% of the data in the world today has been created in the last two years alone."[66] The point is that an unprecedented amount of (varying quality) data exists due to technologies such as RS, smartphones, and inexpensive data storage. In times past, researchers used tens to hundreds, maybe thousands of data training samples, but nothing on the order of magnitude as today. In areas such as CV, high data volume and variety are at the heart of advancements in performance, meaning reported results are a reflection of advances in data and machine learning.

To date, a number of approaches have been explored relative to large-scale deep networks (e.g., hundreds of layers) and big data (e.g., high volume of data). For example, Raina et al.[67]

put forth central processing unit (CPU) and GPU ideas to accelerate DBNs and sparse coding. They reported a 5- to 15-fold speed-up for networks with 100 million plus parameters versus previous works that used only a few million parameters at best. On the other hand, CNNs typically use back propagation and they can be implemented either by pulling or pushing.[68] Furthermore, ideas such as circular buffers[69] and multi-GPU-based CNN architectures, e.g., Krizhevsky et al.,[28] have been put forth. Outside of hardware speedups, operators such as ReLUs have been shown to run several times faster than other common nonlinear functions. Deng et al.[70] put forth a deep stacking network (DSN) that consists of specialized NNs (called modules), each of which has a single hidden layer. Hutchinson et al.[71] put forth Tensor-DSN as an efficient and parallel extension of DSNs for CPU clusters. Furthermore, DistBelief is a library for distributed training and learning of deep networks with large models (billions of parameters) and massive sized data sets.[72] DistBelief makes use of machine clusters to manage the data and parallelism via methods such as multithreading, message passing, synchronization, and machine-to-machine communication. DistBelief uses different optimization methods, namely SGD and Sandblaster.[72] Last, but not least, there are network architectures such as highway networks, residual networks, and dense nets.[30,73–76] For example, highway networks are based on LSTM recurrent networks and they allow for the efficient training of deep networks with hundreds of layers based on gradient descent.[73–75]

Najafabadi et al.[77] discuss some aspects of DL in big data analysis and challenges associated with these efforts, including nonstationary data, high-dimensional data, and large-scale models. The survey paper of Chen and Len[78] also discusses challenges associated with big data and DL, including high volumes, variety, and velocity of data. The survey paper of Landset et al.[79] discusses open-source tools for ML with big data in Hadoop ecosystem.

2.3.5 *Tools*

Tools are also a large factor in DL research and development. Wan et al.[80] observed that DL is at the intersection of NNs, graphical modeling, optimization, pattern recognition, and signal processing, which means there is a fairly high background level required for this area. Good DL tools allow researchers and students to try some basic architectures and create new ones more efficiently.

Table 1 lists some popular DL toolkits and links to the code. Herein, we review some of the DL tools, and the tool analysis below is based on our experiences with these tools. We thank our graduate students for providing detailed feedback on these tools.

AlexNet[28] was a revolutionary paper that reintroduced the world to the results that DL can offer. AlexNet uses ReLU because it is several times faster to evaluate than the hyperbolic tangent. AlexNet revealed the importance of preprocessing by incorporating some data augmentation techniques and was able to combat overfitting using max pooling and dropout layers.

Caffe[81] was the first widely used DL toolkit. Caffe is C++ based and can be compiled on various devices, and offers command line, Python, and MATLAB interfaces. There are many useful examples provided. The cons of Caffe are that is relatively hard to install, due to lack of documentation and not being developed by an organized company. For those interested in something other than image processing (e.g., image classification, image segmentation) it is not really suitable for other areas, such as audio signal processing.

TensorFlow[82] is arguably the most popular DL tool available. Its pros are that TensorFlow (1) is relatively easy to install both with CPU and GPU version on Ubuntu (The GPU version needs CUDA and cuDNN to be installed ahead of time, which is a little complicated); (2) has most of the state-of-the-art models implemented, and while some original implementations are not implemented in TensorFlow, it is relatively easy to find a reimplementation in TensorFlow; (3) has very good documentation and regular updates; (4) supports both Python and C++ interfaces; and (5) is relatively easy to expand to other areas besides image processing, as long as you understand the tensor processing. One con of TensorFlow is that it is really restricted to Linux applications, as the windows version is barely usable.

Table 1 Some popular DL tools.

Tool and citation	Tool summary and website
AlexNet[28]	A large-scale CNN with a nonsaturating, neurons and a very efficient GPU parallel implementation of the convolution operation to make training faster. Website: http://code.google.com/p/cuda-convnet/
Caffe[81]	C++ library with Python and MATLAB® interfaces. Website: http://caffe.berkeleyvision.org/
cuda-convnet2[28]	The DL tool cuda-convnet2 is a fast C++/CUDA CNN implementation and can also model any directed acyclic graphs. Training is performed using BP. Offers faster training on Kepler-generation GPUs and multi-GPU training support. Website: https://code.google.com/p/cuda-convnet2/
gvnn[84]	The DL package gvnn is an NN library in Torch aimed toward bridging the gap between classic geometric computer vision and DL. This DL package is used for recognition, end-to-end visual odometry, depth estimation, etc. Website: https://github.com/ankurhanda/gvnn
Keras[85]	Keras is a high-level Python NN library capable of running on top of either TensorFlow or Theano and was developed with a focus on enabling fast experimentation. Keras (1) allows for easy and fast prototyping, (2) supports both convolutional networks and recurrent networks, (3) supports arbitrary connectivity schemes, and (4) runs seamlessly on CPUs and GPUs. Website: https://keras.io/ and https://github.com/fchollet/keras
MatConvNet[83]	A MATLAB® toolbox implementing CNNs with many pretrained CNNs for image classification, segmentation, etc. Website: http://www.vlfeat.org/matconvnet/
MXNet[86]	MXNet is a DL library. Features include declarative symbolic expression with imperative tensor computation and differentiation to derive gradients. MXNet runs on mobile devices to distributed GPU clusters. Website: https://github.com/dmlc/mxnet/
TensorFlow[82]	An open-source software library for tensor data flow graph computation. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile devices. Website: https://www.tensorflow.org/
Theano[87]	A Python library that allows you to define, optimize, and efficiently evaluate mathematical expressions involving multidimensional arrays. Theano features (1) tight integration with NumPy, (2) transparent use of a GPU, (3) efficient symbolic differentiation, and (4) dynamic C code generation. Website: http://deeplearning.net/software/theano
Torch[88]	Torch is an embeddable scientific computing framework with GPU optimizations, which uses the LuaJIT scripting language and a C/CUDA implementation. Torch includes (1) optimized linear algebra and numeric routines, (2) neural network and energy-based models, and (3) GPU support. Website: http://torch.ch/

MatConvNet[83] is a convenient tool, with unique abstract implementations for those very comfortable with using MATLAB. It offers many popular trained CNNs, and the data sets used to train them. It is fairly easy to install. Once the GPU setup is ready (installation of drivers + CUDA support), training with the GPU is very simple. It also offers windows support. The cons are (1) there is a substantially smaller online community compared to TensorFlow and Caffe, (2) code documentation is not very detailed and in general does not have good online tutorials besides the manual. Lack of getting started help besides a very simple example, and (3) GPU setup can be quite tedious. For windows, visual studio is required, due to restrictions on MATLAB and its mex setup, as well as Nvidia drivers and CUDA support. On Linux, one has much more freedom but must be willing to adapt to manual installations of Nvidia drivers, CUDA support, and more.

2.4 *DL in Other Domains*

DL has been used in other areas than RS, namely human behavior analysis,[89–92] speech recognition,[93–95] stereo vision,[96] robotics,[97] signal-to-text,[98–102] physics,[103,104] cancer detection,[105–107] time-series analysis,[108–110] image synthesis,[111–117] stock market analysis,[118] and security applications.[119] These diverse set of applications show the power of DL.

For the interested reader, we provide some useful survey paper references. Arel et al.[120] provide a survey paper on DL. Deng[2] provide two important reasons for DL success: (1) GPU units and (2) recent advances in DL research. They discuss generative, discriminative, and hybrid deep architectures and show there is vast room to improve the current optimization techniques in DL. Liu et al.[121] give an overview of the autoencoder, the CNN, and DL applications. Wang and Raj[122] provide a history of DL. Yu and Deng[123] provide a review of DL in signal and image processing. Comparisons are made to shallow learning, and DL advantages are given. Two good overviews of DL are the survey paper of Schmidhuber et al.[56] and the book by Goodfellow et al.[23] Zhang et al.[3] give a general framework for DL in RS, which covers four RS perspectives: (1) image processing, (2) pixel-based classification, (3) target recognition, and (4) scene understanding. In addition, they review many DL applications in RS. Cheng et al.[1] discuss both shallow and DL methods for feature extraction. Some good DL papers are the introductory DL papers of Arnold et al.[124] and Wang and Raj,[122] the DL book by Goodfellow et al.,[23] and the DL survey papers in Refs. 2, 56, 64, 77, 78, 80, 121, 122, and 125.

3 DL Approaches in RS

There are many RS tasks that use RS data, including automated target detection, pansharpening, land cover and land-use classification, time-series analysis, and change detection. Many of these tasks use shape analysis, object recognition, dimensionality reduction, image enhancement, and other techniques, which are all amenable to DL approaches. Table 2 groups DL papers reviewed

Table 2 DL paper subject areas in remote sensing.

Area	References	Area	References
3-D (depth and shape) analysis	126–134	Advanced driver-assistance systems	135–139
Animal detection	140	Anomaly detection	141
ATR	142–153	Change detection	154–158
Classification	159–208	Data fusion	209
Dimensionality reduction	210, 211	Disaster analysis/assessment	212
Environment and water analysis	213–216	Geo-information extraction	217
Human detection	218–220	Image denoising/enhancement	221, 222
Image registration	223	Land cover classification	224–228
Land use/classification	229–239	Object recognition and detection	240–250
Object tracking	251, 252	Pansharpening	253
Planetary studies	254	Plant and agricultural analysis	255–260
Road segmentation/extraction	261–267	Scene understanding	268–270
Semantic segmentation/annotation	271–283	Segmentation	284–289
Ship classification/detection	290–292	Super-resolution	293–296
Traffic flow analysis	297, 298	Underwater detection	299–302
Urban/building	303–313	Vehicle detection/recognition	314–327
Weather forecasting	328–330		

in this paper into these basic categories. From the table, it can be seen that there is a large diversity of applications, indicating that RS researchers have seen value in using DL methods in many different areas.

Due to the large number of recent RS papers, we cannot review all of the papers using DL or FL in RS applications. Instead, herein we focus on several papers in different areas of interest that offer creative solutions to problems encountered in DL and FL and should also have a wide interest to the readers. We do provide a summary of all of the DL in RS papers we reviewed online at http://cs-chan.com/source/FADL/Online_Paper_Summary_Table.pdf.

3.1 *Classification*

Classification is the task of labeling pixels (or regions in an image) into one of several classes. The DL methods outlined as follows use many forms of DL to learn features from the data itself and perform classification at state-of-the-art levels. The following discusses classification in HSI, 3-D, satellite imagery, traffic sign detection, and synthetic aperture radar (SAR).

3.1.1 *Hyperspectral image*

HSI data classification is of major importance to RS applications, so many of the DL results we reviewed were on HSI classification. HSI processing has many challenges, including high data dimensionality and usually low numbers of training samples. Chen et al.[331] propose an DBN-based HSI classification framework. The input data are converted to a one-dimensional (1-D) vector and processed via a DBN with three RBM layers, and the class labels are output from a two-layer logistic regression NN. A spatial classifier using principal component analysis (PCA) on the spectral dimension followed by 1-D flattening of a 3-D box, a three-level DBN, and two-level logistic regression classifier. A third architecture uses combinations of the 1-D spectrum and the spatial classifier architecture. He et al.[170] developed a DBN for HSI classification that does not require SGD training. Nonlinear layers in the DBN allow for the nonlinear nature of HSI data and a logistic regression classifier is used to classify the outputs of the DBN layers. A parametric depth study showed depth of nine layers produced the best results of depths from 1 to 15, and after a depth of nine, no improvement resulted by adding more layers.

Some of the HSI DL approaches use both spectral and spatial information. Ma et al.[187] created an HSI spatial updated deep AE, which integrates spatial information. Small training sets are mitigated by a collaborative, representation-based classifier and salt-and-pepper noise is mitigated by a graph-cut-based spatial regularization. Their method is more efficient than comparable kernel-based methods, and the collaborative representation-based classification makes their system relatively robust to small training sets. Yang et al.[199] use a two-channel CNN to jointly learn spectral and spatial features. Transfer learning is used when the number of training samples is limited, where low-level and midlevel features are transferred from other scenes. The network has a spectral CNN and spatial CNN, and the results are combined in three FC layers. A softmax classifier produces the final class labels. Pan et al.[193] proposed the so-called rolling guidance filter and vertex component analysis network (R-VCANet), which also attempts to solve the common problem of lack of HSI training data. The network combines spectral and spatial information. The rolling guidance filter is an edge-preserving filter used to remove noise and small details from imagery. The VCANet is a combination of vertex component analysis,[332] which is used to extract pure endmembers, and PCANet.[333] A parameter analysis of the number of training samples, rolling times, and the number and size of the convolution kernels is discussed. The system performs well even when the training ratio is only 4%. Lee and Kwon[177] designed a contextual deep fully convolutional DL network with 14 layers that jointly exploit spatial and HSI spectral features. Variable size convolutional features are used to create a spectral–spatial feature map. A feature of the architecture is the initial layers use both $[3 \times 3 \times B]$ convolutional masks to learn spatial features, and $[1 \times 1 \times B]$ for spectral features, where B is the number of spectral bands. The system is trained with a very small number of training samples (200/class).

3.1.2 *3-D*

In 3-D analysis, there are several interesting DL approaches. Chen et al.[334] used a 3-D CNN-based feature extraction model with regularization to extract effective spectral–spatial features from HSI. L_2 regularization and dropout are used to help prevent overfitting. In addition, a virtual-enhanced method imputes training samples. Three different CNN architectures are examined: (1) a 1-D using only spectral information, consisting of convolution, pooling, convolution, pooling, stacking, and logistic regression; (2) a 2-D CNN with spatial features, with 2-D convolution, pooling, 2-D convolution, pooling, stacking, and logistic regression; (3) 3-D convolution (2-D for spatial and third dimension is spectral); the organization is same as 2-D case except with 3-D convolution. The 3-D CNN achieves near-perfect classification on the data sets.

Chen et al.[209] proposed an innovative 3-D CNN to extract the spectral–spatial features of HSI data, a deep 2-D CNN to extract the elevation features of LiDAR data, and then an FC DNN to fuse the 2-D and 3-D CNN outputs. The HSI data are processed via two layers of 3-D convolution followed by pooling. The LiDAR elevation data are processed via two layers of 2-D convolution followed by pooling. The results are stacked and processed by an FC layer followed by a logistic regression layer.

Cheng et al.[243] developed a rotation-invariant CNN (RICNN), which is trained by optimizing an objective function with a regularization constraint that explicitly enforces the training feature representations before and after rotating to be mapped close to each other. New training samples are imputed by rotating the original samples by k rotation angles. The system is based on AlexNet,[28] which has five convolutional layers followed by three FC layers. The AlexNet architecture is modified by adding a rotation-invariant layer that used the output of AlexNet's FC7 layer, and replacing the 1000-way softmax classification layer with a $(C+1)$-layer softmax classifier layer. AlexNet is pretrained, then fine-tuned using the small number of HSI training samples. Haque et al.[128] developed an attention-based human body detector that leverages four-dimensional (4-D) spatiotemporal signatures and detects humans in the dark (depth images with no RGB content). Their DL system extracts voxels then encodes data using a CNN, followed by an LSTM. An action network gives the class label and a location network selects the next glimpse location. The process repeats at the next time step.

3.1.3 *Traffic sign recognition*

In the area of traffic sign recognition, a nice result came from Ciresan et al.,[138] who created a biologically plausible DNN that is based on the feline visual cortex. The network is composed of multiple columns of DNNs, coded for parallel GPU speedup. The output of the columns is averaged. It outperforms humans by a factor of two in traffic sign recognition.

3.1.4 *Satellite imagery*

In the area of satellite imagery analysis, Zhang et al.[204] proposed a gradient-boosting random convolutional network (GBRCN) to classify very high-resolution (VHR) satellite imagery. In GBRCN, a sum of functions (called boosts) is optimized. A modified multiclass softmax function is used for optimization, making the optimization task easier. SGD is used for optimization. Proposed future work was to use a variant of this method on HSI. Zhong et al.[208] use efficient small CNN kernels and a deep architecture to learn hierarchical spatial relationships in satellite imagery. A softmax classifier output class labels based on the CNN DL outputs. The CPU handles preprocessing (data splitting and normalization), while the GPU performs convolution, ReLU and pooling operations, and the CPU handles dropout and softmax classification. Networks with one to three convolution layers are analyzed, with RFs from 10×10 to 1000×1000. SGD is used for optimization. A hyperparameter analysis of the learning rate, momentum, training-to-test ratio, and number of kernels in the first convolutional layer was also performed.

3.1.5 *SAR*

In the area of SAR processing, De and Bhattacharya[305] used DL to classify urban areas, even when rotated. Rotated urban target exhibits different scattering mechanisms, and the network learns the α and γ parameters from the HH, VV, and HV bands (H, horizontal; V, vertical polarization). Bentes et al.[143] use a constant false alarm rate (CFAR) processor on SAR data followed by N AEs. The final layer associates the learned features with class labels. Geng et al.[168] used an eight-layer network with a convolutional layer to extract texture features from SAR imagery, a scale transformation layer to aggregate neighbor features, four-stacked AE (SAE) layers for feature optimization and classification, and a two-layer postprocessor. Gray-level co-occurrence matrix and Gabor features are also extracted, and average pooling is used in layer two to mitigate noise.

3.2 *Transfer Learning*

Transfer learning uses training in one image (or domain) to enable better results in another image (or domain). If the learning crosses domains, then it may be possible to use lower to midlevel features learned in the original domain as useful features in the new domain.

Marmanis et al.[276] attacked the common problem in RS of limited training data by using transfer learning across domains. They used a CNN pretrained on the ImageNet data set and extracted an initial set of representations from orthoimagery. These representations are then transferred to a CNN classifier. This paper developed a cross-domain feature fusion system. Their system has seven convolution layers followed by two long MLP layers, three convolution layers, two more large MLP layers, and finally a softmax classifier. They extract features from the last layer, since the work of Donahue et al.[335] showed that most of the discriminative information is contained in the deeper layers. In addition, they take features from the large $(1 \times 1 \times 4096)$ MLP, which is a very long vector output, and transform it into a 2-D array followed by a large convolution (91×91) mask layer. This is done because the large feature vector is a computational bottleneck, while the 2-D data can very effectively be processed via a second CNN. This approach will work if the second CNN can learn (disentangle) the information in the 2-D representation through its layers. This approach is unique and it raises some interesting questions about alternate DL architectures. This approach was also successful because the features learned by the original CNN were effective in the new image domain.

Penatti et al.[236] asked if deep features generalize from everyday objects to RS and aerial scene domains. A CNN was trained for recognizing everyday objects using ImageNet. The CNNs analyzed performed well, in areas well outside of their training. In a similar vein, Salberg[140] use CNNs pretrained on ImageNet to detect seal pups in aerial RS imagery. A linear SVM was used for classification. The system was able to detect seals with high accuracy.

3.3 *3-D Processing and Depth Estimation*

Cadena et al.[126] used multimodal AEs for RGB imagery, depth images, and semantic labels. Through the AE, the system learns a shared representation of the distinct inputs. The AEs first denoise the given inputs. Depth information is processed as inverse depth (so sky can be handled). Three different architectures are investigated. Their system was able to make a sparse depth map more dense by fusing RGB data.

Feng et al.[127] developed a content-based 3-D shape retrieval system. The system uses a low-cost 3-D sensor (e.g., Kinect or Realsense) and a database of 3-D objects. An ensemble of AEs learns compressed representations of the 3-D objects, and the AE acts as probabilistic models which output a likelihood score. A domain adaptation layer uses weakly supervised learning to learn cross-domain representations [noisy imagery and 3-D computer-aided design (CAD)]. The system uses the AE-encoded objects to reconstruct the objects, and then additional layers rank the outputs based on similarity scores. Sedaghat et al.[133] use a 3-D voxel net that predicts the object pose as well as its class label, since 3-D objects can appear very differently based on their poses. The results were tested on LiDAR data,

CAD models, and RGB plus depth (RGBD) imagery. Finally, Zelener and Stamos[336] label missing 3-D LiDAR points to enable the CNN to have higher accuracy. A major contribution of this method is creating normalized patches of low-level features from the 3-D LiDAR point cloud. The LiDAR data are divided into multiple scan lines, and positive and negative samples. Patches are randomly selected for training. A sliding block scheme is used to classify the entire image.

3.4 *Segmentation*

Segmentation means to process imagery and divide it into regions (segments) based on the content. Basaeed et al.[286] use a committee of CNNs that perform multiscale analysis on each band to estimate region boundary confidence maps, which are then interfused to produce an overall confidence map. A morphological scheme integrates these maps into a hierarchical segmentation map for the satellite imagery.

Couprie et al.[271] used a multiscale CNN to learn features directly from RGBD imagery. The image RGB channels and the depth image are transformed through a Laplacian pyramid approach, where each scale is fed to a three-stage convolutional network that creates feature maps. The feature maps of all scales are concatenated (the coarser-scale feature maps are upsampled to match the size of the finest-scale map). A parallel segmentation of the image into superpixels is computed to exploit the natural contours of the image. The final labeling is obtained by the aggregation of the classifier predictions into the superpixels.

In his master's thesis, Kaiser[274] (1) generated new ground-truth data sets for three cities consisting of VHR aerial images with ground sampling distance on the order of centimeters and corresponding pixel-wise object labels, (2) developed FC networks (FCNs) were used to perform pixel-dense semantic segmentation, (3) created two modifications of the FCN architecture were found that gave performance improvements, and (4) used transfer learning, where the FCN model was trained on huge and diverse ground-truth data of the three cities and achieved good semantic segmentations of areas not used for training.

Längkvist et al.[176] applied a CNN to orthorectified multispectral imagery (MSI) and a digital surface model of a small city for a full, fast, and accurate per-pixel classification. The predicted low-level pixel classes are then used to improve the high-level segmentation. Various design choices of the CNN architecture are evaluated and analyzed.

3.5 *Object Detection and Tracking*

Object detection and tracking is important in many RS applications. It requires understanding at a higher level than just at the pixel level. Tracking then takes the process one step further and estimates the location of the object over time.

Diao et al.[245] propose a pixel-wise DBN for object recognition. A sparse RBM is trained in an unsupervised manner. Several layers of RBM are stacked to generate a DBN. For fine-tuning, a supervised layer is attached to the top of the DBN, and the network is trained using backpropagation (BP) with a sparse penalty constraint. Ondruska and Posner[251] used RNN to track multiple objects from 2-D laser data. This system uses no hand-coded plant or sensor models (these are required in Kalman filters). Their system uses an end-to-end RNN approach that maps raw sensor data to a hidden sensor space. The system then predicts the unoccluded state from the sensor space data. The system learns directly from the data.

Schwegmann et al.[290] use a very deep highway network for ship discrimination in SAR imagery, and a three-class SAR data set is also provided. Deep networks of 2, 20, 50, and 100 layers were tested, and the 20-layer network had the best performance. Tang et al.[291] used a hybrid approach in both feature extraction and machine learning. For feature extraction, the discrete wavelet transform (DWT) LL, LH, HL, and HH (L, low frequency and H, high frequency) features from the JPEG2000 CDF9/7 encoder were used. The LL features were inputs to a stacked DAE (SDAE). The high-frequency DWT subbands LH, HL, and HH are inputs to a second SDAE. Thus, the hand-coded wavelets provide features, while the two SDAEs learn

features from the wavelet data. After initial segmentation, the segmentation area, major-to-minor axis ratio, and compactness, which are classical machine learning features, are also used to reduce false positives. The training data are normalized to zero mean and unity variance, and the wavelet features are normalized to the [0,1] range. The training batches have different class mixtures, and 20% of inputs are dropped to the SDAEs and there is a 50% dropout in the hidden units. The extreme learning machine is used to fuse the low- and high-frequency subbands. An online-sequential extreme learning machine, which is a feedforward shallow NN, is used for classification.

Two of the most interesting results were developed to handle incomplete training data and how object detectors emerge from CNN scene classifiers. Mnih and Hinton[263] developed two robust loss functions to deal with incomplete training labeling and misregistration (location of object in map) is inaccurate. An NN is used to model pixel distributions (assuming they are independent). Optimization is performed using expectation maximization. Zhou et al.[250] show that object detectors emerge from CNNs trained to perform scene classification. They demonstrated that the same CNN can perform both scene recognition and object localization in a single forward pass, without having to explicitly learn the notion of objects. Images had their edges removed such that each edge removal produces the smallest change to the classification discriminant function. This process is repeated until the image is misclassified. The final product of that analysis is a set of simplified images which still have high classification accuracies. For instance, in bedroom scenes, 87% of these contained a bed. To estimate the empirical RF, the images were replicated and random 11×11 occluded patches were overlaid. Each occluded image is input to the trained DL network, and the activation function changes are observed; a large discrepancy indicates the patch was important to the classification task. From this analysis, a discrepancy map is built for each image. As the layers get deeper in the network, the RF size gradually increases, and the activation regions are semantically meaningful. Finally, the objects that emerge in one specific layer indicated that the networks were learning object categories (dogs, humans, etc.) This work indicates there is still extensive research to be performed in this area.

3.6 *Super-Resolution*

Super-resolution analysis attempts to infer subpixel information from the data. Dong et al.[294] used a DL network that learns a mapping between the low-resolution (LR) and the high-resolution (HR) images. The CNN takes the LR image as input and the HR image as output. In this method, all layers of the DL system are jointly optimized. In a typical super-resolution pipeline with sparse dictionary learning, image patches are densely sampled from the image and encoded in a sparse dictionary. The DL system does not explicitly learn the sparse dictionaries or manifolds for modeling the image patches. The proposed system provides better results than traditional methods and has a fast online implementation. The results improve when more data are available or when deeper networks are used.

3.7 *Weather Forecasting*

Weather forecasting attempts to use physical laws combined with atmospheric measurements to predict weather patterns, precipitation, etc. The weather affects virtually every person on the planet, so it is natural that there are several RS papers using DL to improve weather forecasting. DL ability to learn from data and understand highly nonlinear behavior shows much promise in this area of RS.

Chen et al.[213] use DBNs for drought prediction. A three-step process (1) computes the standardized precipitation index (SPI), which is effectively a probability of precipitation, (2) normalizes the SPI, and (3) determines the optimal network architecture (number of hidden layers) experimentally. Firth[328] introduced a differential integration time step network composed of a traditional NN and a weighted summation layer to produce weather predictions. The NN computes the derivatives of the inputs. These elemental building blocks are used to model the various equations that govern weather. Using time-series data, forecast convolutions feed time derivative

networks which perform time integration. The output images are then fed back to the inputs at the next time step. The recurrent deep network can be unrolled. The network is trained using BP. A pipelined, parallel version is also developed for efficient computation. The model outperformed standard models. The model is efficient and works on a regional level, versus previous models that are constrained to local levels.

Kovordányi and Roy[329] used NNs in cyclone track forecasting. The system uses a multilayer NN designed to mimic portions of the human visual system to analyze National Oceanic and Atmospheric Administration's Advanced Very High-Resolution Radiometer (NOAA AVHRR) imagery. At the first network level, shape recognition focuses on narrow spatial regions, e.g., detecting small cloud segments. Image regions can be processed in parallel using a matrix feature detector architecture. Rotational variations, which are paramount in cyclone analysis, are incorporated into the architecture. Later stages combine previous activations to learn more complex and larger structures from the imagery. The output at the end of the processing system is a directional estimator of cyclone motion. The simulation tool Leabra++ (http://ccnbook.colorado.edu/) was used. This tool is designed for simulating brain-like artificial NNs (ANNs). There are a total of five layers in the system: an input layer, three processing layers, and an output layer. During training, images were divided into smaller blocks and rotated, shifted, and enlarged. During training, the network was first given inputs and allowed to settle to steady state. Weak activations were suppressed, with at most k nodes were allowed to stay active. Then the inputs and correct outputs were presented to the network, and the weights are all zeroed. The learned weights are a combination of the two schemes. Conditional PCA and contrast's Hebbian learning were used to train the network. The system was very effective if the cyclone's center varied about 6% or less of the original image size, and less effective if there was more variation.

Shi et al.[216] extended the FC LSTM (FC-LSTM) network that they call ConvLSTM, which has convolutional structures in the input-to-state and state-to-state transitions. The application is precipitation nowcasting, which takes weather data and predicts immediate future precipitation. ConvLSTM used 3-D tensors whose last two dimensions are spatial to encode spatial data into the system. An encoding LSTM compresses the input sequence into a latent tensor, while the forecasting LSTM provides the predictions.

3.8 *Automated Object and Target Detection and Identification*

Automated object and automated target detection and identification are an important RS task for military applications, border security, intrusion detection, advanced driver-assistance systems, etc. Both automated target detection and identification are hard tasks, because usually there are very few training samples for the target (but almost all samples of the training data are available as nontarget), and often there are large variations in aspect angles, lighting, etc.

Ghazi et al.[255] used DL to identify plants in photographs using transfer parameter optimization. The main contributions of this work are (1) a state-of-the-art plant detection transfer learning system and (2) an extensive study of fine-tuning, iteration size, batch size, and data imputation (rotation, translation, reflection, and scaling). It was found that transfer learning (and fine-tuning) provided better results than training from scratch. Also, if training from scratch, smaller networks performed better, probably due to smaller training data. The authors suggest using smaller networks in these cases. Performance was also directly related to the network depth. By varying the iteration sizes, it is seen that the validation accuracies rise quickly initially and then grow slowly. The networks studied are all resilient to overfitting. The batch sizes were varied, and larger batch sizes resulted in higher performance at the expense of longer training times. Data imputation also had a significant effect on performance. The number of iterations had the most effect on the output, followed by the number of patches, and the batch size had the least significant effect. There were significant differences in training times of the systems. Li et al.[141] used DL for anomaly detection. In this work, a reference image with pixel pairs (a pair of samples from the same class and a pair from different classes) is required. Using transfer learning, the system is used on another image from the same sensor. Using vicinal pixels, the algorithm recognizes central pixels as anomalies. A 16-level network

contains layers of convolution followed by ReLUs. A fully connected layer then provides output labels.

3.9 *Image Enhancement*

Image enhancement includes many areas such as pansharpening, denoising, and image registration. Image enhancement is often performed prior to feature extraction or other image processing steps. Huang et al.[253] use a modified sparse denoising autoencoder (SPDAE), denoted MSDAE, which uses the SPDAE to represent the relationship between the HR image patches as clean data to the lower spatial resolution, high spectral resolution MSI image as corrupted data. The reconstruction error drives the cost function and layer-by-layer training is used. Quan et al.[223] use DL for SAR image registration, which is in general a harder problem than RGB image registration due to high speckle noise. The RBM learns features useful for image registration, and the random sample consensus algorithm is run multiple times to reduce outlier points.

Wei et al.[221] applied a five-layer DL network to perform image quality improvement. In their approach, degraded images are modeled as downsampled images that are degraded by a blurring function and additive noise. Instead of trying to estimate the inverse function, a DL network performs feature extraction at layer 1, then the second layer learns a matrix of kernels and biases to perform nonlinear operations to layer 1 outputs. Layers 3 and 4 repeat the operations of layers 1 and 2. Finally, an output layer reconstructs the enhanced imagery. They demonstrated results with nonuniform haze removal and random amounts of Gaussian noise. Zhang et al.[222] applied DL to enhance thermal imagery, based on first compensating for the camera transfer function (small- and large-scale nonlinearities), and then super-resolution target signature enhancement via DL. Patches are extracted from LR imagery, and the DL learns feature maps from this imagery. A nonlinear mapping of these feature maps to a HR image is then learned. SGD is used to train the network.

3.10 *Change Detection*

Change detection is the process of using two registered RS images taken at different times and detecting the changes, which can be due to natural phenomenon such as drought or flooding, or due to man-made phenomenon, such as adding a new road or tearing down an old building. We note that there is a paucity of DL research into change detection. Pacifici et al.[156] used DL for change detection in VHR satellite imagery. The DL system exploits the multispectral and multitemporal nature of the imagery. Saturation is avoided by normalizing data to $[-1,1]$ range. To mitigate illumination changes, band ratios such as blue/green are used. These images are classified according to (1) man-made surfaces, (2) green vegetation, (3) bare soil and dry vegetation, and (4) water. Each image undergoes a classification, and a multitemporal operator creates a change mask. The two classification maps and the change mask are fused using an AND operator.

3.11 *Semantic Labeling*

Semantic labeling attempts to label scenes or objects semantically, such as "there is a truck next to a tree." Sherrah[280] used the recent development of fully connected convolutional neural networks (FC-CNNs), which were developed by Long et al.[337] The FC-CNN is applied to remote-sensed VHR imagery. In their network, there is no downsampling. The system labels images semantically pixel-by-pixel. Xie et al.[310] used transfer learning to avoid training issues due to scarce training data, transfer learning is used. The FC-CNN trains in daytime imagery and predicts nighttime lights. The system also can infer poverty data from the night lights, as well as delineating man-made structures, such as roads, buildings, and farmlands. The CNN was trained on ImageNet and uses the NOAA nighttime RS satellite imagery. Poverty data were derived from a living standards measurement survey in Uganda. Mini-batch gradient descent with momentum, random mirroring for data augmentation, and 50% dropout were used to help avoid overfitting. The transfer learning approach gave higher performance in accuracy, F1 scores, precision, and area under the curve.

3.12 *Dimensionality Reduction*

HSI is inherently highly dimensional, and often contains highly correlated data. Dimensionality reduction can significantly improve results in HSI processing. Ran et al.[210] split the spectrum into groups based on correlation, then apply m CNNs in parallel, one for each band group. The CNN output is concatenated and then classified via a two-layer FC-CNN. Zabalza et al.[211] used segmented SAEs that are used for dimensionality reduction. The spectral data are segmented into k regions, each of which has an SAE to reduce dimensionality. Then the features are concatenated into a reduced profile vector. The segmented regions are determined using the correlation matrix of the spectrum. In Ball et al.,[338] it was shown that band selection is task and data dependent, and often better results can be found by fusing similarity measures versus using correlation, so both of these methods could be improved using similar approaches. Dimensionality reduction is an important processing step in many classification algorithms,[339,340] pixel unmixing,[16,17,341–345] etc.

4 Unsolved Challenges and Opportunities for DL in RS

DL applied to RS has many challenges and open issues. Table 3 gives some representative DL and FL survey papers and discusses their main content. Based on these reviews and the reviews

Table 3 Representative DL and FL survey papers.

References	Paper contents
120	A survey paper on DL. Covers CNNs, DBNs, etc.
346	Brief intro to neural networks in remote sensing.
64	Overview of unsupervised FL and deep learning. Provides overview of probabilistic models (undirected graphical, RBM, AE, SAE, DAE, contractive autoencoders, manifold learning, difficulty in training deep networks, handling high-dimensional inputs, evaluating performance, etc.)
347	Examines big-data impacts on SVM machine learning.
1	Covers about 170 publications in the area of scene classification and discusses limitations of data sets and problems associated with HR imagery. They discuss limitations of hand-crafted features, such as texture descriptors, GIST, SIFT, HOG.
2	A good overview of architectures, algorithms, and applications for DL. Three important reasons for DL success are (1) GPU units, (2) recent advances in DL research, and (3) the success of DL approaches in many image processing challenges. DL is at the intersection of machine learning, neural networks, optimization, graphical modeling, pattern recognition, probability theory, and signal processing. They discuss generative, discriminative, and hybrid deep architectures. They show there is vast room to improve the current optimization techniques in DL.
348	Overview of NN in image processing.
349	Discusses trends in extreme learning machines, which are linear, single hidden layer feedforward neural networks. ELMs are comparable or better than SVMs in generalization ability. In some cases, ELMs have comparable performance to DL approaches. They generally have high generalization capability, are universal approximators, do not require iterative learning, and have a unified learning theory.
350	Provides overview of feature reduction in remote sensing imagery.
121	A survey of deep neural networks, including the AE, the CNN, and applications.
351	Survey of image classification methods in remote sensing.
352	Short survey of DL in hyperspectral remote sensing. In particular, in one study, there was a definite sweet spot shown in the DL depth.

Table 3 (*Continued*).

References	Paper contents
353	Overview of shallow HSI processing.
15	Overview of shallow endmember extraction algorithms.
56	An in-depth historical overview of DL.
122	History of DL.
354	A review of road extraction from remote sensing imagery.
123	A review of DL in signal and image processing. Comparisons are made to shallow learning, and DL advantages are given.
3	Provides a general framework for DL in remote sensing. Covers four RS perspectives: (1) image processing, (2) pixel-based classification, (3) target recognition, and (4) scene understanding.

of many survey papers in RS, we have identified the following major open issues in DL in RS. Herein, we focus on unsolved challenges and opportunities as it relates to

i. inadequate data sets (Sec. 4.1),
ii. human-understandable solutions for modeling physical phenomena (Sec. 4.2),
iii. big data (Sec. 4.3),
iv. nontraditional heterogeneous data sources (Sec. 4.4),
v. DL architectures and learning algorithms for spectral, spatial, and temporal data (Sec. 4.5),
vi. transfer learning (Sec. 4.6),
vii. an improved theoretical understanding of DL systems (Sec. 4.7),
viii. high barriers to entry (Sec. 4.8), and
ix. training and optimizing the DL (Sec. 4.9).

4.1 *Inadequate Data Sets*

4.1.1 *Open question #1a: how can DL systems work well with limited data sets?*

There are two main issues with most current RS data sets. Table 4 provides a summary of the more common open-source data sets for the DL papers using HSI data. Many of these papers used custom data sets, and these are not reported. Table 4 shows that the most commonly used data sets were Indian Pines, Pavia University, Pavia City Center, and Salinas.

A detailed online table (too large to put in this paper) is provided, which lists each paper cited in Table 2. For each paper, a summary of the contributions is given, the data sets used are listed, and the papers are categorized in areas (e.g., HSI/MSI, SAR, 3-D, etc.). The interested reader can find this table at http://cs-chan.com/source/FADL/Online_Paper_Summary_Table.pdf. In addition, a second online table lists details about data sets from the papers we reviewed. It is located at http://cs-chan.com/source/FADL/Online_Dataset_Summary_Table.pdf

Although these are all good data sets, the accuracies from many of the DL papers are nearly saturated. This is shown clearly in Table 5. Table 5 shows results for the HSI DL papers against the commonly used data sets Indian Pines, Kennedy Space Center, Pavia City Center, Pavia University, Salinas, and the Washington DC Mall. First, overall accuracy (OA) results from different papers must be taken with a grain of salt, since (1) the number of training samples per class can differ for each paper, (2) the number of testing samples can also differ, (3) classes with few relative training samples can even have 0% OA, and if there is a large number of test samples of the other classes, the final overall accuracies can still be high. Nevertheless, it is clear from

Table 4 HSI Data set usage.

Data set and reference	Number of uses
IEEE GRSS 2013 Data Fusion Contest[355]	4
IEEE GRSS 2015 Data Fusion Contest[355]	1
IEEE GRSS 2016 Data Fusion Contest[355]	2
Indian Pines[356]	27
Kennedy Space Center[357]	8
Pavia City Center[358]	13
Pavia University[358]	19
Salinas[359]	11
Washington DC Mall[360]	2

Table 5 HSI OA results in percent.

References	IP	KSC	PaCC	PaU	Sal	DCM
284	93.4	—	—	—	—	—
160	98.0	98.0	—	98.4	—	95.4
161	97.6	—	—	—	—	—
162	97.6	—	—	—	—	—
362	—	98.8	98.5	—	—	—
331	96.0	—	**99.1**	—	—	—
167	—	—	—	94.3	—	—
209	89.6	—	—	87.1	—	—
170	—	96.6	—	—	—	—
172	90.2	—	—	92.6	92.6	—
174	—	84.2	—	—	—	—
177	92.1	—	—	94.0	—	—
178	—	—	—	**99.9**	—	—
179	96.3	—	—	—	—	—
181	94.3	—	—	96.5	94.8	—
182	97.6	—	—	**99.4**	98.8	—
183	—	96.0	85.6	—	—	—
185	91.9	—	**99.8**	96.7	95.5	—
186	—	—	94.0	93.5	—	—
233	86.5	—	—	82.6	—	—
187	**99.2**	—	**99.9**	—	—	—

Table 5 (*Continued*).

References	IP	KSC	PaCC	PaU	Sal	DCM
188	—	—	96.0	—	—	83.8
228	98.9	—	**99.9**	—	**99.5**	—
190	95.7	—	—	**99.6**	97.4	—
191	96.8	—	—	—	—	—
234	79.3		—	—	—	—
193	97.9	97.9	—	96.8	—	—
197	—	80.5	—	—	—	—
210	93.1	—	—	95.6	—	—
363	96.6	—	—	—	—	—
238	73.0	89.0	—	—	—	—
198	93.1	—	—	90.4	**99.4**	—
199	95.6	—	—	—	—	—
201	—	—	—	67.9	85.2	—
202	—	—	95.2	—	—	—
211	82.1	—	97.4	—	—	—
205	**99.7**	—	—	98.8	—	—
206	—	—	**99.7**	96.8	—	—

Note: Results higher than 99% are in bold.
Note: IP, Indian Pines, KSC, Kennedy Space Center, PaCC, Pavia City Center, Pau, Pavia University, Sal, Salinas, and DCM, Washington DC Mall.

examination of the table that the Indian Pines, Pavia City Center, Pavia University, and Salinas data sets are basically saturated.

In general, it is good to compare new methods with commonly used data sets, new and challenging data sets are required. Cheng et al.[1,361] point out that many existing RS data sets lack image variation, diversity, and have a small number of classes, and some data sets are also saturating with accuracy. They created a large-scale benchmark data set, "NWPU-RESISC45," which attempts to address all of these issues, and made it available to the RS community. The RS community can also benefit from a common practice in the CV community: publishing both data sets and algorithms online, allowing for more comparisons. A typical RS paper may only test their algorithm on two or three images and against only a few other methods. In the CV community, papers usually compare against a large amount of other methods and with many data sets, which may provide more insight about the proposed solution and how it compares with previous work.

An extensive table of all the data sets used in the papers reviewed for this survey paper is made available online, because it is too large to include in this paper. The table is available at http://cs-chan.com/source/FADL/Online_Paper_Summary_Table.pdf. The table lists the data set name, briefly describes the data sets, provides a URL (if one is available), and a reference. Hopefully, this table will assist researchers who are looking for publicly available data sets.

4.1.2 *Open question #1b: how can DL systems work well with limited training data?*

The second issue is that most RS data have a very small amount of training data available. Ironically, in the CV community, DL has an insatiable hunger for larger and larger data sets

(millions or tens of millions of training images), while in the RS field, there is also a large amount of imagery; however, there is usually only a small amount with labeled training samples. RS training data are expensive, error-prone, and usually require some expert interpretation, which is typically expensive (in terms of time, effort involved, and money) and often requires large amounts of field work and many hours or days postprocessing the data. Many DL systems, especially those with large numbers of parameters, require large amounts of training data, or else they can easily overtrain and not generalize well. This problem has also plagued other shallow systems as well, such as SVMs.

Approaches used to mitigate small training samples are (1) transfer learning, where one trains on other imagery to obtain low- to midlevel features which can still be used, or on other images from the same sensor—transfer learning is discussed in Sec. 4.6; (2) data augmentation, including affine transformations, rotations, small patch removal, etc.; (3) using ancillary data, such as data from other sensor modalities [e.g., LiDAR, digital elevation models (DEMs), etc.]; and (4) unsupervised training, where training labels are not required, e.g., AEs and SAEs. SAEs that have a diabolo shape will force the AE network to learn a low-dimensional representation.

Ma et al.[187] used a DAE and employed a collaborative representation-based classification, where each test sample can be linearly represented by the training samples in the same class with the minimum residual. In classification, features of each sample are approximated with a linear combination of features of all training sample within each class, and the label can be derived according to the class that best approximates the test features. Interested readers are referred to Refs. 46–48 in Ref. 187 for more information on collaborative representation. Tao et al.[364] used a stacked sparse AE (SSAE) that was shown to be very generalizable and performed well in cases when there were limited training samples. Ghamisi et al.[224] use Darwinian particle swarm optimization in conjunction with CNNs to select an optimal band set for classifying HSI data. By reducing the input dimensionality, fewer training samples are required. Yang et al.[199] used dual CNNs and transfer learning to improve performance. In this method, the lower and middle layers can be trained on other scenes and train the top layers on the limited training samples. Ma et al.[188] imposed a relative distance prior on the SAE DL network to deal with training instabilities. This approach extends the SAE by adding the new distance prior term and corresponding SGD optimization. LeCun[365] reviews a number of unsupervised learning algorithms using AEs, which can possibly aid when training data are minimal. Pal[366] reviews kernel methods in RS and argues that SVMs are a good choice when there are a small number of training samples. Petersson et al.[352] suggest using SAEs to handle small training samples in HSI processing.

4.2 *Human-Understandable Solutions for Modeling Physical Phenomena*

4.2.1 *Open question #2a: how can DL improve model-based RS?*

Many RS applications depend on models (e.g., a model of crop output given rain, fertilizer and soil nitrogen content, and time of year), many of which are very complicated and often highly nonlinear. Model outputs can be very inaccurate if the models do not adequately capture the true input data and properly handle the intricate interrelationships among input variables.

Abdel-Rahman and Ahmed[367] pointed out that more accurate estimation of nitrogen content and water availability can aid biophysical parameter estimation for improving plant yield models. Ali et al.[368] examine biomass estimation, which is a nonlinear and highly complex problem. The retrieval problem is ill-posed, and the electromagnetic response is the complex result of many contributions. The data pixels are usually mixed, making this a hard problem. ANNs and support vector regression have shown good results. They anticipate that DL models can provide good results. Both Adam et al.[369] and Ozesmi and Bauer[370] agree that there is need for improvement in wetland vegetation mapping. Wetland species are hard to detect and identify compared to terrestrial plants. Hyperspectral sensing with narrow bandwidths in frequency can aid. Pixel unmixing is important since canopy spectra are similar and combine with underlying hydrologic regime and atmospheric vapor. Vegetation spectra highly correlated among species,

making separation difficult. Dorigo et al.[371] analyzed inversion-based models for plant analysis, which is inherently an ill-posed and hard task. They found that using artificial neural network (ANN) inversion techniques have shown good results. DL may be able to help improve results. Canopy reflections are governed by large number of canopy elements interacting and by external factors. Since DL networks can learn very complex nonlinear systems, it seems like there is much room for improvement in applying DL models. DBNs or other DL systems seem like a natural fit for these types of problems.

Kuenzer et al.[372] and Wang et al.[373] assess biodiversity modeling. Biodiversity occurs at all levels from molecular to individual animals, to ecosystem, and to global. This requires a large variety of sensors and analysis at multiple scales. However, a main challenge is low temporal resolution. There needs to be a focus beyond just pixel-level processing and using spatial patterns and objects. DL systems have been shown to learn hierarchical features, with smaller scale features learned at the beginning of the network, and more complex and abstract features learned in the deeper portions.

4.2.2 *Open question #2b: what tools and techniques are required to "understand" how the DL works?*

It is also worth mentioning that many of these applications involve biological and scientific end-users, who will definitely want to understand how the DL systems work. For instance, modeling some biological processes using a linear model is easily understood—both the mathematical model and the statistics resulting from estimating the model parameters are well understood by scientists and biologists. However, a DL system can be so large and complex as to defy analysis. We note that this is not specific to RS, but a general problem in the broader DL community.

The DL system is seen by many researchers, especially scientists and RS end-users, as a black box that is hard to understand what is happening "under the hood." Egmont-Petersen et al.[348] and Fassnacht et al.[374] both state that disadvantages of NNs are understanding what they are actually doing, which can be difficult to understand. In many RS applications, just making a decision is not enough; people need to understand how reliable the decision is and how the system arrived at that decision. Ali et al.[368] also echo this view in their review paper on improving biomass estimation. Visualization tools, which show the convolutional filters, learning rates, and tools with deconvolution capabilities to localize the convolutional firings, are all helpful.[22,375–378] Visualization of what the DL is actually learning is an open area of research. Tools and techniques capable of visualizing what the network is learning and measures of how robust the network is (estimating how well it may generalize) would be of great benefit to the RS community (and the general DL community).

4.3 *Big Data*

4.3.1 *Open question #3: what happens when DL meets big data?*

As already discussed in Sec. 2.3.4, a number of mathematics, algorithms, and hardware have been put forth to date relative to large-scale DL networks and DL in big data. However, this challenge is not close to being solved. Most approaches to date have focused on big data challenges in RGB or RGBD data for tasks, such as face and object detection or speech. With respect to RS, we have many of the same problems as CV, but there are unique challenges related to different sensors and data. First, we can break big data into its different so-called "parts," e.g., volume, variety, and velocity. With respect to DBNs, CNNs, AEs, etc., we are primarily concerned with creating new robust and distributed mathematics, algorithms, and hardware that can ingest massive streams of large, missing, noisy data from different sources, such as sensors, humans, and machines. This means being able to combine image stills, video, audio, text, etc., with symbolic and semantic variations. Furthermore, we require real-time evaluation and possibly online learning. As big data in DL are large topic, we restrict our focus herein to factors that are unique to RS.

The first factor that we focus on is high spatial and more-so spectral dimensionality. Traditional DLs operate on relatively small grayscale or RGB imagery. However, SAR imagery has challenges due to noise, and MSI and HSI can have from four to hundreds to possibly thousands of channels. As Arel et al.[120] pointed out, a very difficult question is how well DL architectures scale with dimensionality. To date, preliminary research has tried to combat dimensionality by applying dimensionality reduction or feature selection prior to DL, e.g., Benediktsson et al.[379] reference a different band selection, grouping, feature extraction, and subspace identification in HSI RS.

Ironically, most RS areas suffer from a lack of training data. Whereas they may have massive amounts of temporal and spatial data, there may not be the seasonal variations, times of day, object variation (e.g., plants, crops, etc.), and other factors that ultimately lead to sufficient variety needed to train a DL model. For example, most online hyperspectral data sets have little-to-no variety and it is questionable about what they are, and really can at that, learn. In stark contrast, most DL systems in CV use very large training sets, e.g., millions or billions of faces in different illuminations, poses, inner class variations, etc. Unless the RS DL applies a method such as transfer learning, DL in RS often has very limited training data. For example, Ma et al.[188] tried to address this challenge by developing a new prior to deal with the instability of parameter estimation for HSI classification with small training samples. The SAE is modified by adding the relative distance prior in the fine-tuning process to cluster the samples with the same label and separate the ones with different labels. Instead of minimizing classification error, this network enforces intraclass compactness and attempts to increase interclass discrepancy.

Part of this open question is also an open question in the DL community in general, that is, what is the best training method? There is currently no consensus, and many times people use whatever method is easy to use based on their DL software. Packages, such as TensorFlow, offer many different methods, so the user can try them and experiment. Significantly different results can be found running the same algorithms with different data on different machines (especially machines with GPUs versus machines with no GPUs, etc.).

4.4 *Nontraditional Heterogeneous Data Sources*

4.4.1 *Open question #4a: how can DL work with nontraditional data sources?*

Nontraditional data sources, such a Twitter and YouTube, offer data that can be useful to RS. Analyzing these types of data will probably never replace traditional RS methods, but usually offer benefits to augment RS data, or provide quality real-time data before RS methods, which usually take longer to execute.

Fohringer et al.[380] used information extracted from social media photos to enhance RS data for flood assessments. They found one major challenge was filtering posts to a manageable amount of relevant ones to further assess. The data from Twitter and Flickr proved useful for flood depth estimation prior to RS-based methods, which typically take 24 to 28 h. Frias-Martinez and Frias-Martinez[381] take advantage of large amounts of geolocated content in social media by analyzing tweets as a complimentary source of data for urban land-use planning. Data from Manhattan (New York), London (UK), and Madrid (Spain) were analyzed using a self-organizing map[382] followed by a Voronoi tessellation. Middleton et al.[383] match geolocated tweets and created real-time crisis maps via statistical analysis, which are compared to the National Geospatial Agency postevent impact assessments. A major issue is that only about 1% of tweets contain geolocation data. These tweets usually follow a pattern of a small number of first-hand reports and many retweets and comments. High-precision results were obtained. Singh[384] aggregates user's social interest about any particular theme from any particular location into so-called "social pixels," which are amenable to media processing techniques (e.g., segmentation and convolution), which allow semantic information to be derived. They also developed a declarative operator set to allow queries to visualize, characterize, and analyze social media data. Their approach would be a promising front-end to any social media analysis system. In the survey paper of Sui and Goodchild,[385] the convergence of geographic information system (GIS) and social media is examined. They observed that GIS has moved from software helping

a user at a desk to a means of communicating earth surface data to the masses (e.g., OpenStreetMap, Google Maps, etc.).

In all of the aforementioned methods, DL can play a significant role of parsing data, analyzing data, and estimating results from the data. It seems that social media is not going away, and data from social media can often be used to augment RS data in many applications. Thus, the question is what work awaits the researcher in the area of using DL to combine nontraditional data sources with RS?

4.4.2 *Open question #4b: how does DL ingest heterogeneous data?*

Fusion can take place at numerous so-called "levels," including signal, feature, algorithm, and decision. For example, signal in–signal out (SISO) is where multiple signals are used to produce a signal out. For $\Re$-valued signal data, a common example is the trivial concatenation of their underlying vectorial data, i.e., $X = \{\hat{x}_1, \ldots, \hat{x}_N\}$ becomes $[\hat{x}_1 \hat{x}_2 \ldots \hat{x}_N]$ of length $|\hat{x}_1| + \ldots + |\hat{x}_N|$. Feature in–feature out (FIFO), which is often related to if not the same as SISO, is where multiple features are combined, e.g., an HOG and an LBP, and the result is a new feature. One example is multiple kernel learning (MKL), e.g., ℓ-p norm genetic algorithm MKL (GAMKLp).[386] Typically, the input is N $\Re$-valued Cartesian spaces, and the result is a Cartesian space. Most often, one engages in MKL to search for space in which pattern obeys some property, e.g., they are nicely linearly separable and a machine learning tool, such as a SVM, can be employed. On the other hand, decision in–decision out (DIDO), e.g., the Choquet integral (ChI), is often used for the fusion of input from decision makers, e.g., human experts, algorithms, classifiers, etc.[387] Technically speaking, a CNN is typically a signal in–decision out (SIDO) or feature in–decision out (FIDO) system. Internally, the FL part of the CNN is an SIFO or FIFO and the classifier is an FIDO. To date, most DL approaches have "fused" via (1) concatenation of $\Re$-valued input data (SISO or FIFO) relative to a single DL, (2) each source has its own DL, minus classification, that is later combined into a single DL, or (3) multiple DLs are used, one for each source, and their results are once again concatenated and subjected to the classifier (either an MLP, SVM, or other classifier).

Herein, we highlight the challenges of syntactic and semantic fusion. Most DL approaches to date syntactically have addressed how N things, which are typically homogeneous mathematically, can be ingested by a DL. However, the more difficult challenge is semantically how should these sources be combined, e.g., what is a proper architecture, what is learned (can we understand it), and why should we trust the solution? This is of particular importance to numerous challenges in RS that require a physically meaningful/grounded solution, e.g., model-based approaches. The most typical example of fusion in RS is the combining of data from two (or more) sensors. Whereas there may be semantic variation but little-to-no semantic variation, e.g., both are possibly $\Re$-valued vector data, the reality is most sensors record objective evidence about our universe. However, if human information (e.g., linguistic or text) is involved or algorithmic outputs (e.g., binary decisions, labels/symbols, probabilities, etc.), fusion becomes increasingly more difficult syntactically and semantically. Many theoretical (mathematical and philosophical) investigations, which are beyond the scope of this work, have concerned themselves with how to meaningfully combine objective versus subjective information, qualitative versus quantitative information, and mixed uncertainty information (e.g., beliefs, evidence, probabilities, etc.). It is a naive and dangerous belief that one can simply just "cram" data/information into a DL and get a meaningful result. How is fusion occurring? Where is it occurring? Fusion is further compounded if one is using uncertain information, e.g., probabilistic, possibilities, or other interval or distribution-based input. The point is, heterogeneous, be it mathematical representation, associated uncertainty, etc., is a real and serious challenge. If the DL community wishes to fuse multiple inputs or sources (humans, sensors, and algorithms), then DL must theoretically rise to the occasion to ensure that the proposed DL architectures be able to answer the question of what is the DL system really learning, and is it meaningful?

Two preliminary examples of DL works employing multisensor fusion include fusion of (1) HSI with LiDAR[388] (two sensors yielding objective data) and (2) text with imagery or

video[389] (thus high-level human information sensor data). The point is, the question remains, how can/does DL fuse data/information arising from one or more sources?

4.5 *DL Architectures and Learning Algorithms for Spectral, Spatial, and Temporal Data*

4.5.1 *DL Open question #5: what architectural extensions will DL systems require in order to tackle complicated RS problems?*

Current DL architectures, components (e.g., convolution), and optimization techniques may not be adequate to solve complex RS problems. In many cases, researchers have developed network architectures, new layer structures with their associated SGD or BP equations for training, or new combinations of multiple DL networks. This problem is also an open issue in the broader CV community. This question is at the heart of DL research. Other questions related to the open issues are:

- What architecture should be used?
- How deep should a DL system be, and what architectural elements will allow it to work at that depth?
- What architectural extensions (new components) are required to solve this problem?
- What training methods are required to solve this problem?

We examine several general areas where DL systems have evolved to handle RS data: (i) multisensor processing, (ii) using multiple DL systems, (iii) rotation and displacement-invariant DL systems, (iv) new DL architectures, (v) SAR, (vi) ocean and atmospheric processing, (vii) 3-D processing, (viii) spectral–spatial processing, and (ix) multitemporal analysis. Furthermore, we examine some specific RS applications noted in several RS survey papers as areas that DL can benefit: (a) oil spill detection, (b) pedestrian detection, (c) urban structure detection, (d) pixel unmixing, and (e) road extraction. This is by no means an exhaustive list, but meant to highlight some of the important areas.

Multisensor processing. Chen et al.[209] use two deep networks, one analyzing HSI pixel neighbors (spatial data) and the other LiDAR data. The outputs are stacked and an FC and logistic regression lay provides outputs. Huang et al.[253] use a modified sparse DAE (MSDAE) to train the relationship between HR and LR image patches. The stacked MSDAE (S-MSDAE) is used to pretrain a DNN. The HR MSI image is then reconstructed from the observed LR MSI image using the trained DNN.

Multi-DL system. In certain problems, multiple DL systems can provide significant benefit. Chen et al.[315] use parallel DNNs with no cross-connections to both speed up processing and provide good results in vehicle detection from satellite imagery. Ciresan et al.[138] use multiple parallel DNNs that are averaged for image classification. Firth[328] use 186 RNNs to perform accurate weather prediction. Hou et al.[171] use RBMs to train from polarimetric SAR data, and a three-layer DBN is used for classification. Kira et al.[390] used stereo-imaging for robotic human detection, using a CNN which was trained on appearance and stereo disparity-based features, and a second CNN, which is used for long-range detection. Marmanis et al.[277] used an ensemble of CNNs to segment VHR aerial imagery using an FCN to perform pixel-based classification. They trained multiple networks with different initializations and average the ensemble results. The authors also found errors in the data set, Vaihingen.[391]

Rotation- and displacement-invariant systems. Some RS problems require systems that are rotation and displacement-invariant. CNNs have some robustness to translation, but not in general to rotations. Cheng et al.[243] incorporated a rotation-invariant layer into a DL CNN architecture to detect objects in satellite imagery. Du et al.[146] developed a displacement- and

rotation-insensitive deep CNN for SAR automated target recognition (ATR) processing that is trained by augmented data set and specialized training procedure.

DL architectures. Some problems in RS require DL architectures. Dong et al.[294] use a CNN that takes the LR image and outputs the HR image. He et al.[170] proposed a DSN for HSI classification that uses nonlinear activations in the hidden layers and does not require SGD for training. Kontschieder et al.[175] developed deep neural decision forests, which uses a stochastic and differentiable decision tree model that steers the representation learning usually conducted in the initial layers of a deep CNN. Lee and Kwon[177] analyze HSI by applying multiple local 3-D convolutional filters of different sizes jointly exploiting spatial and spectral features, followed by a fully convolutional layers to predict pixel classes. Zhang et al.[204] propose GBRCN to classify VHR satellite imagery. Ouyang and Wang[219] developed a probabilistic parts-detector-based model to robustly handle human detection with occlusions are large deformations using a discriminative RBM to learn the visibility relationship among overlapping parts. The RBM has three layers that handle different size parts. Their results can possibly be improved by adding additional rotation invariance.

DL SAR architectures. SAR imagery has unique challenges due to noise and the grainy nature of the images. Geng et al.[168] developed a deep convolutional AE, which is a combination of a CNN, AE, classification, and postprocessing layers to classify HR SAR images. Hou et al.[171] developed a polarimetric SAR DBN. Filters are extracted from the RBMs, and a final three-layer DBN performs classification. Liu et al.[227] use a deep sparse filtering network to classify terrain using polarimetric SAR data. The proposed network is based on sparse filtering,[392] and the proposed network performs a minimization on the output l_1 norm to enforce sparsity. Qin et al.[196] performed object-oriented classification of polarimetric SAR data using an RBM and built an adaptive boosting framework (AdaBoost[393]) vice a stacked DBN in order to handle small training data. They also put forth the RBM-AdaBoost algorithm. Schwegmann et al.[290] used a very deep highway network configuration as a ship discrimination stage for SAR ship detection. They also presented a three-class SAR data set that allows for more meaningful analysis of ship discrimination performances. Zhou et al.[394] proposed a three-class change detection approach for multitemporal SAR images using an RBM. These images either increases or decreases in the backscattering values for changes, so the proposed approach classifies the changed areas into the positive and negative change classes, or no change if none is detected.

Oceanic and atmospheric studies. Oceanic and atmospheric studies present unique challenges to DL systems that require developments. Ducournau and Fablet[295] developed a CNN architecture, which analyzes sea surface temperature fields and provides a significant gain in terms of peak signal-to-noise ratio compared to classical downscaling techniques. Shi et al.[216] extended the FC-LSTM network that they call ConvLSTM, which has convolutional structures in the input-to-state and state-to-state transitions for precipitation nowcasting.

3-D processing. Guan et al.[256] use voxel-based filtering to remove ground points from LiDAR data, then a DL architecture generates high-level features from the trees 3-D geometric structure. Haque et al.[128] use both of CNN and RNN to process 4-D spatiotemporal signatures to identify humans in the dark.

Spectral–spatial HSI processing. HSI processing can be improved by fusion of spectral and spatial information. Ma et al.[187] propose a spatial updated deep AE which adds a similarity regularization term to the energy function to enforce spectral similarity. The regularization term is a cosine similarity term (basically the spectral angle mapper) between the edges of a graph, where the nodes are samples, which enforces keeping the sample correlations. Ran et al.[210]

classify HSI data by learning multiple CNN-based submodels for each correlated set of bands, while in parallel a conventional CNN learns spatial–spectral characteristics. The models are combined at the end. Li et al.[181] incorporated vicinal pixel information by combining the center pixel and vicinal pixels, and using a voting strategy to classify the pixels.

Multitemporal analysis. Multitemporal analysis is a subset of RS analysis that has its own challenges. Data revisit rates are often long, and ground-truth data are even more expensive as multiple imagery sets have to be analyzed, and images must be coregistered for most applications.

Jianya et al.[12] review multitemporal analysis, and observe that it is hard, the changes are often nonlinear, and changes occur on different timescales (seasons, weeks, years, etc.). The process from ground objects to images is not reversible, and image change to earth change is a very difficult task. Hybrid method involving classification, object analysis, physical modeling, and time-series analysis can all potentially benefit from DL approaches. Arel et al.[120] ask if DL frameworks can understand trends over short, medium, and long times? This is an open question for RNNs.

Change detection is an important subset of multitemporal analysis. Hussain et al.[11] state that change detection can benefit from texture analysis, accurate classifications, and from the ability to detect anomalies. DL has huge potential to address these issues, but it is recognized that DL algorithms are not common in image processing software in this field, and large training sets and large training times may also be required. In cases of nonnormal distributions, ANNs have shown superior results to other statistical methods. They also recognize that DL-based change detection can go beyond traditional pixel-based change detection methods. Tewkesbury et al.[395] observe that change detection can occur at the pixel, kernel (group of pixels), image-object, multitemporal image-object (created by segmenting over time series), vector-polygon, and hybrid. While the pixel level is suitable for many applications, hybrid approaches can yield better results in many cases. Approaches to change detection can use DL to (1) coregister images and (2) detect changes at hierarchical (e.g., more than just pixel levels).

Some selected specific applications that can benefit from DL analysis. This section discusses some selected applications where applying DL techniques can benefit the results. This is by no means an exhaustive list, and many other areas can potentially benefit from DL approaches. In oil spill detection, Brekke and Solberg[396] point out that training data are scarce. Oil spills are very rare, which usually means oil spill detection approaches are anomaly detectors. Physical proximity, slick shape, and texture play important roles. SAR imagery is very useful, but there are look-alike phenomena that cause false positives. Algal information fusion from optical sensors and probability models can aid detection. Current algorithms are not reliable, and DL has great promise in this area.

In the area of pedestrian detection, Dollar et al.[8] discuss that many images with pedestrians have only a small number of pixels. Robust detectors must handle occlusions. Motion features can achieve very high performance, but few have used them. Context (ground plane) approaches are needed, especially at lower resolutions. More data sets are needed, especially with occlusions. Again, DL can provide significant results in this area.

For urban structure analysis, Mayer[397] reports that scale-space analysis is required due to different scales of urban structures. Local contexts can be used in the analysis. Analyzing parts (dormers, windows, etc.) can improve results. Sensor fusion can aid results. Object variability is not treated sufficiently (e.g., highly nonplanar roofs). The DL system's ability to learn hierarchical components and learn parts makes is a good candidate for improving results in this area.

In pixel unmixing, Shi and Wang[20] and Somers et al.[398] review papers both point out that whether an unmixing system uses a spectral library or extracts endmembers spectra from the imagery, the accuracy highly depends on the selection of appropriate endmembers. Adding information from a spatial neighborhood can enhance the unmixing results. DL methods such as CNNs or other tailored systems can potentially inherently combine spectral and spatial

information. DL systems using denoising, iterative unmixing, feature selection, spectral weighting, and spectral transformations can benefit unmixing.

Finally in the area of road extraction, Wang et al.[354] point out that roads can have large variability, are often curves, and can change size. In bad weather, roads can be very hard to identify. Object shadows, occlusions, etc. can cause the road segmentation to miss sections. Multiple models and multiple features can improve results. The natural ability of DL to learn complicated hierarchical features from data makes them a good candidate for this application area also.

4.6 *Transfer Learning*

4.6.1 *Open question #6: how can DL in RS successfully use transfer learning?*

In general, we note that transfer learning is also an open question in DL in general, not just in DL related to RS. Section 4.9 discusses transfer learning in the broader contact of the entire field of DL, which this section discusses transfer learning in an RS context.

According to Tuia et al.[399] and Pan and Yang,[400] transfer learning seeks to learn from one area to another in one of the four ways: instance-transfer, feature-representation transfer, parameter transfer, and relational-knowledge transfer. Typically in RS, when changing sensors or changing to a different part of a large image or other imagery collected at different times, the transfer fails. RS systems need to be robust, but many do not necessarily require near-perfect knowledge. Transfer among HSI images where the number and types of endmembers are different has very few studies. Ghazi et al.[255] suggest that two options for transfer learning are to (1) use pretrained network and learn new features in the imagery to be analyzed or (2) fine-tune the weights of the pretrained network using the imagery to be analyzed. The choice depends on the size and similarity of the training and testing data sets. There are many open questions about transfer learning in HSI RS:

- How does HSI transfer work when the number and type of endmembers are different?
- How can DL systems transfer low- to midlevel features from other domains into RS?
- How can DL transfer learning be made robust to imagery collected at different times and under different atmospheric conditions?

Although in general these open questions remain, we do note that the following papers have successfully used transfer learning in RS applications: Yang et al.[199] trained on other RS imagery and transferred low- to midlevel features to other imagery. Othman et al.[235] used transfer learning by training on the ILSVRC-12 challenge data set, which has 1.2 million 224×224 RGB images belonging to 1000 classes. The trained system was applied to the UC Merced Land Use[401] and Banja-Luka[402] data sets. Iftene et al.[173] applied a pretrained CaffeNet and GoogLeNet models on the ImageNet data set, and then applying the results to the VHR imagery denoted the WHU-RS data set.[403,404] Xie et al.[310] trained a CNN on nighttime imagery and used it in a poverty mapping. Ghazi et al.[255] and Lee et al.[405] used a pretrained networks AlexNet, GoogLeNet, and VGGNet on the LifeCLEF 2015 plant task data set[406] and MalayaKew data set[407] for plant identification. Alexandre[240] used four independent CNNs, one for each channel of RGBD, instead of using a single CNN receiving the four input channels. The four independent CNNs are then trained in a sequence using the weights of a trained CNN as starting point to train the other CNNs that will process the remaining channels. Ding et al.[408] used transfer learning for automatic target recognition from midwave infrared (MWIR) to longwave IR (LWIR). Li et al.[141] used transfer learning by using pixel-pairs based on reference data with labeled sampled using Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) hyperspectral data.

4.7 *Improved Theoretical Understanding of DL Systems*

4.7.1 *DL open question #7: what new developments will allow researchers to better understand DL systems theoretically?*

The CV and NN image processing communities understand BP and SGD, but until recently, researchers struggled to train deep networks. One issue has been identified as vanishing or

exploding gradients.[61,409] Using normalized initialization and normalization layers can help alleviate this problem. Using special architectures, such as deep residual learning[30] or highway networks[76] feed data into the deeper layers, thus allowing very deep networks to be trained. FCNs[337] have achieved success in pixel-based semantic segmentation tasks and are another alternative to going deep. Sokolic et al.[410] determined that the spectral norm of the NN's Jacobian matrix in the neighborhood of the training samples must be bounded in order for the network to generalize well. All of these methods deal with a central problem in training very deep NNs. The gradients must not vanish, explode, or become too uncorrelated, or else learning is severely hindered.

The DL field needs practical (and theoretical) methods to go deep, and ways to train efficiently with good generalization capabilities. Many DL RS systems will probably require new components, and these networks with the new components need to be analyzed to see if the methods above (or new methods not yet invented) will enable efficient and robust network training. Egmont-Petersen et al.[348] point out that DL training is sensitive to the initial training samples, and it is a well-known problem in SGD and BP of potentially reaching a local minimum solution but not being at the global minimum. In the past, seminal papers, such as Hinton et al.'s,[57] that allow efficient training of the network have allowed researchers to break past a previously difficult barrier.

4.8 *High Barriers to Entry*

4.8.1 *DL open question #8: how to best handle high entry barriers to DL?*

Most DL papers assume that the reader is familiar with DL concepts, e.g., BP, SGD, etc. This is in reality a steep learning curve that takes a long time to master. Good tutorials and online training can aid students and practitioners who are willing to learn. Implementing BP or SGD on a large DL system is a difficult task, and simple errors can be hard to determine. Furthermore, BP can fail in large networks, so alternate architectures, such as highway nets, may be required.

Many DL systems have a large number of parameters to learn and often require large amounts of training data. Computers with GPUs and GPU-capable DL programs can greatly benefit by offloading computations onto the GPUs. However, multi-GPU systems are expensive, and students often use laptops that cannot be equipped with a GPU. Some DL systems run under Microsoft Windows, while others run under variants of Linux (e.g., Ubuntu or Red Hat). Furthermore, DL systems are programmed in a variety of languages, including MATLAB®, C, C++, Lua, and Python. Thus, practitioners and researchers have a potentially steep learning curve to create custom DL solutions.

Finally, a large variety of data types in RS, including RGB imagery, RGBD imagery, MSI, HSI, SAR, LiDAR, stereo imagery, tweets, and GPS data, all of which may require different architectures of DL systems. Often, many of the tasks in the RS community require components that are not part of a standard DL library tool. A good understanding of DL systems and programming is required to integrate these components into off-the-shelf DL systems.

One of the authors provides a set of MatConvNet DL example codes and codes for fusion on his website. Interested readers are referred to http://derektanderson.com/FuzzyLibrary/.

4.9 *Training*

4.9.1 *Open question #9: how to train and optimize the DL system?*

Training a DL system can be difficult. Large systems can have millions of parameters. There are many methods that DL researchers use to effectively train systems. These methods are discussed as follows.

Data imputation. Data imputation,[57] also called data augmentation, is important in RS, since there are often a small number of training samples. In imagery, image patched can be extracted and stretched with affine transformations, rotated, and made lighter or darker (scaling). Also, patched can be zeroed (removed) from training data to help the DL be more robust to

occlusions. Data can also be augmented by simulations. Another method that can be useful in some circumstances is domain transfer, discussed below (transfer learning). Another common approach for RGB imagery is to train with rotated version of the image and by varying the image brightness or intensity.

Pretraining. Erhan et al.[411] performed a detailed study trying to answer the questions: "How does unsupervised pretraining work?" and "Why does unsupervised pretraining help DL?" Their empirical analysis shows that unsupervised pretraining guides the learning toward attraction basins of minima that support better generalization and pretraining also acts as a regularizer. Furthermore, early training examples have a large impact on the overall DL performance. Of course, these are experimental results, and results on other data sets or using other DL methods can yield different results. Many DL systems use pretraining followed by fine-tuning.

Transfer learning. Transfer learning is also discussed in Sec. 4.6. Transfer learning attempts to transfer learned features (which can also be thought of as DL layer activations or outputs) from one image to another, from one part of an image to another part, or from one sensor to another. This is a particularly difficult issue in RS, due to variations in atmosphere, lighting conditions, etc. Pan and Yang[400] point out that typically in RS applications, when changing sensors or changing to a different part of a large image or other imagery collected at different times, the transfer fails. RS systems need to be robust, but they do not necessarily require near-perfect knowledge. Also, transfer among images where the number and types of endmembers are different has very few studies. Zhang et al.[3] also cite transfer learning as an open issue in DL in general, and not just in RS.

Regularization. Regularization is defined by Goodfellow et al.[23] as "any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error." There are many forms of regularizer—parameter size penalty terms (such as the l_2 or l_1 norm, and other regularizers that enforce sparse solutions; diagonal loading of a matrix so the matrix inverse (which is required for some algorithms) is better conditioned; dropout and early stopping (both are described below); adding noise to weights or inputs; semisupervised learning, which usually means that some function that has a very similar representation to examples from the same class is learned by the NN; bagging (combining multiple models); and adversarial training, where a weighted sum of the sample and an adversarial sample is used to boost performance. The interested reader is referred to chapter 7 of Ref. 23 for further information. An interesting DL example in RS is, Mei et al.,[190] who used a PReLU,[412] which can help improve model fitting without adding computational cost and with little overfitting risk.

Early stopping. Early stopping is a method where the training validation error is monitored and previous coefficient values are recorded. Once the training level reaches stopping criteria, then the coefficients are used. Early stopping helps to mitigate overtraining. It also acts as a regularizer, constraining the parameter space to be close to the initial configuration.[411]

Dropout. Dropout usually uses some number of randomly selected links (or a probability that a link will be dropped).[413] As the network is trained, these links are zeroed, basically stopping data from flowing from the shallower to deeper layers in the DL system. Dropout basically allows a bagging-like effect, but instead of the individual networks being independent, they share values, but the mixing occurs at the dropout layer, and the individual subnetworks share parameters.[23] Dropout may not be needed or advantageous if batch normalization is used.

Batch normalization. Batch normalization was developed by Ioffe and Szegedy.[414] Batch normalization breaks the data into small batches and then normalizes the data to be zero mean and unity variance. Batch normalization can also be added internally as layers in the network. Batch normalization reduces the so-called internal covariate shift problem for each training mini-batch. Applying batch normalization had the following benefits: (1) allowed a higher learning rate, (2) the DL network was not as sensitive to initialization, (3) dropout was not required to mitigate overfitting, (4) the l_2 weight regularization could be reduced, increasing accuracy. Adding batch normalization increases the two extra parameters per activation.

Optimization. Optimization of DL networks is a major area of study in DL. It is nontrivial to train a DL network, much less squeeze out high performance on both the training and testing data sets. SGD is a training method that uses small batches of training data to generate an estimate of the gradients. Le et al.[415] argues the SGD is not inherently parallel and often requires training many models and choosing the one that performs best on the validation set. They also show that no one method works best in all cases. They found that optimization performance varies per problem. A nice review paper for many gradient descent algorithms is provided by Ruder.[416] According to Ruder, complications for gradient descent algorithms include:

- how to choose a proper learning rate,
- how to properly adjust learning-rate schedules for optimal performance,
- how to adjust learning rates independently for each parameter, and
- how to avoid getting trapped in local minima and saddle points when one dimension slopes up and one down (the gradients can get very small and training halts).

Various gradient descent methods, such as AdaGrad,[53] which adapts the learning rate to the parameters, AdaDelta,[417] which uses a fixed size window of past data, and Adam,[55] which also has both mean and variance terms for the gradient descent, can be used for training. Another recent approach seeks to optimize the learning rate from the data is described in Schaul et al.[418] Finally, Sokolic et al.[410] concluded experimentally that for a DNN to generalize well, the spectral norm of the NN's Jacobian matrix in the neighborhood of the training samples must be bounded. They furthermore show that the generalization error can be bounded independent of the DL network's depth or width, provided that the Jacobian spectral norm is bounded. They also analyze residual networks, weight normalized networks, CNN's with batch normalization and Jacobian regularization, and residual networks with Jacobian regularization. The interested reader is referred to chapter 8 of Ref. 23 for further information.

Data propagation Both highway networks[73] and residual networks[30] are methods that take data from one layer and incorporate it, either directly (highway networks) or as a difference (residual networks) into deeper layers. These methods both allow very deep networks to be trained, at the expense of some additional components. Balduzzi et al.[419] examined networks and determined that there is a so-called "shattered gradient" problem in DNN, which is manifested by the gradient correlation decaying exponentially with depth and thus gradients resemble white noise. A "looks linear" initialization is developed that prevents the gradient shattering. This method appears not to require skip connections (highway networks, residual networks).

5 Conclusions

We performed a thorough review and analyzed 205 RS papers that use FL and DL, as well as 57 survey papers in DL and RS. We provided researchers with a clustered set of 12 areas where DL has been applied to RS. We quickly summarize these findings. First, the bulk of attention has

been placed on tasks such as classification, segmentation, object detection, and tracking. This is as expected, since it is the closest to what the CV community has studied and it is potentially the lowest hanging fruit for DL in RS. We also highlighted interesting preliminary work in DL-based transfer learning, primarily as it relates to a single sensing modality (e.g., RGB imagery). However, it is not yet clear how to carry out transfer learning for non-RGB RS specific sensors and/or multisensor platforms, i.e., visible, near-IR, hyperspectral, SAR, etc. RS researchers have also just started to investigate DL for 3-D processing. However, non-RS research in DL-based stereo vision for CV is arguably more mature and a possible area of interest and future integration. Similar to transfer learning, DL-based super-resolution was also highlighted and we discussed that it needs new theory and applications with respect to multimodal (sensor) systems. To our surprise, weather forecasting had a lot more DL work than what we expected. However, whereas many exciting works have started to emerge, we observed that new abstract theory, versus a case-by-case solution, is needed as it relates to the integration (and fusion) of domain models and/or knowledge in RS DL. Next, we discussed image enhancement and change detection, which are massive areas in RS and to the best of our knowledge, limited DL work has been performed on them to date. As stated in the case of 3-D processing in RS, semantic labeling, either as symbols or natural language statements, is another topic in RS that is in an infancy stage and could likely benefit from integration of work in CV surrounding signal-to-text. Last, dimensionality reduction is at the heart of tasks such as multisensor ("multispectral") and HSI signal processing. However, to date most methods have focused on using "human created" transforms (e.g., principle components analysis) to reduce the dimensionality pre-DL versus addressing how DL can accomplish this feat (possibly in a more effective and/or more efficient way).

Herein, we also examined why DL is popular and we looked at some factors that enable DL, from theory to applications and systems/hardware. We showed what data sets are being used in different DL RS research—sadly not many and not a lot of agreement at that—and we highlighted the fact that new high-volume and high variety benchmark community data sets are direly needed for purposes such as reproducible research and comparability across DL approaches, and many commonly used RS data sets are saturated. Furthermore, we emphasized that fundamental DL questions plague the field at large (and therefore RS); e.g., how many neurons, layers, what "types" of layers to use, what learning algorithm, penalty terms (or methods such as batch normalization), drop out, etc. are needed. In addition, we summarized existing DL tools (libraries) that RS researchers should be aware of. It is hard to pick a "winner." These packages vary in what they support, e.g., types of data, programming language, other dependencies (e.g., MATLAB®). Whereas there is a variety in languages and etc., TensorFlow has recently become a dominant figure in terms of factors, such as performance, programming language, community support, and documentation. We also critically looked at the DL RS field and identified nine general areas with unsolved challenges and opportunities. Specifically, we enumerated 11 difficult thought-provoking open questions. Whereas we summarized the 12 clusters of existing work above, we cannot succinctly and sufficiently summarize the nine unsolved challenges. The reader should refer to Sec. 4 for full detail, i.e., contributing factors, recommendations, related work, etc. Last, we provided multiple tables, e.g., a table of survey papers that address, in one form or another, DL and/or FL in RS.

In conclusion, DL and FL are hot emerging topics, with no drop-off in sight, that have much promise and relevance to RS. However, whereas there has been a good deal of initial RS work to date, much more work is needed, in both theory and in application. Although many tasks in RS can be thought about as signal/image processing challenges, RS offers a great deal of complexity (and therefore should be viewed as an opportunity). To make advancements in DL RS, we need more individuals trained in a variety of topics from electromagnetics to multisensor systems, signal processing, machine learning (including DL), and data fusion, to name a few. The point is, the cost of entry to DL RS is high and the field is multidisciplinary.

Disclosures

The authors declare no conflict of interest.

Acknowledgments

The authors wish to thank graduate students Vivi Wei, Julie White, and Charlie Veal for their valuable inputs related to DL tools.

References

1. G. Cheng, J. Han, and X. Lu, "Remote sensing image scene classification: benchmark and state of the art," arXiv:1703.00121 [cs.CV] (2017).
2. L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning," *APSIPA Trans. Signal Inf. Process.* **3**(e2), 1–29 (2014).
3. L. Zhang, L. Zhang, and V. Kumar, "Deep learning for remote sensing data," *IEEE Geosci. Remote Sens. Mag.* **4**, 22–40 (2016).
4. J. M. Bioucas-dias et al., "Hyperspectral remote sensing data analysis and future challenges," *IEEE Geosci. Remote Sens. Mag.* **1**, 6–36 (2013).
5. G. Camps-Valls and L. Bruzzone, "Kernel-based methods for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.* **43**(6), 1351–1362 (2005).
6. G. Camps-Valls et al., "Advances in hyperspectral image classification: Earth monitoring with statistical learning methods," *IEEE Signal Process. Mag.* **31**(1), 45–54 (2013).
7. H. Deborah, N. Richard, and J. Y. Hardeberg, "A comprehensive evaluation of spectral distance functions and metrics for hyperspectral image processing," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **8**(6), 3224–3234 (2015).
8. P. Dollar et al., "Pedestrian detection: an evaluation of the state of the art," *IEEE Trans. Pattern Anal. Mach. Intell.* **34**(4), 743–761 (2012).
9. P. Du et al., "Multiple classifier system for remote sensing image classification: a review," *Sensors*, **12**(4), 4764–4792 (2012).
10. M. Fauvel et al., "Advances in spectral–spatial classification of hyperspectral images," *Proc. IEEE* **101**(3), 652–675 (2013).
11. M. Hussain et al., "Change detection from remotely sensed images: from pixel-based to object-based approaches," *ISPRS J. Photogramm. Remote Sens.* **80**, 91–106 (2013).
12. G. Jianya et al., "A review of multi-temporal remote sensing data change detection algorithms," *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.* **37**(B7), 757–762 (2008).
13. D. J. Lary et al., "Machine learning in geosciences and remote sensing," *Geosci. Front.* **7**(1), 3–10 (2016).
14. D. Lunga et al., "Manifold-learning-based feature extraction for classification of hyperspectral data: a review of advances in manifold learning," *IEEE Signal Process. Mag.* **31**(1), 55–66 (2014).
15. A. Plaza et al., "A quantitative and comparative analysis of endmember extraction algorithms from hyperspectral data," *IEEE Trans. Geosci. Remote Sens.* **42**(3), 650–663 (2004).
16. N. Keshava and J. F. Mustard, "Spectral unmixing," *IEEE Signal Process. Mag.* **19**(1), 44–57 (2002).
17. N. Keshava, "A survey of spectral unmixing algorithms," *Lincoln Lab. J.* **14**(1), 55–78 (2003).
18. M. Parente and A. Plaza, "Survey of geometric and statistical unmixing algorithms for hyperspectral images," in *2nd Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS 2010)*, pp. 1–4, IEEE (2010).
19. A. Plaza et al., "Recent developments in endmember extraction and spectral unmixing," in *Optical Remote Sensing*, S. Prasad, L. M. Bruce, and J. Chanussot, Eds., pp. 235–267, Springer, Berlin, Germany (2011).
20. C. Shi and L. Wang, "Incorporating spatial information in spectral unmixing: a review," *Remote Sens. Environ.* **149**, 70–87 (2014).
21. M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *2011 Int. Conf. on Computer Vision*, pp. 2018–2025 (2011).
22. M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European Conf. on Computer Vision*, pp. 818–833, Springer (2014).

23. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, Cambridge, Massachusetts (2016).
24. D. Arpit et al., "Why regularized auto-encoders learn sparse representation?," in *Int. Conf. on Machine Learning*, pp. 136–144 (2016).
25. M. Chen et al., "Marginalized denoising autoencoders for domain adaptation," arXiv:1206.4683 (2012).
26. Ujjwalkarn, "An intuitive explanation of convolutional neural networks," https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/ (2016).
27. Y. LeCun et al., "Gradient-based learning applied to document recognition," *Proc. IEEE* **86**(11), 2278–2324 (1998).
28. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, G. Tesauro, D. Touretzky, and T. Leed, Eds., pp. 1097–1105 (2012).
29. C. Szegedy et al., "Going deeper with convolutions," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 1–9 (2015).
30. K. He et al., "Deep residual learning for image recognition," CoRR abs/1512.03385 (2015).
31. G. Huang et al., "Densely connected convolutional networks," arXiv:1608.06993 (2016).
32. A. Deshpande, "A beginner's guide to understanding convolutional neural networks, part 2," http://www.kdnuggets.com/2016/09/beginners-guide-understanding-convolutional-neural-networks-part-2.html (29 July 2017).
33. A. Deshpande, "A beginner's guide to understanding convolutional neural networks, part 1," https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/ (29 July 2017).
34. V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," arXiv:1603.07285 (2016).
35. G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science* **313**(5786), 504–507 (2006).
36. R. Salakhutdinov and G. Hinton, "Deep Boltzmann machines," in *Proc. of the Twelfth Int. Conf. on Artificial Intelligence and Statistics (AISTATS'09)*, Vol. **5**, pp. 448–455 (2009).
37. S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.* **9**(8), 1735–1780 (1997).
38. D. Britz, "Recurrent neural networks tutorial, part 1 introduction to RNNs," http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/ (29 July 2017).
39. "Recurrent neural networks," https://www.tensorflow.org/tutorials/recurrent (03 August 2017).
40. Q. V. Le, "A tutorial on deep learning part 2: autoencoders, convolutional neural networks and recurrent neural networks," *Google Brain*, pp. 1–20 (2015).
41. "A beginner's guide to recurrent networks and LSTMs," https://deeplearning4j.org/lstm.html (03 August 2017).
42. M. D. Zeiler et al., "Deconvolutional networks," in *Computer Vision and Pattern Recognition (CVPR)* (2010).
43. H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," CoRR abs/1505.04366 (2015).
44. O. Russakovsky et al., "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vision* **115**(3), 211–252 (2015).
45. L. Brown, "Deep learning with GPUs," www.nvidia.com/content/events/geoInt2015/LBrown_DL.pdf (2015).
46. G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control Signals Syst.* **2**(4), 303–314 (1989).
47. M. Telgarsky, "Benefits of depth in neural networks," arXiv:1602.04485 (2016).
48. O. Sharir and A. Shashua, "On the expressive power of overlapping operations of deep networks," arXiv:1703.02065 (2017).
49. S. Liang and R. Srikant, "Why deep neural networks for function approximation?," arXiv:1610.04161 (2016).

50. L. Zhang and P. N. Suganthan, "A survey of randomized algorithms for training neural networks," *Inf. Sci.* **364**, 146–155 (2016).
51. T. Schaul, I. Antonoglou, and D. Silver, "Unit tests for stochastic optimization," arXiv:1312.6055 (2013).
52. B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Comput. Math. Math. Phys.* **4**(5), 1–17 (1964).
53. J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.* **12**, 2121–2159 (2011).
54. T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Networks Mach. Learn.* **4**(2), 26–31 (2012).
55. D. Kingma and J. Ba, "Adam: a method for stochastic optimization," arXiv:1412.6980 (2014).
56. J. Schmidhuber, "Deep learning in neural networks: an overview," *Neural Networks* **61**, 85–117 (2015).
57. G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.* **18**(7), 1527–1554 (2006).
58. Y. Bengio et al., "Greedy layer-wise training of deep networks," *Adv. Neural Inf. Process. Syst.* **19**, 153 (2007).
59. R. R. Salakhutdinov and G. E. Hinton, "A better way to pretrain deep Boltzmann machines," in *Advances in Neural Information Processing Systems*, F. Pereira et al., Eds., pp. 2447–2455, Curran Associates, Inc. (2012).
60. S. Shi et al., "Benchmarking state-of-the-art deep learning software tools," arXiv:1608.07249 (2016).
61. Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Networks* **5**(2), 157–166 (1994).
62. R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Int. Conf. on Machine Learning*, pp. 1310–1318 (2013).
63. J. Martens and I. Sutskever, "Learning recurrent neural networks with Hessian-free optimization," in *Proc. of the 28th Int. Conf. on Machine Learning (ICML 2011)*, pp. 1033–1040 (2011).
64. Y. Bengio, A. C. Courville, and P. Vincent, "Unsupervised feature learning and deep learning: a review and new perspectives," CoRR, abs/1206.55381 (2012).
65. Y. Ma et al., "Remote sensing big data computing: challenges and opportunities," *Future Gener. Comput. Syst.* **51**, 47–60 (2015)Special section: a note on new trends in data-aware scheduling and resource provisioning in modern HPC systems.
66. M. Chi et al., "Big data for remote sensing: challenges and opportunities," *Proc. IEEE* **104**, 2207–2219 (2016).
67. R. Raina, A. Madhavan, and A. Y. Ng, "Large-scale deep unsupervised learning using graphics processors," in *Proc. of the 26th Annual Int. Conf. on Machine Learning (ICML 2009)*, pp. 873–880, ACM, New York (2009).
68. D. C. Cireşan et al., "Flexible, high performance convolutional neural networks for image classification," in *Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI 2011)*, Vol. 2, pp. 1237–1242, AAAI Press (2011).
69. D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *Proc. of the 20th Int. Conf. on Artificial Neural Networks: Part III (ICANN 2010)*, pp. 92–101, Springer-Verlag, Berlin, Heidelberg (2010).
70. L. Deng, D. Yu, and J. Platt, "Scalable stacking and learning for building deep architectures," in *2012 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2133–2136 (2012).
71. B. Hutchinson, L. Deng, and D. Yu, "Tensor deep stacking networks," *IEEE Trans. Pattern Anal. Mach. Intell.* **35**, 1944–1957 (2013).
72. J. Dean et al., "Large scale distributed deep networks," in *Proc. of the 25th Int. Conf. on Neural Information Processing Systems (NIPS 2012)*, pp. 1223–1231, Curran Associates Inc., USA (2012).
73. R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," CoRR abs/1505.00387 (2015).

74. K. Greff, R. K. Srivastava, and J. Schmidhuber, "Highway and residual networks learn unrolled iterative estimation," CoRR abs/1612.07771 (2016).
75. J. G. Zilly et al., "Recurrent highway networks," CoRR abs/1607.03474 (2016).
76. R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," CoRR abs/1507.06228 (2015).
77. M. M. Najafabadi et al., "Deep learning applications and challenges in big data analytics," *J. Big Data* **2**(1), 1–21 (2015).
78. X.-W. Chen and X. Len, "Big data deep learning: challenges and perspectives," *IEEE Access* **2**, 514–525 (2014).
79. S. Landset et al., "A survey of open source tools for machine learning with big data in the Hadoop ecosystem," *J. Big Data* **2**(1), 24 (2015).
80. J. Wan et al., "Deep learning for content-based image retrieval: a comprehensive study," in *Proc. of the 22nd ACM Int. Conf. on Multimedia*, pp. 157–166 (2014).
81. Y. Jia et al., "Caffe: convolutional architecture for fast feature embedding," in *ACM Int. Conf. on Multimedia*, pp. 675–678 (2014).
82. M. Abadi et al., "TensorFlow: a system for large-scale machine learning," in *Proc. of the 12th USENIX Symp. on Operating Systems Design and Implementation (OSDI)*, Savannah, Georgia (2016).
83. A. Vedaldi and K. Lenc, "MatConvNet: convolutional neural networks for MATLAB," in *Proc. of the 23rd ACM Int. Conf. on Multimedia*, pp. 689–692, ACM (2015).
84. A. Handa et al., "GVNN: neural network library for geometric computer vision," in *Computer Vision Workshop (ECCV 2016)*, pp. 67–82, Springer International (2016).
85. F. Chollet, "Keras," https://github.com/fchollet/keras (2015).
86. T. Chen et al., "MXNet: a flexible and efficient machine learning library for heterogeneous distributed systems," 1–6, arXiv:1512.01274 (2015).
87. R. Al-Rfou et al., "Theano: a Python framework for fast computation of mathematical expressions," arXiv:1605.02688 (2016).
88. "Deep learning with torch: the 60-minute Blitz," https://github.com/soumith/cvpr2015/blob/master/Deep Learning with Torch.ipynb (18 February 2017).
89. M. Baccouche et al., "Sequential deep learning for human action recognition," in *Int. Workshop on Human Behavior Understanding*, pp. 29–39, Springer (2011).
90. Z. Shou et al., "CDC: convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos," arXiv:1703.01515 (2017).
91. D. Tran et al., "Learning spatiotemporal features with 3D convolutional networks," in *IEEE Int. Conf. on Computer Vision (ICCV 2015)*, pp. 4489–4497 (2015).
92. L. Wang et al., "Temporal segment networks: towards good practices for deep action recognition," in *European Conf. on Computer Vision*, pp. 20–36 (2016).
93. G. E. Dahl et al., "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Trans. Audio Speech Lang. Process.* **20**(1), 30–42 (2012).
94. G. Hinton et al., "Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups," *IEEE Signal Process. Mag.* **29**(6), 82–97 (2012).
95. A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP 2013)*, pp. 6645–6649, IEEE (2013).
96. W. Luo, A. G. Schwing, and R. Urtasun, "Efficient deep learning for stereo matching," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 5695–5703 (2016).
97. S. Levine et al., "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," arXiv:1603.02199 (2016).
98. S. Venugopalan et al., "Sequence to sequence-video to text," in *Proc. of the IEEE Int. Conf. on Computer Vision*, pp. 4534–4542 (2015).
99. R. Collobert, "Deep learning for efficient discriminative parsing," in *Proc. of the Fourteenth Int. Conf. on Artificial Intelligence and Statistics*, Vol. **15**, pp. 224–232 (2011).
100. S. Venugopalan et al., "Translating videos to natural language using deep recurrent neural networks," arXiv:1412.4729 (2014).

101. Y. H. Tan and C. S. Chan, "phi-LSTM: a phrase-based hierarchical LSTM model for image captioning," in *13th Asian Conf. on Computer Vision (ACCV)*, pp. 101–117 (2016).
102. A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 3128–3137 (2015).
103. P. Baldi, P. Sadowski, and D. Whiteson, "Searching for exotic particles in high-energy physics with deep learning," *Nat. Commun.* **5**, 4308 (2014).
104. J. Wu et al., "Galileo: perceiving physical object properties by integrating a physics engine with deep learning," in *Advances in Neural Information Processing Systems*, C. Cortes et al., Eds., pp. 127–135, Curran Associates, Inc. (2015).
105. A. A. Cruz-Roa et al., "A deep learning architecture for image representation, visual interpretability and automated basal-cell carcinoma cancer detection," in *Int. Conf. on Medical Image Computing and Computer-Assisted Intervention*, pp. 403–410, Springer (2013).
106. R. Fakoor et al., "Using deep learning to enhance cancer diagnosis and classification," in *Proc. of the Int. Conf. on Machine Learning* (2013).
107. K. Sirinukunwattana et al., "Locality sensitive deep learning for detection and classification of nuclei in routine colon cancer histology images," *IEEE Trans. Med. Imaging* **35**(5), 1196–1206 (2016).
108. M. Längkvist, L. Karlsson, and A. Loutfi, "A review of unsupervised feature learning and deep learning for time-series modeling," *Pattern Recognit. Lett.* **42**, 11–24 (2014).
109. S. Sarkar et al., "Early detection of combustion instability from hi-speed flame images via deep learning and symbolic time series analysis," in *Annual Conf. of the Prognostics and Health Management* (2015).
110. T. Kuremoto et al., "Time series forecasting using a deep belief network with restricted Boltzmann machines," *Neurocomputing* **137**, 47–56 (2014).
111. A. Dosovitskiy, J. Tobias Springenberg, and T. Brox, "Learning to generate chairs with convolutional neural networks," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 1538–1546 (2015).
112. J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based generative adversarial network," arXiv:1609.03126 (2016).
113. S. E. Reed et al., "Learning what and where to draw," in *Advances in Neural Information Processing Systems*, D. D. Lee et al., Eds., pp. 217–225, Curran Associates, Inc. (2016).
114. D. Berthelot, T. Schumm, and L. Metz, "Began: boundary equilibrium generative adversarial networks," arXiv:1703.10717 (2017).
115. W. R. Tan et al., "ArtGAN: artwork synthesis with conditional categorial gans," arXiv:1702.03410 (2017).
116. K. Gregor et al., "Draw: a recurrent neural network for image generation," arXiv:1502.04623 (2015).
117. A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," arXiv:1511.06434 (2015).
118. X. Ding et al., "Deep learning for event-driven stock prediction," in *Proc. of the Twenty-Fourth Int. Joint Conf. on Artificial Intelligence (IJCAI '15)*, pp. 2327–2333 (2015).
119. Z. Yuan et al., "Droid-sec: deep learning in android malware detection," *ACM SIGCOMM Comput. Commun. Rev.* **44**(4), 371–372 (2014).
120. I. Arel, D. C. Rose, and T. P. Karnowski, "Deep machine learning—a new frontier in artificial intelligence research," *IEEE Comput. Intell. Mag.* **5**(4), 13–18 (2010).
121. W. Liu et al., "A survey of deep neural network architectures and their applications," *Neurocomputing* **234**, 11–26 (2017).
122. H. Wang and B. Raj, "A survey: time travel in deep learning space: an introduction to deep learning models and how deep learning models evolved from the initial ideas," arXiv:1510.04781 abs/1510.04781 (2015).
123. D. Yu and L. Deng, "Deep learning and its applications to signal and information processing [exploratory DSP]," *IEEE Signal Process. Mag.* **28**(1), 145–154 (2011).
124. L. Arnold et al., "An introduction to deep learning," in *11th Int. Conf. on Information Science, Signal Processing and their Applications (ISSPA '12)*, pp. 1438–1439 (2012).

125. L. Deng and D. Yu, "Deep learning: methods and applications," *Found. Trends Signal Process.* **7**(3–4), 197–387 (2013).
126. C. Cadena, A. Dick, and I. D. Reid, "Multi-modal auto-encoders as joint estimators for robotics scene understanding," in *Proc. of Robotics: Science and Systems Conf. (RSS)* (2016).
127. J. Feng, Y. Wang, and S.-F. Chang, "3D shape retrieval using a single depth image from low-cost sensors," in *IEEE Winter Conf. on Applications of Computer Vision (WACV '16)*, pp. 1–9 (2016).
128. A. Haque, A. Alahi, and L. Fei-fei, "Recurrent attention models for depth-based person identification," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR '16)*, pp. 1229–1238 (2016).
129. V. Hegde and R. Zadeh, "FusionNet: 3D object classification using multiple data representations," arXiv:1607.05695 (2016).
130. J. Huang and S. You, "Point cloud labeling using 3D convolutional neural network," in *Int. Conf. on Pattern Recognition* (2016).
131. W. Kehl et al., "Deep learning of local RGB-D patches for 3D object detection and 6D pose estimation," in *European Conf. on Computer Vision*, pp. 205–220 (2016).
132. C. Li, A. Reiter, and G. D. Hager, "Beyond spatial pooling: fine-grained representation learning in multiple domains," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Vol. 2, pp. 4913–4922 (2015).
133. N. Sedaghat, M. Zolfaghari, and T. Brox, "Orientation-boosted voxel nets for 3D object recognition," arXiv:1604.03351 (2016).
134. Z. Xie et al., "Projective feature learning for 3D shapes with multi-view depth images," in *Computer Graphics Forum*, Vol. **34**(7), pp. 1–11, Wiley Online Library (2015).
135. C. Chen, "DeepDriving: learning affordance for direct perception in autonomous driving," in *Proc. of the IEEE Int. Conf. on Computer Vision*, pp. 2722–2730 (2015).
136. X. Chen et al., "Multi-view 3D object detection network for autonomous driving," arXiv:1611.07759 (2016).
137. A. Chigorin and A. Konushin, "A system for large-scale automatic traffic sign recognition and mapping," in *CMRT13–City Models, Roads and Traffic 2013*, pp. 13–17 (2013).
138. D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR '12)*, pp. 3642–3649 (2012).
139. Y. Zeng et al., "Traffic sign recognition using extreme learning classifier with deep convolutional features," in *Int. Conf. on Intelligence Science and Big Data Engineering (IScIDE '15)*, Suzhou, China (2015).
140. A.-B. Salberg, "Detection of seals in remote sensing images using features extracted from deep convolutional neural networks," in *IEEE Int. Geoscience and Remote Sensing Symp. (IGARSS'15)*, pp. 1893–1896 (2015).
141. W. Li, G. Wu, and Q. Du, "Transferred deep learning for anomaly detection in hyperspectral imagery," *IEEE Geosci. Remote Sens. Lett.* **14**(5), 597–601 (2017).
142. J. Becker et al., "Deep belief networks for false alarm rejection in forward-looking ground-penetrating radar," *Proc. SPIE* **9454**, 94540W (2015).
143. C. Bentes, D. Velotto, and S. Lehner, "Target classification in oceanographic SAR images with deep neural networks: architecture and initial results," in *IEEE Int. Geoscience and Remote Sensing Symposium (IGARSS '15)*, pp. 3703–3706, IEEE (2015).
144. L. E. Besaw, "Detecting buried explosive hazards with handheld GPR and deep learning," *Proc. SPIE* **9823**, 98230N (2016).
145. L. E. Besaw and P. J. Stimac, "Deep learning algorithms for detecting explosive hazards in ground penetrating radar data," *Proc. SPIE* **9072**, 90720Y (2014).
146. K. Du et al., "SAR ATR based on displacement-and rotation-insensitive CNN," *Remote Sens. Lett.* **7**(9), 895–904 (2016).
147. S. Chen and H. Wang, "SAR target recognition based on deep learning," in *Int. Conf. on Data Science and Advanced Analytics (DSAA '14)*, pp. 541–547, IEEE (2014).
148. D. A. E. Morgan, "Deep convolutional neural networks for ATR from SAR imagery," *Proc. SPIE* **9475**, 94750F (2015).

149. J. C. Ni and Y. L. Xu, "SAR automatic target recognition based on a visual cortical system," in *6th Int. Congress on Image and Signal Processing (CISP '13)*, Vol. 2, pp. 778–782, IEEE (2013).
150. Z. Sun, L. Xue, and Y. Xu, "Recognition of SAR target based on multilayer auto-encoder and SNN," *Int. J. Innovative Comput. Inf. Control* **9**(11), 4331–4341 (2013).
151. H. Wang et al., "Application of deep-learning algorithms to MSTAR data," in *IEEE Int. Geoscience and Remote Sensing Symp. (IGARSS '15)*, pp. 3743–3745, IEEE (2015).
152. L. Zhang et al., "A multifeature tensor for remote-sensing target recognition," *IEEE Geosci. Remote Sens. Lett.* **8**(2), 374–378 (2011).
153. L. Zhang, Z. Shi, and J. Wu, "A hierarchical oil tank detector with deep surrounding features for high-resolution optical satellite imagery," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **8**(10), 4895–4909 (2015).
154. P. F. Alcantarilla et al., "Streetview change detection with deconvolutional networks," in *Robotics: Science and Systems Conf. (RSS)* (2016).
155. M. Gong, Z. Zhou, and J. Ma, "Change detection in synthetic aperture radar images based on deep neural networks," *IEEE Trans. Neural Networks Learn. Syst.* **27**(1), 125–138 (2016).
156. F. Pacifici et al., "An innovative neural-net method to detect temporal changes in high-resolution optical satellite imagery," *IEEE Trans. Geosci. Remote Sens.* **45**(9), 2940–2952 (2007).
157. S. Stent, "Detecting change for multi-view, long-term surface inspection," in *Proc. of the 2015 British Machine Vision Conf. (BCMV '15)*, pp. 1–12 (2015).
158. J. Zhao et al., "Deep learning to classify difference image for image change detection," in *Int. Joint Conf. on Neural Networks (IJCNN '14)*, pp. 411–417, IEEE (2014).
159. S. Basu et al., "DeepSat: a learning framework for satellite imagery," in *Proc. of the 23rd SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems*, pp. 1–22 (2015).
160. Y. Bazi et al., "Differential evolution extreme learning machine for the classification of hyperspectral images," *IEEE Trans. Geosci. Remote Sens.* **11**(6), 1066–1070 (2014).
161. J. Cao, Z. Chen, and B. Wang, "Deep convolutional networks with superpixel segmentation for hyperspectral image classification," in *IEEE Geoscience and Remote Sensing Symp. (IGARSS '16)*, pp. 3310–3313 (2016).
162. J. Cao, Z. Chen, and B. Wang, "Graph-based deep convolutional networks with superpixel segmentation for hyperspectral image classification," in *2016 IEEE Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 3310–3313 (2016).
163. C. Chen, W. Li, and H. Su et al., "Spectral–spatial classification of hyperspectral image based on kernel extreme learning machine," *Remote Sens.* **6**(6), 5795–5814 (2014).
164. G. Cheng et al., "Scene classification of high resolution remote sensing images using convolutional neural networks," in *IEEE Geoscience and Remote Sensing Symp. (IGARSS '16)*, pp. 767–770 (2016).
165. F. Del Frate et al., "Use of neural networks for automatic classification from high-resolution images," *IEEE Trans. Geosci. Remote Sens.* **45**(4), 800–809 (2007).
166. Z. Fang, W. Li, and Q. Du, "Using CNN-based high-level features for remote sensing scene classification," in *IEEE Geoscience and Remote Sensing Symp. (IGARSS '16)*, pp. 2610–2613 (2016).
167. Q. Fu et al., "Semi-supervised classification of hyperspectral imagery based on stacked autoencoders," *Proc. SPIE* **10033**, 100332B (2016).
168. J. Geng et al., "High-resolution SAR image classification via deep convolutional autoencoders," *IEEE Trans. Geosci. Remote Sens. Lett.* **12**(11), 2351–2355 (2015).
169. X. Han et al., "Scene classification based on a hierarchical convolutional sparse auto-encoder for high spatial resolution imagery," *Int. J. Remote Sens.* **38**(2), 514–536 (2017).
170. M. He et al., "Hyperspectral image classification based on deep stacking network," in *IEEE Geoscience and Remote Sensing Symp. (IGARSS '16)*, pp. 3286–3289 (2016).
171. B. Hou et al., "Polarimetric SAR images classification using deep belief networks with learning features," in *IEEE Int. Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 2366–2369 (2015).

172. W. Hu et al., "Deep convolutional neural networks for hyperspectral image classification," *J. Sens.* **2015**, 1–12 (2015).
173. M. Iftene, Q. Liu, and Y. Wang, "Very high resolution images classification by fine tuning deep convolutional neural networks," *Proc. SPIE* **10033**, 100332D (2016).
174. P. Jia et al., "Convolutional neural network based classification for hyperspectral data," in *IEEE Geoscience and Remote Sensing Symp. (IGARSS '16)*, pp. 5075–5078 (2016).
175. P. Kontschieder et al., "Deep neural decision forests," in *Proc. of the IEEE Int. Conf. on Computer Vision*, pp. 1467–1475 (2015).
176. M. Längkvist et al., "Classification and segmentation of satellite orthoimagery using convolutional neural networks," *Remote Sens.* **8**(4), 329 (2016).
177. H. Lee and H. Kwon, "Contextual deep CNN based hyperspectral classification," in *IEEE Geoscience and Remote Sensing Symp. (IGARSS '16)*, pp. 3322–3325 (2016).
178. J. Li, "Active learning for hyperspectral image classification with a stacked autoencoders based neural network," in *IEEE Int. Conf. on Image Processing (ICIP '16)*, pp. 1062–1065 (2016).
179. J. Li, L. Bruzzone, and S. Liu, "Deep feature representation for hyperspectral image classification," in *IEEE Int. Geoscience and Remote Sensing Symp. (IGARSS '15)*, pp. 4951–4954 (2015).
180. T. Li, J. Zhang, and Y. Zhang, "Classification of hyperspectral image based on deep belief networks," in *IEEE Int. Conf. on Image Processing (ICIP '14)*, pp. 5132–5136 (2014).
181. W. Li et al., "Hyperspectral image classification using deep pixel-pair features," *IEEE Trans. Geosci. Remote Sens.* **55**(2), 844–853 (2017).
182. Y. Li, W. Xie, and H. Li, "Hyperspectral image reconstruction by deep convolutional neural network for classification," *Pattern Recognit.* **63**, 371–383 (2017).
183. Z. Lin et al., "Spectral–spatial classification of hyperspectral image using autoencoders," in *9th Int. Conf. on Information, Communications and Signal Processing (ICICS '13)*, pp. 1–5, IEEE (2013).
184. Y. Liu et al., "Hyperspectral classification via deep networks and superpixel segmentation," *Int. J. Remote Sens.* **36**(13), 3459–3482 (2015).
185. Y. Liu et al., "Hyperspectral classification via learnt features," in *Int. Conf. on Image Processing (ICIP '15)*, pp. 1–5 (2015).
186. P. Liu, H. Zhang, and K. B. Eom, "Active deep learning for classification of hyperspectral images," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **10**(2), 712–724 (2017).
187. X. Ma, H. Wang, and J. Geng, "Spectral–spatial classification of hyperspectral image based on deep auto-encoder," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **9**(9), 4073–4085 (2016).
188. X. Ma et al., "Hyperspectral image classification with small training set by deep network and relative distance prior," in *IEEE Geoscience and Remote Sensing Symp. (IGARSS '16)*, pp. 3282–3285, IEEE (2016).
189. X. Mei et al., "Infrared ultraspectral signature classification based on a restricted Boltzmann machine with sparse and prior constraints," *Int. J. Remote Sens.* **36**(18), 4724–4747 (2015).
190. S. Mei et al., "Integrating spectral and spatial information into deep convolutional neural networks for hyperspectral classification," in *IEEE Geoscience and Remote Sensing Symp. (IGARSS'15)*, pp. 5067–5070 (2016).
191. A. Merentitis and C. Debes, "Automatic fusion and classification using random forests and features extracted with deep learning," in *IEEE Int. Geoscience and Remote Sensing Symp. (IGARSS '15)*, pp. 2943–2946 (2015).
192. K. Nogueira, O. A. B. Penatti, and J. A. dos Santos, "Towards better exploiting convolutional neural networks for remote sensing scene classification," *Pattern Recognit.* **61**, 539–556 (2017).
193. B. Pan, Z. Shi, and X. Xu, "R-VCANet: a new deep-learning-based hyperspectral image classification method," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **10**(5), 1975–1986 (2017).

194. M. Papadomanolaki et al., "Benchmarking deep learning frameworks for the classification of very high resolution satellite multispectral data," in *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. **3**, no. 7, pp. 83–88 (2016).
195. S. Piramanayagam et al., "Classification of remote sensed images using random forests and deep learning framework," *Proc. SPIE* **10004**, 100040L (2016).
196. F. Qin, J. Guo, and W. Sun, "Object-oriented ensemble classification for polarimetric SAR imagery using restricted Boltzmann machines," *Remote Sens. Lett.* **8**(3), 204–213 (2017).
197. S. Rajan, J. Ghosh, and M. M. Crawford, "An active learning approach to hyperspectral data classification," *IEEE Trans. Geosci. Remote Sens.* **46**(4), 1231–1242 (2008).
198. Z. Wang, N. M. Nasrabadi, and T. S. Huang, "Semisupervised hyperspectral classification using task-driven dictionary learning with Laplacian regularization," *IEEE Trans. Geosci. Remote Sens.* **53**(3), 1161–1173 (2015).
199. J. Yang et al., "Hyperspectral image classification using two-channel deep convolutional neural network," in *IEEE Geoscience and Remote Sensing Symp. (IGARSS '16)*, pp. 5079–5082 (2016).
200. S. Yu, S. Jia, and C. Xu, "Convolutional neural networks for hyperspectral image classification," *Neurocomputing* **219**, 88–98 (2017).
201. J. Yue, S. Mao, and M. Li, "A deep learning framework for hyperspectral image classification using spatial pyramid pooling," *Remote Sens. Lett.* **7**(9), 875–884 (2016).
202. J. Yue et al., "Spectral–spatial classification of hyperspectral images using deep convolutional neural networks," *Remote Sens. Lett.* **6**(6), 468–477 (2015).
203. A. Zeggada and F. Melgani, "Multilabel classification of UAV images with convolutional neural networks," in *IEEE Geoscience and Remote Sensing Symp. (IGARSS '16)*, pp. 5083–5086, IEEE (2016).
204. F. Zhang, B. Du, and L. Zhang, "Scene classification via a gradient boosting random convolutional network framework," *IEEE Trans. Geosci. Remote Sens.* **54**(3), 1793–1802 (2016).
205. H. Zhang et al., "Spectral–spatial classification of hyperspectral imagery using a dual-channel convolutional neural network," *Remote Sens. Lett.* **8**(5), 438–447 (2017).
206. W. Zhao and S. Du, "Learning multiscale and deep representations for classifying remotely sensed imagery," *ISPRS J. Photogramm. Remote Sens.* **113**, 155–165 (2016).
207. Y. Zhong, F. Fei, and L. Zhang, "Large patch convolutional neural networks for the scene classification of high spatial resolution imagery," *J. Appl. Remote Sens.* **10**(2), 025006 (2016).
208. Y. Zhong et al., "SatCNN: satellite image dataset classification using agile convolutional neural networks," *Remote Sens. Lett.* **8**(2), 136–145 (2017).
209. Y. Chen et al., "Deep fusion of hyperspectral and LiDAR data for thematic classification," in *IEEE Geoscience and Remote Sensing Symp. (IGARSS '16)*, pp. 3591–3594 (2016).
210. L. Ran et al., "Bands sensitive convolutional network for hyperspectral image classification," in *Proc. of the Int. Conf. on Internet Multimedia Computing and Service*, pp. 268–272, ACM (2016).
211. J. Zabalza et al., "Novel segmented stacked autoencoder for effective dimensionality reduction and feature extraction in hyperspectral imaging," *Neurocomputing* **185**, 1–10 (2016).
212. Y. Liu and L. Wu, "Geological disaster recognition on optical remote sensing images using deep learning," *Procedia Comput. Sci.* **91**, 566–575 (2016).
213. J. Chen, Q. Jin, and J. Chao, "Design of deep belief networks for short-term prediction of drought index using data in the Huaihe river basin," *Math. Prob. Eng.* **2012**, 235929 (2012).
214. P. Landschützer et al., "A neural network-based estimate of the seasonal to inter-annual variability of the Atlantic Ocean carbon sink," *Biogeosciences* **10**(11), 7793–7815 (2013).
215. M. Shi et al., "Cloud detection of remote sensing images by deep learning," in *IEEE Geoscience and Remote Sensing Symp. (IGARSS '16)*, pp. 701–704, IEEE (2016).
216. X. Shi et al., "Convolutional LSTM network: a machine learning approach for precipitation nowcasting," in *Advances in Neural Information Processing Systems*, C. Cortes et al., Eds., pp. 802–810, Curran Associates, Inc. (2015).

217. S. Lee, H. Zhang, and D. J. Crandall, "Predicting geo-informative attributes in large-scale image collections using convolutional neural networks," in *IEEE Winter Conf. on Applications of Computer Vision (WACV '15)*, pp. 550–557 (2015).
218. Y. Kim and T. Moon, "Human detection and activity classification based on micro-Doppler signatures using deep convolutional neural networks," *IEEE Geosci. Remote Sens. Lett.* **13**(1), 1–5 (2015).
219. W. Ouyang and X. Wang, "A discriminative deep model for pedestrian detection with occlusion handling," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR '12)*, pp. 3258–3265, IEEE (2012).
220. D. Tomè et al., "Deep convolutional neural networks for pedestrian detection," *Signal Process. Image Commun.* **47**, 482–489 (2016).
221. Y. Wei et al., "A universal remote sensing image quality improvement method with deep learning," in *IEEE Geoscience and Remote Sensing Symp. (IGARSS '16)*, pp. 6950–6953 (2016).
222. H. Zhang et al., "Systematic infrared image quality improvement using deep learning based techniques," *Proc. SPIE* **10008**, 100080P (2016).
223. D. Quan et al., "Using deep neural networks for synthetic aperture radar image registration," in *IEEE Geoscience and Remote Sensing Symp. (IGARSS '16)*, pp. 2799–2802 (2016).
224. P. Ghamisi, Y. Chen, and X. X. Zhu, "A self-improving convolution neural network for the classification of hyperspectral data," *IEEE Geosci. Remote Sens. Lett.* **13**(10), 1537–1541 (2016).
225. N. Kussul et al., "Deep learning approach for large scale land cover mapping based on remote sensing data fusion," in *IEEE Geoscience and Remote Sensing Symp. (IGARSS '16)*, pp. 198–201, IEEE (2016).
226. W. Li et al., "Stacked autoencoder-based deep learning for remote-sensing image classification: a case study of African land-cover mapping," *Int. J. Remote Sens.* **37**(23), 5632–5646 (2016).
227. H. Liu et al., "Terrain classification with polarimetric SAR based on deep sparse filtering network," in *IEEE Geoscience and Remote Sensing Symp. (IGARSS '16)*, pp. 64–67 (2016).
228. K. Makantasis et al., "Deep supervised learning for hyper-spectral data classification through convolutional neural networks," in *IEEE Int. Geoscience and Remote Sensing Symp. (IGARSS)*, pp. 4959–4962 (2015).
229. M. Castelluccio et al., "Land use classification in remote sensing images by convolutional neural networks," 1–11, arXiv:1508.00092 (2015).
230. G. Cheng et al., "Effective and efficient midlevel visual elements-oriented land-use classification using VHR remote sensing images," *IEEE Trans. Geosci. Remote Sens.* **53**(8), 4238–4249 (2015).
231. F. P. S. Luus et al., "Multiview deep learning for land-use classification," *IEEE Geosci. Remote Sens. Lett.* **12**(12), 2448–2452 (2015).
232. Q. Lv et al., "Urban land use and land cover classification using remotely sensed SAR data through deep belief networks," *J. Sens.* **2015**, 1–10 (2015).
233. X. Ma, H. Wang, and J. Wang, "Semisupervised classification for hyperspectral image based on multi-decision labeling and deep feature learning," *ISPRS J. Photogramm. Remote Sens.* **120**, 99–107 (2016).
234. M. E. Midhun et al., "Deep model for classification of hyperspectral image using restricted Boltzmann machine," in *Proc. of the 2014 Int. Conf. on Interdisciplinary Advances in Applied Computing (ICONIAAC '14)*, pp. 1–7 (2014).
235. E. Othman et al., "Using convolutional features and a sparse autoencoder for land-use scene classification," *Int. J. Remote Sens.* **37**(10), 2149–2167 (2016).
236. A. B. Penatti, K. Nogueira, and J. A. Santos, "Do deep features generalize from everyday objects to remote sensing and aerial scenes domains?" in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition Workshops*, pp. 44–51 (2015).
237. A. Romero, C. Gatta, and G. Camps-Valls, "Unsupervised deep feature extraction for remote sensing image classification," *IEEE Trans. Geosci. Remote Sens.* **54**(3), 1349–1362 (2016).

238. Y. Sun et al., "Active learning based autoencoder for hyperspectral imagery classification," in *IEEE Geoscience and Remote Sensing Symp. (IGARSS '16)*, pp. 469–472 (2016).
239. N. K. Uba, "Land use and land cover classification using deep learning techniques," PhD Thesis, Arizona State University (2016).
240. L. Alexandre, "3D object recognition using convolutional neural networks with transfer learning between input channels," in *13th Int. Conf. on Intelligent Autonomous Systems*, pp. 889–898, Springer International (2016).
241. X. Chen et al., "Aircraft detection by deep belief nets," in *Proc. 2nd IAPR Asian Conf. on Pattern Recognition (ACPR '13)*, pp. 54–58 (2013).
242. X. Chen and Y. Zhu, "3D object proposals for accurate object class detection," in *Advances in Neural Information Processing Systems*, C. Cortes et al., Eds., pp. 424–432, Curran Associates, Inc. (2015).
243. G. Cheng, P. Zhou, and J. Han, "Learning rotation-invariant convolutional neural networks for object detection in VHR optical remote sensing images," *IEEE Trans. Geosci. Remote Sens.* **54**(12), 7405–7415 (2016).
244. M. Dahmane et al., "Object detection in pleiades images using deep features," in *IEEE Geoscience and Remote Sensing Symp. (IGARSS '16)*, pp. 1552–1555 (2016).
245. W. Diao et al., "Object recognition in remote sensing images using sparse deep belief networks," *Remote Sens. Lett.* **6**(10), 745–754 (2015).
246. G. Georgakis et al., "Multiview RGB-D dataset for object instance detection," in *IEEE Fourth Int. Conf. on 3D Vision (3DV '16)*, pp. 426–434 (2016).
247. D. Maturana and S. Scherer, "3D convolutional neural networks for landing zone detection from LiDAR," in *IEEE Int. Conf. on Robotics and Automation (ICRA '15)*, pp. 3471–3478 (2015).
248. C. Wang and K. Siddiqi, "Differential geometry boosts convolutional neural networks for object detection," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition Workshops*, pp. 51–58 (2016).
249. Q. Wu et al., "Shape-based object extraction in high-resolution remote-sensing images using deep Boltzmann machine," *Int. J. Remote Sens.* **37**(24), 6012–6022 (2016).
250. B. Zhou et al., "Object detectors emerge in deep scene CNNs," http://hdl.handle.net/1721./96942 (2015).
251. P. Ondruska and I. Posner, "Deep tracking: seeing beyond seeing using recurrent neural networks," in *Proc. of the 30th Conf. on Artificial Intelligence (AAAI '16)*, pp. 3361–3367 (2016).
252. G. Masi et al., "Pansharpening by convolutional neural networks," *Remote Sens.* **8**(7), 594 (2016).
253. W. Huang et al., "A new pan-sharpening method with deep neural networks," *IEEE Geosci. Remote Sens. Lett.* **12**(5), 1037–1041 (2015).
254. L. Palafox, A. Alvarez, and C. Hamilton, "Automated detection of impact craters and volcanic rootless cones in mars satellite imagery using convolutional neural networks and support vector machines," in *46th Lunar and Planetary Science Conf.*, pp. 1–2 (2015).
255. M. M. Ghazi, B. Yanikoglu, and E. Aptoula, "Plant identification using deep neural networks via optimization of transfer learning parameters," *Neurocomputing* **235**, 228–235 (2017).
256. H. Guan et al., "Deep learning-based tree classification using mobile LiDAR data," *Remote Sens. Lett.* **6**(11), 864–873 (2015).
257. P. K. Goel et al., "Classification of hyperspectral data by decision trees and artificial neural networks to identify weed stress and nitrogen status of corn," *Comput. Electron. Agric.* **39**(2), 67–93 (2003).
258. K. Kuwata and R. Shibasaki, "Estimating crop yields with deep learning and remotely sensed data," in *IEEE Int. Geoscience and Remote Sensing Symp. (IGARSS '15)*, pp. 858–861 (2015).
259. J. Rebetez et al., "Augmenting a convolutional neural network with local histograms—a case study in crop classification from high-resolution UAV imagery," in *European Symp.*

on Artificial Neural Networks, Computational Intelligence and Machine Learning, pp. 515–520 (2016).
260. S. Sladojevic et al., "Deep neural networks based recognition of plant diseases by leaf image classification," *Comput. Intell. Neurosci.* **2016**, 1–11 (2016).
261. D. Levi, N. Garnett, and E. Fetaya, "StixelNet: a deep convolutional network for obstacle detection and road segmentation," in *Proc. of the British Machine Vision Conf. (BMVC '15)* (2015).
262. P. Li et al., "Road network extraction via deep learning and line integral convolution," in *IEEE Int. Geoscience and Remote Sensing Symp. (IGARSS '16)*, pp. 1599–1602, IEEE (2016).
263. V. Mnih and G. Hinton, "Learning to label aerial images from noisy data," in *Proc. of the 29th Int. Conf. on Machine Learning (ICML-12)*, pp. 567–574 (2012).
264. J. Wang et al., "Road network extraction: a neural-dynamic framework based on deep learning and a finite state machine," *Int. J. Remote Sens.* **36**(12), 3144–3169 (2015).
265. Y. Yu et al., "Automated extraction of 3D trees from mobile LiDAR point clouds," in *ISPRS Int. Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. 40, No. 5, pp. 629–632 (2014).
266. Y. Yu, H. Guan, and Z. Ji, "Automated extraction of urban road manhole covers using mobile laser scanning data," *IEEE Trans. Intell. Transp. Syst.* **16**(4), 2167–2181 (2015).
267. Z. Zhong et al., "Fully convolutional networks for building and road extraction: preliminary results," in *IEEE Geoscience and Remote Sensing Symp. (IGARSS '16)*, pp. 1591–1594 (2016).
268. R. Hadsell et al., "Learning long-range vision for autonomous off-road driving," *J. Field Rob.* **26**(2), 120–144 (2009).
269. L. Mou and X. X. Zhu, "Spatiotemporal scene interpretation of space videos via deep neural network and tracklet analysis," in *IEEE Geoscience and Remote Sensing Symp. (IGARSS '16)*, pp. 1823–1826, IEEE (2016).
270. Y. Yuan, L. Mou, and X. Lu, "Scene recognition by manifold regularized deep learning architecture," *IEEE Trans. Neural Networks Learn. Syst.* **26**(10), 2222–2233 (2015).
271. C. Couprie et al., "Indoor semantic segmentation using depth information," 1–8, arXiv:1301.3572 (2013).
272. Y. Gong et al., "Deep convolutional ranking for multilabel image annotation," 1–9, arXiv:1312.4894 (2013).
273. M. Kampffmeyer, A.-B. Salberg, and R. Jenssen, "Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition Workshops*, pp. 1–9 (2016).
274. P. Kaiser, "Learning city structures from online maps," Master's Thesis, ETH Zurich (2016).
275. A. Lagrange et al., "Benchmarking classification of earth-observation data: from learning explicit features to convolutional networks," in *IEEE Int. Geoscience and Remote Sensing Symp. (IGARSS '15)*, pp. 4173–4176 (2015).
276. D. Marmanis et al., "Deep learning earth observation classification using ImageNet pretrained networks," *IEEE Geosci. Remote Sens. Lett.* **13**(1), 105–109 (2016).
277. D. Marmanis et al., "Semantic segmentation of aerial images with an ensemble of CNNs," in *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. 3, pp. 473–480 (2016).
278. S. Paisitkriangkrai et al., "Effective semantic pixel labelling with convolutional networks and conditional random fields," in *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition Workshops*, pp. 36–43 (2015).
279. B. Qu et al., "Deep semantic understanding of high resolution remote sensing image," in *Int. Conf. on Computer, Information and Telecommunication Systems (CITS '16)*, pp. 1–5 (2016).
280. J. Sherrah, "Fully convolutional networks for dense semantic labelling of high-resolution aerial imagery," arXiv:1606.02585 (2016).

281. C. Vaduva, I. Gavat, and M. Datcu, "Deep learning in very high resolution remote sensing image information mining communication concept," in *Proc. of the 20th European Signal Processing Conf. (EUSIPCO '12)*, pp. 2506–2510, IEEE (2012).
282. M. Volpi and D. Tuia, "Dense semantic labeling of sub-decimeter resolution images with convolutional neural networks," *IEEE Trans. Geosci. Remote Sens.* **55**(2), 881–893 (2017).
283. D. Zhang et al., "Co-saliency detection via looking deep and wide," in *Proc. of the IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, pp. 2994–3002 (2015).
284. F. I. Alam et al., "CRF learning with CNN features for hyperspectral image segmentation," in *IEEE Geoscience and Remote Sensing Symp. (IGARSS '16)*, pp. 6890–6893 (2016).
285. N. Audebert, B. L. Saux, and S. Lefèvre, "How useful is region-based classification of remote sensing images in a deep learning framework?" in *IEEE Geoscience and Remote Sensing Symp. (IGARSS '16)*, pp. 5091–5094 (2016).
286. E. Basaeed et al., "A supervised hierarchical segmentation of remote-sensing images using a committee of multi-scale convolutional neural networks," *Int. J. Remote Sens.* **37**(7), 1671–1691 (2016).
287. E. Basaeed, H. Bhaskar, and M. Al-Mualla, "Supervised remote sensing image segmentation using boosted convolutional neural networks," *Knowl. Based Syst.* **99**, 19–27 (2016).
288. S. Pal, S. Chowdhury, and S. K. Ghosh, "DCAP: a deep convolution architecture for prediction of urban growth," in *IEEE Int. Geoscience and Remote Sensing Symp. (IGARSS '16)*, pp. 1812–1815, IEEE (2016).
289. J. Wang et al., "Deep hierarchical representation and segmentation of high resolution remote sensing images," in *IEEE Int. Geoscience and Remote Sensing Symp. (IGARSS '15)*, pp. 4320–4323 (2015).
290. C. P. Schwegmann et al., "Very deep learning for ship discrimination in synthetic aperture radar imagery," in *IEEE Geoscience and Remote Sensing Symp. (IGARSS '16)*, pp. 104–107, IEEE (2016).
291. J. Tang et al., "Compressed-domain ship detection on spaceborne optical image using deep neural network and extreme learning machine," *IEEE Trans. Geosci. Remote Sens.* **53**(3), 1174–1185 (2015).
292. R. Zhang et al., "S-CNN-based ship detection from high-resolution remote sensing images," in *Int. Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. XLI-B7, pp. 423–430 (2016).
293. Z. Cui et al., "Deep network cascade for image super-resolution," *Lect. Notes Comput. Sci.* **8693**, 49–64 (2014).
294. C. Dong et al., "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.* **38**(2), 295–307 (2016).
295. A. Ducournau and R. Fablet, "Deep learning for ocean remote sensing: an application of convolutional neural networks for super-resolution on satellite-derived SST data," in *9th Workshop on Pattern Recognition in Remote Sensing* (2016).
296. L. Liebel and M. Körner, "Single-image super resolution for multispectral remote sensing data using convolutional neural networks," in *XXIII ISPRS Congress Proc.*, pp. 883–890 (2016).
297. W. Huang et al., "Deep architecture for traffic flow prediction: deep belief networks with multitask learning," *IEEE Trans. Intell. Transp. Syst.* **15**(5), 2191–2201 (2014).
298. Y. Lv et al., "Traffic flow prediction with big data: a deep learning approach," *IEEE Trans. Intell. Transp. Syst.* **16**(2), 865–873 (2015).
299. M. Elawady, "Sparse coral classification using deep convolutional neural networks," PhD Thesis, University of Burgundy, University of Girona, Heriot-Watt University (2014).
300. H. Qin et al., "DeepFish: accurate underwater live fish recognition with a deep architecture," *Neurocomputing* **187**, 49–58 (2016).
301. H. Qin et al., "When underwater imagery analysis meets deep learning: a solution at the age of big visual data," in *7th Int. Symp. on Image and Signal Processing and Analysis (ISPA '15)*, pp. 259–264 (2015).

302. D. P. Williams, "Underwater target classification in synthetic aperture sonar imagery using deep convolutional neural networks," in *Proc. 23rd Int. Conf. on Pattern Recognition (ICPR)* (2016).
303. F. Alidoost and H. Arefi, "Knowledge based 3D building model recognition using convolutional neural networks from Lidar and aerial imageries," in *ISPRS-Int. Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pp. 833–840 (2016).
304. C.-A. Brust et al., "Efficient convolutional patch networks for scene understanding," in *Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9 (2015).
305. S. De and A. Bhattacharya, "Urban classification using PolSAR data and deep learning," in *IEEE Int. Geoscience and Remote Sensing Symp. (IGARSS)*, pp. 353–356, IEEE (2015).
306. Z. Huang et al., "Building extraction from multi-source remote sensing images via deep deconvolution neural networks," in *Geoscience and Remote Sensing Symp. (IGARSS)*, pp. 1835–1838, IEEE (2016).
307. D. Marmanis et al., "Deep neural networks for above-ground detection in very high spatial resolution digital elevation models," in *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences II-3/W4*, pp. 103–110 (2015).
308. S. Saito and Y. Aoki, "Building and road detection from large aerial imagery," *Proc. SPIE* **9405**, 94050K (2015).
309. M. Vakalopoulou et al., "Building detection in very high resolution multispectral data with deep learning features," in *IEEE Int. Geoscience and Remote Sensing Symp. (IGARSS 2015)*, pp. 1873–1876 (2015).
310. M. Xie et al., "Transfer learning from deep features for remote sensing and poverty mapping," 16, arXiv:1510.00098 (2015).
311. W. Yao, P. Poleswki, and P. Krzystek, "Classification of urban aerial data based on pixel labelling with deep convolutional neural networks and logistic regression," *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.* **XLI-B7**, 405–410 (2016).
312. Q. Zhang et al., "CNN based suburban building detection using monocular high resolution Google Earth images," in *IEEE Int. Geoscience and Remote Sensing Symp. (IGARSS 2016)*, pp. 661–664, IEEE (2016).
313. Z. Zhang et al., "A CNN based functional zone classification method for aerial images," in *IEEE Int. Geoscience and Remote Sensing Symp. (IGARSS 2016)*, pp. 5449–5452, IEEE (2016).
314. L. Cao et al., "Robust vehicle detection by combining deep features with exemplar classification," *Neurocomputing* **215**, 225–231 (2016).
315. X. Chen et al., "Vehicle detection in satellite images by hybrid deep convolutional neural networks," *IEEE Geosci. Remote Sens. Lett.* **11**(10), 1797–1801 (2014).
316. K. Goyal and D. Kaur, "A novel vehicle classification model for urban traffic surveillance using the deep neural network model," *Int. J. Educ. Manage. Eng.* **6**(1), 18–31 (2016).
317. A. Hu et al., "Deep Boltzmann machines based vehicle recognition," in *26th Chinese Control and Decision Conf. (2014 CCDC)*, pp. 3033–3038 (2014).
318. J. Huang and S. You, "Vehicle detection in urban point clouds with orthogonal-view convolutional neural network," in *2016 IEEE Int. Conf. on Image Processing (ICIP)*, pp. 2593–2597 (2016).
319. B. Huval et al., "An empirical evaluation of deep learning on highway driving," 1–7, arXiv:1504.01716 (2015).
320. Q. Jiang et al., "Deep neural networks-based vehicle detection in satellite images," in *Int. Symp. on Bioelectronics and Bioinformatics (ISBB)*, pp. 184–187, IEEE (2015).
321. G. V. Konoplich, E. O. Putin, and A. A. Filchenkov, "Application of deep learning to the problem of vehicle detection in UAV images," in *XIX IEEE Int. Conf. on Soft Computing and Measurements (SCM)*, pp. 4–6, IEEE (2016).
322. A. Krishnan and J. Larsson, "Vehicle detection and road scene segmentation using deep learning," PhD Thesis, Chalmers University of Technology (2016).
323. S. Lange, F. Ulbrich, and D. Goehring, "Online vehicle detection using deep neural networks and lidar based preselected image patches," in *Proc. IEEE Intelligent Vehicles Symp.*, pp. 954–959 (2016).

324. B. Li, "3D fully convolutional network for vehicle detection in point cloud," arXiv:1611.08069 (2016).
325. H. Wang, Y. Cai, and L. Chen, "A vehicle detection algorithm based on deep belief network," *Sci. World J.* **2014**, 647380 (2014).
326. J.-G. Wang et al., "Appearance-based brake-lights recognition using deep learning and vehicle detection," in *Intelligent Vehicles Symp.*, pp. 815–820, IEEE (2016).
327. H. Wang et al., "Night-time vehicle sensing in far infrared image with deep learning," *J. Sens.* **2016**, 3403451 (2015).
328. R. J. Firth, "A novel recurrent convolutional neural network for ocean and weather forecasting," PhD Thesis, Louisiana State University (2016).
329. R. Kovordányi and C. Roy, "Cyclone track forecasting based on satellite images using artificial neural networks," *ISPRS J. Photogramm. Remote Sens.* **64**(6), 513–521 (2009).
330. C. Yang and J. Guo, "Improved cloud phase retrieval approaches for China's FY-3A/VIRR multi-channel data using artificial neural networks," *Optik* **127**(4), 1797–1803 (2016).
331. Y. Chen, X. Zhao, and X. Jia, "Spectral–spatial classification of hyperspectral data based on deep belief network," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **8**(6), 2381–2392 (2015).
332. J. M. P. Nascimento and J. M. B. Dias, "Vertex component analysis: a fast algorithm to unmix hyperspectral data," *IEEE Trans. Geosci. Remote Sens.* **43**(4), 898–910 (2005).
333. T. H. Chan et al., "PCANet: a simple deep learning baseline for image classification?," *IEEE Trans. Image Process.* **24**(12), 5017–5032 (2015).
334. Y. Chen et al., "Deep feature extraction and classification of hyperspectral images based on convolutional neural networks," *IEEE Trans. Geosci. Remote Sens.* **54**(10), 6232–6251 (2016).
335. J. Donahue et al., "DeCAF: a deep convolutional activation feature for generic visual recognition," *ICML* **32**, 647–655 (2014).
336. A. Zelener and I. Stamos, "CNN-based object segmentation in urban lidar with missing points," in *Fourth Int. Conf. on 3D Vision (3DV)*, pp. 417–425, IEEE (2016).
337. J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 3431–3440 (2015).
338. J. E. Ball, D. T. Anderson, and S. Samiappan, "Hyperspectral band selection based on the aggregation of proximity measures for automated target detection," *Proc. SPIE* **9088**, 908814 (2014).
339. J. E. Ball and L. M. Bruce, "Level set hyperspectral image classification using best band analysis," *IEEE Trans. Geosci. Remote Sens.* **45**(10), 3022–3027 (2007).
340. J. E. Ball et al., "Level set hyperspectral image segmentation using spectral information divergence-based best band selection," in *IEEE Int. Geoscience and Remote Sensing Symp. (IGARSS 2007)*, pp. 4053–4056, IEEE (2007).
341. D. T. Anderson and A. Zare, "Spectral unmixing cluster validity index for multiple sets of endmembers," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **5**(4), 1282–1295 (2012).
342. M. E. Winter, "Comparison of approaches for determining end-members in hyperspectral data," in *Aerospace Conf. Proc.*, pp. 305–313, IEEE (2000).
343. A. S. Charles, B. A. Olshausen, and C. J. Rozell, "Learning sparse codes for hyperspectral imagery," *IEEE J. Sel. Top. Signal Process.* **5**(5), 963–978 (2011).
344. A. Romero, C. Gatta, and G. Camps-Valls, "Unsupervised deep feature extraction of hyperspectral images," in *Proc. 6th Workshop Hyperspectral Image Signal Processing Evaluation Remote Sensing (WHISPERS)* (2014).
345. J. E. Ball, L. M. Bruce, and N. H. Younan, "Hyperspectral pixel unmixing via spectral band selection and DC-insensitive singular value decomposition," *IEEE Geosci. Remote Sens. Lett.* **4**(3), 382–386 (2007).
346. P. M. Atkinson and A. Tatnall, "Introduction neural networks in remote sensing," *Int. J. Remote Sens.* **18**(4), 699–709 (1997).

347. G. Cavallaro et al., "On understanding big data impacts in remotely sensed image classification using support vector machine methods," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **8**(10), 4634–4646 (2015).
348. M. Egmont-Petersen, D. DeRidder, and H. Handels, "Image processing with neural networks—a review," *Pattern Recognit.* **35**, 2279–2301 (2002).
349. G. Huang et al., "Trends in extreme learning machines: a review," *Neural Networks* **61**, 32–48 (2015).
350. X. Jia, B.-C. Kuo, and M. M. Crawford, "Feature mining for hyperspectral image classification," *Proc. IEEE* **101**(3), 676–697 (2013).
351. D. Lu and Q. Weng, "A survey of image classification methods and techniques for improving classification performance," *Int. J. Remote Sens.* **28**(5), 823–870 (2007).
352. H. Petersson, D. Gustafsson, and D. Bergstrom, "Hyperspectral image analysis using deep learning—a review," in *6th Int. Conf. on Image Processing Theory Tools and Applications (IPTA)*, pp. 1–6, IEEE (2016).
353. A. Plaza et al., "Recent advances in techniques for hyperspectral image processing," *Remote Sens. Environ.* **113**, S110–S122 (2009).
354. W. Wang et al., "A review of road extraction from remote sensing images," *J. Traffic Transp. Eng.* **3**(3), 271–282 (2016).
355. "IEEE GRSS data fusion contest," http://www.grss-ieee.org/community/technical-committees/data-fusion/ (20 November 2016).
356. "Indian Pines dataset," http://dynamo.ecn.purdue.edu/biehl/MultiSpec (20 November 2016).
357. "Kennedy Space Center," http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes#Kennedy_Space_Center_.28KSC.29 (20 November 2016).
358. "Pavia dataset," http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes#Pavia_Centre_and_University (20 November 2016).
359. "Salinas dataset," http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes#Salinas (20 November 2016).
360. "Washington DC Mall," https://engineering.purdue.edu/biehl/MultiSpec/hyperspectral.html (20 November 2016).
361. G. Cheng and J. Han, "A survey on object detection in optical remote sensing images," *ISPRS J. Photogramm. Remote Sens.* **117**, 11–28 (2016).
362. Y. Chen et al., "Deep learning-based classification of hyperspectral data," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **7**(6), 2094–2107 (2014).
363. V. Slavkovikj et al., "Hyperspectral image classification with convolutional neural networks," in *Proc. of the 23rd ACM Int. Conf. on Multimedia*, pp. 1159–1162, ACM (2015).
364. C. Tao et al., "Unsupervised spectral–spatial feature learning with stacked sparse autoencoder for hyperspectral imagery classification," *IEEE Geosci. Remote Sens. Lett.* **12**(12), 2438–2442 (2015).
365. Y. LeCun, "Learning invariant feature hierarchies," in *Computer Vision (ECCV 2012)*, pp. 496–505, Springer (2012).
366. M. Pal, "Kernel methods in remote sensing: a review," *ISH J. Hydraul. Eng.* **15**(Suppl. 1), 194–215 (2009).
367. E. M. Abdel-Rahman and F. B. Ahmed, "The application of remote sensing techniques to sugarcane (*Saccharum* spp. hybrid) production: a review of the literature," *Int. J. Remote Sens.* **29**(13), 3753–3767 (2008).
368. I. Ali et al., "Review of machine learning approaches for biomass and soil moisture retrievals from remote sensing data," *Remote Sens.* **7**(12), 16398–16421 (2015).
369. E. Adam, O. Mutanga, and D. Rugege, "Multispectral and hyperspectral remote sensing for identification and mapping of wetland vegetation: a review," *Wetlands Ecol. Manage.* **18**(3), 281–296 (2010).
370. S. L. Ozesmi and M. E. Bauer, "Satellite remote sensing of wetlands," *Wetlands Ecol. Manage.* **10**(5), 381–402 (2002).
371. W. A. Dorigo et al., "A review on reflective remote sensing and data assimilation techniques for enhanced agroecosystem modeling," *Int. J. Appl. Earth Obs. Geoinf.* **9**(2), 165–193 (2007).

372. C. Kuenzer et al., "Earth observation satellite sensors for biodiversity monitoring: potentials and bottlenecks," *Int. J. Remote Sens.* **35**(18), 6599–6647 (2014).
373. K. Wang et al., "Remote sensing of ecology, biodiversity and conservation: a review from the perspective of remote sensing specialists," *Sensors* **10**(11), 9647–9667 (2010).
374. F. E. Fassnacht et al., "Review of studies on tree species classification from remotely sensed data," *Remote Sens. Environ.* **186**, 64–87 (2016).
375. A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 5188–5196 (2015).
376. J. Yosinski et al., "Understanding neural networks through deep visualization," arXiv:1506.06579 (2015).
377. K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: visualising image classification models and saliency maps," arXiv:1312.6034 (2013).
378. D. Erhan et al., "Visualizing higher-layer features of a deep network," *Univ. Montreal* **1341**, 3 (2009).
379. J. A. Benediktsson, J. Chanussot, and W. W. Moon, "Very high-resolution remote sensing: challenges and opportunities," *Proc. IEEE* **100**(6), 1907–1910 (2012).
380. J. Fohringer et al., "Social media as an information source for rapid flood inundation mapping," *Nat. Hazards Earth Syst. Sci.* **15**(12), 2725–2738 (2015).
381. V. Frias-Martinez and E. Frias-Martinez, "Spectral clustering for sensing urban land use using Twitter activity," *Eng. Appl. Artif. Intell.* **35**, 237–245 (2014).
382. T. Kohonen, "The self-organizing map," *Neurocomputing* **21**(1), 1–6 (1998).
383. S. E. Middleton, L. Middleton, and S. Modafferi, "Real-time crisis mapping of natural disasters using social media," *IEEE Intell. Syst.* **29**(2), 9–17 (2014).
384. V. K. Singh, "Social pixels: genesis and evaluation," in *Proc. of the 18th ACM Int. Conf. on Multimedia (ACMM)*, pp. 481–490 (2010).
385. D. Sui and M. Goodchild, "The convergence of GIS and social media: challenges for GIScience," *Int. J. Geogr. Inf. Sci.* **25**(11), 1737–1748 (2011).
386. A. J. Pinar et al., "Efficient multiple kernel classification using feature and decision level fusion," *IEEE Trans. Fuzzy Syst.* **PP**(99), 1–1 (2016).
387. D. T. Anderson et al., "Extension of the fuzzy integral for general fuzzy set-valued information," *IEEE Trans. Fuzzy Syst.* **22**, 1625–1639 (2014).
388. P. Ghamisi, B. Hfle, and X. X. Zhu, "Hyperspectral and lidar data fusion using extinction profiles and deep convolutional neural network," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **PP**(99), 1–14 (2016).
389. J. Ngiam et al., "Multimodal deep learning," in *Int. Conf. on Machine Learning (ICML)*, pp. 689–696, Omnipress (2011).
390. Z. Kira et al., "Long-range pedestrian detection using stereo and a cascade of convolutional network classifiers," in *IEEE Int. Conf. on Intelligent Robots and Systems*, pp. 2396–2403 (2012).
391. F. Rottensteiner et al., "The ISPRS benchmark on urban object classification and 3D building reconstruction," *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **I-3**, 293–298 (2012).
392. J. Ngiam et al., "Sparse filtering," in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor et al., Eds., pp. 1125–1133, Curran Associates, Inc. (2011).
393. Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *European Conf. on Computational Learning Theory*, pp. 23–37, Springer (1995).
394. Q. Zhao et al., "Three-class change detection in synthetic aperture radar images based on deep belief network," in *Bio-Inspired Computing-Theories and Applications*, pp. 696–705, Springer (2015).
395. A. P. Tewkesbury et al., "A critical synthesis of remotely sensed optical image change detection techniques," *Remote Sens. Environ.* **160**, 1–14 (2015).
396. C. Brekke and A. H. S. Solberg, "Oil spill detection by satellite remote sensing," *Remote Sens. Environ.* **95**(1), 1–13 (2005).
397. H. Mayer, "Automatic object extraction from aerial imagery a survey focusing on buildings," *Comput. Vision Image Understanding* **74**(2), 138–149 (1999).

398. B. Somers et al., "Endmember variability in spectral mixture analysis: a review," *Remote Sens. Environ.* **115**(7), 1603–1616 (2011).
399. D. Tuia, C. Persello, and L. Bruzzone, "Domain adaptation for the classification of remote sensing data: an overview of recent advances," *IEEE Geosci. Remote Sens. Mag.* **4**(2), 41–57 (2016).
400. S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.* **22**(10), 1345–1359 (2010).
401. Y. Yang and S. Newsam, "Bag-of-visual-words and spatial extensions for land-use classification," in *Proc. of the 18th SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems*, pp. 270–279, ACM (2010).
402. V. Risojevic, S. Momic, and Z. Babic, "Gabor descriptors for aerial image classification," in *Int. Conf. on Adaptive and Natural Computing Algorithms*, pp. 51–60, Springer (2011).
403. K. Chatfield et al., "Return of the devil in the details: delving deep into convolutional nets," arXiv:1405.3531 (2014).
404. G. Sheng et al., "High-resolution satellite scene classification using a sparse coding based multiple feature combination," *Int. J. Remote Sens.* **33**(8), 2395–2412 (2012).
405. S. H. Lee et al., "How deep learning extracts and learns leaf features for plant classification," *Pattern Recognit.* **71**, 1–13 (2017).
406. A. Joly et al., "LifeCLEF 2016: multimedia life species identification challenges," in *Int. Conf. of the Cross-Language Evaluation Forum for European Languages*, pp. 286–310, Springer (2016).
407. S. H. Lee et al., "Deep-plant: plant identification with convolutional neural networks," in *IEEE Int. Conf. on Image Processing (ICIP 2015)*, pp. 452–456 (2015).
408. Z. Ding, N. Nasrabadi, and Y. Fu, "Deep transfer learning for automatic target classification: MWIR to LWIR," *Proc. SPIE* **9844**, 984408 (2016).
409. X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. of the Thirteenth Int. Conf. on Artificial Intelligence and Statistics*, pp. 249–256 (2010).
410. J. Sokolic et al., "Robust large margin deep neural networks," arXiv:1605.08254 (2016).
411. D. Erhan, A. Courville, and P. Vincent, "Why does unsupervised pre-training help deep learning?," *J. Mach. Learn. Res.* **11**, 625–660 (2010).
412. K. He et al., "Delving deep into rectifiers: surpassing human-level performance on ImageNet classification," in *Proc. of the IEEE Int. Conf. on Computer Vision*, pp. 1026–1034 (2015).
413. N. Srivastava et al., "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.* **15**(1), 1929–1958 (2014).
414. S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," arXiv:1502.03167 (2015).
415. Q. V. Le et al., "On optimization methods for deep learning," in *Proc. of the 28th Int. Conf. on Machine Learning (ICML)*, pp. 265–272 (2011).
416. S. Ruder, "An overview of gradient descent optimization algorithms," arXiv:1609.04747 (2016).
417. M. D. Zeiler, "Adadelta: an adaptive learning rate method," arXiv:1212.5701 (2012).
418. T. Schaul, S. Zhang, and Y. LeCun, "No more pesky learning rates," in *Int. Conf. on Machine Learning (ICML)*, pp. 343–351 (2013).
419. D. Balduzzi, B. McWilliams, and T. Butler-Yeoman, "Neural Taylor approximations: convergence and exploration in rectifier networks," arXiv:1611.02345 (2016).

John E. Ball is an assistant professor of electrical and computer engineering at Mississippi State University (MSU), USA. He received his PhD from MSU in 2007. He is a codirector of the Sensor Analysis and Intelligence Laboratory (SAIL) at MSU. He has authored more than 45 articles, and 22 technical and reports. His research interests are deep learning, remote sensing, and signal/image processing. He is an associate editor for the SPIE *Journal of Applied Remote Sensing*.

Derek T. Anderson received his PhD from the University of Missouri in 2010. He is an associate professor and the Robert D. Guyton Chair in ECE at Mississippi State University, with the Naval Research Laboratory, and codirects the Sensor Analysis and Intelligence Laboratory. His research includes data/information fusion for machine learning and automated decision making in computer vision. He has published more than 110 articles, and he is an AE for IEEE TFS and program cochair of FUZZ-IEEE 2019.

Chee Seng Chan received his PhD from the University of Portsmouth, UK, in 2008. He is currently a senior lecturer in the Department of Artificial Intelligence, Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia. His research interests include computer vision and fuzzy qualitative reasoning, with an emphasis on image/video understanding. He was a recipient of the Hitachi Research Fellowship in 2013 and Young Scientist Network-Academy of Sciences Malaysia in 2015.