Hindawi
Journal of Sensors
Volume 2017, Article ID 3296874, 13 pages
https://doi.org/10.1155/2017/3296874

Review Article

A Review of Deep Learning Methods and Applications for Unmanned Aerial Vehicles

Adrian Carrio, Carlos Sampedro, Alejandro Rodriguez-Ramos, and Pascual Campoy

Computer Vision and Aerial Robotics Group, Centre for Automation and Robotics (CAR) UPM-CSIC, Universidad Politécnica de Madrid, Calle José Gutiérrez Abascal 2, 28006 Madrid, Spain

Correspondence should be addressed to Adrian Carrio; adrian.carrio@upm.es

Received 28 April 2017; Accepted 18 June 2017; Published 14 August 2017

Academic Editor: Vera Tyrsa

Deep learning is recently showing outstanding results for solving a wide variety of robotic tasks in the areas of perception, planning, localization, and control. Its excellent capabilities for learning representations from the complex data acquired in real environments make it extremely suitable for many kinds of autonomous robotic applications. In parallel, Unmanned Aerial Vehicles (UAVs) are currently being extensively applied for several types of civilian tasks in applications going from security, surveillance, and disaster rescue to parcel delivery or warehouse management. In this paper, a thorough review has been performed on recent reported uses and applications of deep learning for UAVs, including the most relevant developments as well as their performances and limitations. In addition, a detailed explanation of the main deep learning techniques is provided. We conclude with a description of the main challenges for the application of deep learning for UAV-based solutions.

1. Introduction

Recent successes of deep learning techniques in solving many complex tasks by learning from raw sensor data have created a lot of excitement in the research community. However, deep learning is not a recent technology. It started being used back in 1971, when Ivakhnenko [1] trained an 8-layer neural network using the Group Method of Data Handling (GMDH) algorithm. The term deep learning began to be used during the 2000s, when Convolutional Neural Networks (CNNs), a computational original model from the 80s [2] but trained efficiently in the 90s [3], were able to provide decent results in visual object recognition tasks. At the time, datasets were small and computers were not powerful enough, so the performance was often similar to or worse than that of classical Computer Vision algorithms. The development of CUDA for Nvidia GPUs which enabled over 1000 GFLOPS per second and the publication of the ImageNet dataset, with 1.2 million images classified in 1000 categories [4], were important facts for the popularization of CNNs with several layers (10^9 to 10^{10} connections and 10^7 to 10^9 parameters). These deep models show great performance not only in Computer Vision tasks but also in other tasks such as speech recognition, signal processing, and natural language processing [5]. More details about recent advances in deep learning can be found in [6, 7].

An evidence of the suitability of deep learning for many kinds of autonomous robotic applications is the increasing trend in *deep learning robot* related scientific publications over the past decades, which is expected to continue growing [8].

Due to the versatility, automation capabilities, and low cost of Unmanned Aerial Vehicles (UAVs), civilian applications in diverse fields have experienced a drastic increase during the last years. Some examples include power line inspection [9], wildlife conservation [10], building inspection [11], and precision agriculture [12]. However, UAVs have limitations in the size, weight, and power consumption of the payload and limited range and endurance. These limitations cannot be overlooked and are particularly relevant when deep learning algorithms are required to run on board a UAV.

In this survey, we have grouped publications according to the taxonomy proposed in Aerostack [13], which is aerial robotics architecture consistent with the usual components

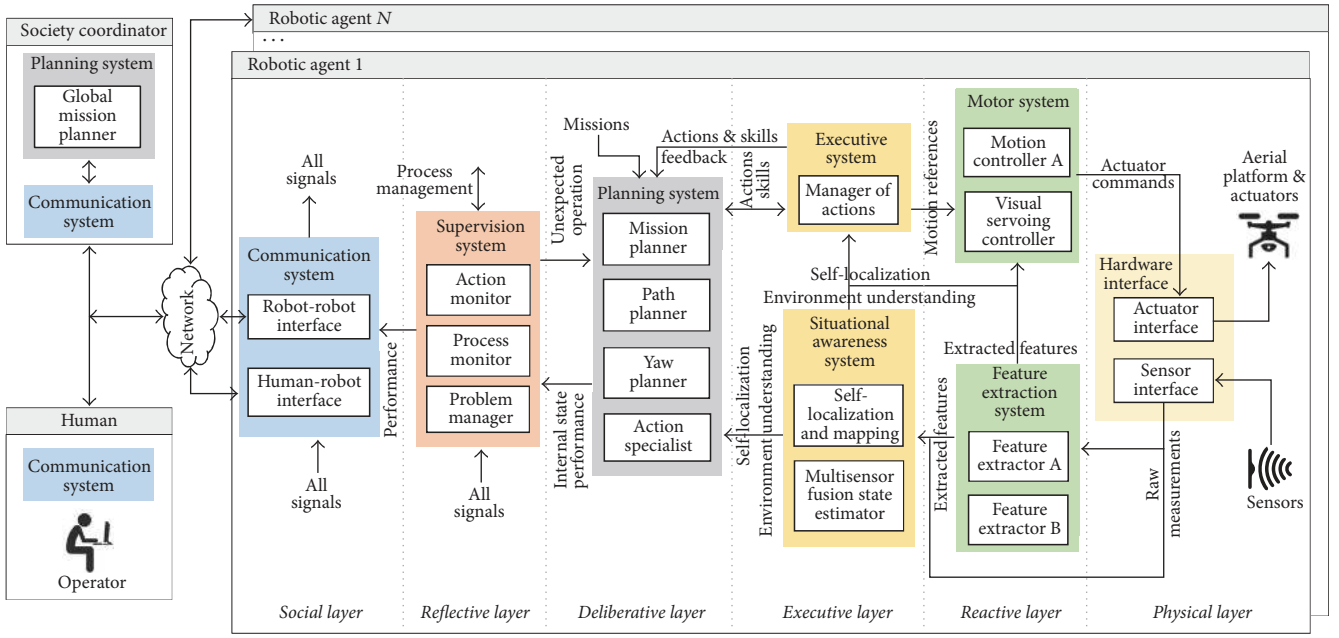

Figure 1: Aerostack architecture, consisting of a layered structure, corresponding to the different abstraction levels in an unmanned aerial robotic system. The architecture has been applied here to systematically classify deep learning-based algorithms available in the state of the art which have been deployed for applications with Unmanned Aerial Vehicles.

related to perception, guidance, navigation, and control of unmanned rotorcraft systems. The purpose of referring to this architecture, depicted in Figure 1, is to achieve a better understanding about the nature of the components to the aerial robotic systems analyzed. Using this taxonomy also helps identify the components in which deep learning has not been applied yet. According to Aerostack, the components constituting an unmanned aerial robotic system can be classified into the following systems and interfaces:

(i) *Hardware interfaces*: this category includes interfaces with both sensors and actuators

(ii) *Motor system*: the components of a motor system are motion controllers, which typically receive commands of desired values for a variable (position, orientation, or speed). These desired values are translated into low-level commands that are sent to actuators

(iii) *Feature extraction system*: feature extraction here refers to the extraction of useful features or representations from sensor data. The task of most deep learning algorithms is to learn data representations, so feature extraction systems are somewhat inherent to deep learning algorithms

(iv) *Situational awareness system*: this system includes components that compile sensor information into state variables regarding the robot and its environment, pursuing environment understanding. An example component within the situational awareness system is SLAM algorithms

(v) *Executive system*: this system receives high-level symbolic actions and generates detailed behaviour sequences

(vi) *Planning system*: this type of system generates global solutions to complex tasks by means of planning (e.g., path planning and mission planning)

(vii) *Supervision system*: components in the supervision system simulate self-awareness in the sense of ability to supervise other integrated systems. We can exemplify this type of component with an algorithm that checks whether the robot is actually making progress towards its goal and reacts in the presence of problems (unexpected obstacles, faults, etc.) with recovery actions

(viii) *Communication system*: the components in the communication system are responsible for establishing an adequate communication with human operators and/or other robots

The remainder of this paper is as follows: firstly, Section 2 covers a description of the currently relevant and prominent deep learning algorithms. For the sake of completeness, deep learning algorithms have been included regardless of their direct use in UAV applications. Section 3 presents the state of the art in deep learning for feature extraction in UAV applications. Section 4 surveys UAV applications of deep learning for the development of components of planning and situation awareness systems. Reported applications of deep learning for motion control in UAVs are presented in Section 5. Finally, a discussion of the main challenges for the application of deep learning for UAVs is covered in Section 6.

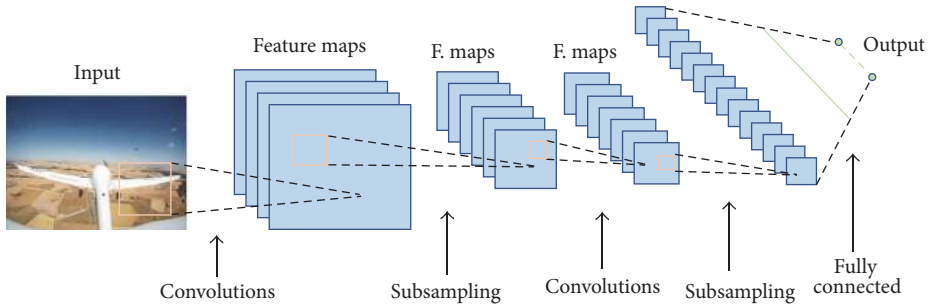

Figure 2: A generic example of a Convolutional Neural Network model. The usual architecture alternates convolution and subsampling layers. Fully connected neurons are used in the last layers.

2. Deep Learning in the Context of Machine Learning

Machine Learning is a capability enabling Artificial Intelligence (AI) systems to learn from data. A good definition for what learning involves is the following: "a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E" [15]. The nature of this experience E is typically considered for classifying Machine Learning algorithms into the following three categories: supervised, unsupervised, and reinforcement learning:

(i) In supervised learning, algorithms are presented with a dataset containing a collection of features. Additionally, labels or target values are provided for each sample. This mapping of features to labels of target values is where the knowledge is encoded. Once it has learned, the algorithm is expected to find the mapping from the features of unseen samples to their correct labels or target values.

(ii) The purpose in unsupervised learning is to extract meaningful representations and explain key features of the data. No labels or target values are necessary in this case in order to learn from the data.

(iii) In reinforcement learning algorithms, an AI agent interacts with a real or simulated environment. This interaction provides feedback between the learning system and the interaction experience which is useful to improve performance in the task being learned.

Deep learning algorithms are a subset of Machine Learning algorithms that typically involve learning representations at different hierarchy levels to enable building complex concepts out of simpler ones. The following paragraphs cover the most relevant deep learning technologies currently available in supervised, unsupervised, and reinforcement learning.

2.1. Supervised Learning. Supervised learning algorithms learn how to associate an input with some output, given a training set of examples of inputs and outputs [16]. The following paragraphs cover the most relevant algorithms nowadays in supervised learning: Feedforward Neural Networks, a popular variation of these called Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and a variation of RNNs called Long Short-Term Memory (LSTM) models.

Feedforward Neural Networks, also known as Multilayer Perceptrons (MLPs), are the most common supervised learning models. Their purpose is to work as function approximators: given a sample vector $\mathbf{x}$ with n features, a trained algorithm is expected to produce an output value or classification category $\mathbf{y}$ that is consistent with the mapping of inputs and outputs provided in the training set. The approximated function is usually built by stacking together several hidden layers that are activated in chain to obtain the desired output. The number of hidden layers is usually referred to as the depth of the model, which explains the origin of the term deep learning: learning using models with several layers. These layers are made up of neurons or units whose activation given an input vector $x \in \mathbb{R}^n$ is given by the following equation:

$$a_\theta(x) = g\left(\theta^T x\right), \tag{1}$$

where θ is a vector of n weights and g is an activation function that is usually chosen to be nonlinear. The activation of unit k in layer m given its n inputs (outputs of the previous layer $m-1$) is given by the following equation:

$$a_k^m = g\left(\Theta_{k0}^{m-1} a_0^{m-1} + \Theta_{k1}^{m-1} a_1^{m-1} + \cdots + \Theta_{kn}^{m-1} a_n^{m-1}\right). \tag{2}$$

During the process of learning, the weights in each unit are updated using backpropagation in order to optimize a cost function, which generally indicates the similarity between the desired outputs and the actual ones.

Convolutional Neural Networks (CNNs), depicted in Figure 2, are a specific type of models conceived to accept 2-dimensional input data, such as images or time series data. These models take their name from the mathematical linear operation of convolution which is always present in at least one of the layers of the network. The most typical convolution

operation used in deep learning is 2D convolution of a 2-dimensional image I with a 2-dimensional kernel K, given by the following equation:

$$C(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n) K(i-m, j-n). \quad (3)$$

The output of the convolution operation is usually run through a nonlinear activation function and then further modified by means of a pooling function, which replaces the output in a certain location with a value obtained from nearby outputs. This pooling function helps make the representation learned invariant to small translations of the input and performs subsampling of the input data. The most common pooling function is max pooling, which replaces the output with the maximum activation within a rectangular neighborhood. Convolution and pooling layers are stacked together to achieve feature learning in a hierarchical way. For example, when learning from images, layers closer to the input learn low-level feature representations (i.e., edges and corners) and those closer to the output learn higher level representations (i.e., contours and parts of objects). Once the features of interest have been learned, their activations are used in final layers, which are usually made up of fully connected neurons, to classify the input or perform value regression with it.

In contrast to MLPs, Recurrent Neural Networks (RNNs) are models in which the output is a function of not only the current inputs but also of the previous outputs, which are encoded into a hidden state h. This means that RNNs have memory of the previous outputs and therefore can encode the information present in the sequence itself, something that MLPs cannot do. As a consequence, this type of model can be very useful to learn from sequential data. The memory is encoded into an internal state and updated as indicated in the following equation:

$$h_t = g[Wx_t + Uh_{t-1}], \quad (4)$$

where h_t represents the hidden state at time step t. The weight matrices W (input-to-hidden) and U (hidden-to-hidden) determine the importance given to the current input and to the previous state, respectively. The activation is computed with a third weight matrix V (hidden-to-output) as indicated by the following equation:

$$a_t = Vh_t. \quad (5)$$

RNNs are usually trained using Backpropagation Through Time (BPTT), an extension of backpropagation which takes into account temporality in order to compute the gradients. Using this method with long temporal sequences can lead to several issues. Gradients accumulated over a long sequence can become immeasurably large or extremely small. These problems are referred to as exploding gradients and vanishing gradients, respectively. Exploding gradients are easier to solve, as they can be truncated or squashed, whereas vanishing gradients can become too small for networks to learn from and for the resolution of a computer to enable its representation.

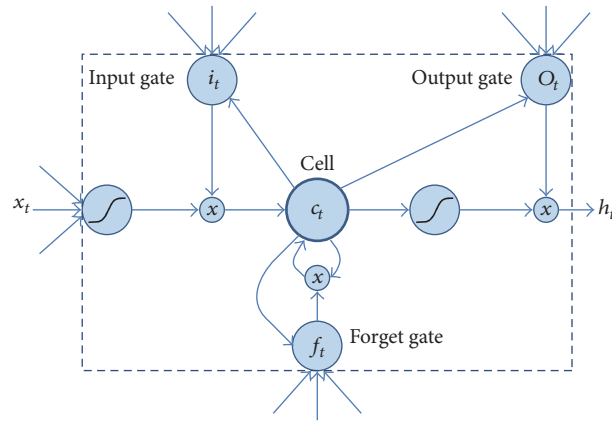

Figure 3: A long-short term memory model, adapted from the original figure in [14]. Learned weights control how data enter and leave and are deleted through the use of gates.

Long Short-Term Memory (LSTM) models are a type of RNN architecture proposed in 1997 by Hochreiter and Schmidhuber [17] which successfully overcomes the problem of vanishing gradients by maintaining a more constant error through the use of gated cells, which effectively allow for continuous learning over a larger number of time steps. A typical LSTM cell is depicted in Figure 3. The input, output, and forget gate vector activations in a standard LSTM are given as follows:

$$\begin{aligned} i_t &= g(W_i x_t + U_i h_{t-1}), \\ o_t &= g(W_o x_t + U_o h_{t-1}), \\ f_t &= g(W_f x_t + U_f h_{t-1}). \end{aligned} \quad (6)$$

The cell state vector activation is given by the following equation:

$$c_t = f_t \circ c_{t-1} + i_t \circ g(W_c x_t + U_c h_{t-1}), \quad (7)$$

where $\circ$ represents the Hadamard product. Finally, the output gate vector activation is given by the following equation:

$$h_t = o_t \circ g(c_t). \quad (8)$$

As it has been already stated, LSTM gated cells in RNNs have internal recurrence, besides the outer recurrence of RNNs. Cells store an internal state, which can be written to and read from them. There are gates controlling how data enter and leave and are deleted from this cell state. Those gates act on the signals they receive, and, similar to a standard neural network, they block or pass on information based on its strength and importance using their own sets of weights. Those weights, as the weights that modulate input and hidden states, are adjusted via the recurrent network's learning process. The cells learn when to allow data to enter and leave or be deleted through the iterative process of making guesses,

backpropagating error, and adjusting weights via gradient descent. This type of model architecture allows successful learning from long sequences, helping to capture diverse time scales and remote dependencies. Practical aspects on the use of LSTMs and other deep learning architectures can be found in [18].

2.2. Unsupervised Learning. Unsupervised learning aims towards the development of models that are capable of extracting meaningful and high-level representations from high-dimensional sensory unlabeled data. This functionality is inspired by the visual cortex which requires very small amount of labeled data.

Deep Generative Models such as Deep Belief Networks (DBNs) [19, 20] allow the learning of several layers of nonlinear features in an unsupervised manner. DBNs are built by stacking several Restricted Boltzmann Machines (RBMs) [21, 22], resulting in a hybrid model in which the top two layers form a RBM and the bottom layers act as a directed graph constituting a Sigmoid Belief Network (SBN). The learning algorithm proposed in [19] is supposed to be one of the first efficient ways of learning DBNs by introducing a greedy layer-by-layer training in order to obtain a deep hierarchical model. In this greedy learning procedure, the hidden activity patterns obtained in the current layer are used as the "visible" data for training the RBM of the next layer. Once the stacked RBMs have been learned and combined to form a DBN, a fine-tuning procedure using a contrastive version of the wake-sleep algorithm [23] is applied.

For a better understanding, the theoretical details of RBMs are provided in the following equations. The energy of a joint configuration $\{\mathbf{v}, \mathbf{h}\}$ can be calculated as follows:

$$E(\mathbf{v}, \mathbf{h}; \theta) = -\sum_{i \in \text{vis}} v_i b_i - \sum_{j \in \text{hid}} h_j a_j - \sum_{i,j} W_{ij} v_i h_j, \tag{9}$$

where $\theta = \{W, b, a\}$ represent the model parameters. $\mathbf{v} \in \{0, 1\}$ are the "visible" stochastic binary units, which are connected to the "hidden" stochastic binary units $\mathbf{h} \in \{0, 1\}$. The bias terms are denoted by b_i for the visible units and a_j for the hidden units.

The probability of a joint configuration over both visible and hidden units depends on the energy of that joint configuration and is given by (10), where $Z(\theta)$ represents the partition function (see (11)):

$$P(\mathbf{v}, \mathbf{h}; \theta) = \frac{1}{Z(\theta)} \exp\left(-E(\mathbf{v}, \mathbf{h}; \theta)\right), \tag{10}$$

$$Z(\theta) = \sum_{\mathbf{v}} \sum_{\mathbf{h}} \left(\exp\left(-E(\mathbf{v}, \mathbf{h}; \theta)\right)\right). \tag{11}$$

The probability assigned by the model to a visible vector $\mathbf{v}$ can be computed as expressed in the following equation:

$$P(\mathbf{v}; \theta) = \frac{1}{Z(\theta)} \sum_{h} \exp\left(-E(\mathbf{v}, \mathbf{h}; \theta)\right). \tag{12}$$

The conditional distributions over hidden variables $\mathbf{h}$ and visible variables $\mathbf{v}$ can be extracted using (13). Once a training sample is presented to the model, the binary states of the hidden variables are set to 1 with probability given by (14). Analogously, once the binary states of the hidden variables are computed, the binary states of the visible units are set to 1 with a probability given by (15).

$$\begin{aligned} P(\mathbf{h} \mid \mathbf{v}; \theta) &= \prod_j p\left(h_j \mid v\right), \\ P(\mathbf{v} \mid \mathbf{h}; \theta) &= \prod_i p\left(v_i \mid h\right), \end{aligned} \tag{13}$$

$$p\left(h_j = 1 \mid v\right) = \sigma\left(\sum_i W_{ij} v_i + a_j\right), \tag{14}$$

$$p\left(v_i = 1 \mid h\right) = \sigma\left(\sum_j W_{ij} h_j + b_i\right), \tag{15}$$

where $\sigma(z) = 1/1 + \exp(-z)$ is the logistic function.

For training the RBM model, the learning is conducted by applying the Contrastive Divergence algorithm [22], in which the update rule applied to the model parameters is given by the following equation:

$$\Delta W_{ij} = \epsilon\left(\left\langle v_i h_j \right\rangle_{\text{data}} - \left\langle v_i h_j \right\rangle_{\text{recons}}\right), \tag{16}$$

where ϵ is the learning rate, $\langle v_i h_j \rangle_{\text{data}}$ represents the expected value of the product of visible and hidden states at thermal equilibrium, when training data is presented to the model, and $\langle v_i h_j \rangle_{\text{recons}}$ is the expected value of the product of visible and hidden states after running a Gibbs chain.

Deep neural networks can also be utilized for dimensionality reduction of the input data. For this purpose, deep "autoencoders" [24, 25] have been shown to provide successful results in a wide variety of applications such as document retrieval [26] and image retrieval [27]. An autoencoder (see Figure 4) is an unsupervised neural network in which the target values are set to be equal to the inputs. Autoencoders are mainly composed of an "encoder" network, which transforms the input data into a low-dimensional code, and a "decoder" network, which reconstructs the data from the code. Training these deep models involves minimizing the error between the original data and its reconstruction. In this process, the weights initialization is critical to avoid reaching a bad local optimum; thus some authors have proposed a pretrained stage based on stacked RBMs and a fine-tuning stage using backpropagation [24, 27]. In addition, the encoder part of the autoencoder can serve as a good unsupervised nonlinear feature extractor. In this field, the use of Stacked Denoising Autoencoders (SDAE) [25] has been proven to be an effective unsupervised feature extractor in different classification problems. The experiments presented in [25] showed that training denoising autoencoders with higher noise levels forced the model to extract more distinctive and less local features.

2.3. Deep Reinforcement Learning. In reinforcement learning, an agent is defined to interact with an environment, seeking to find the best action for each state at any step in time (see

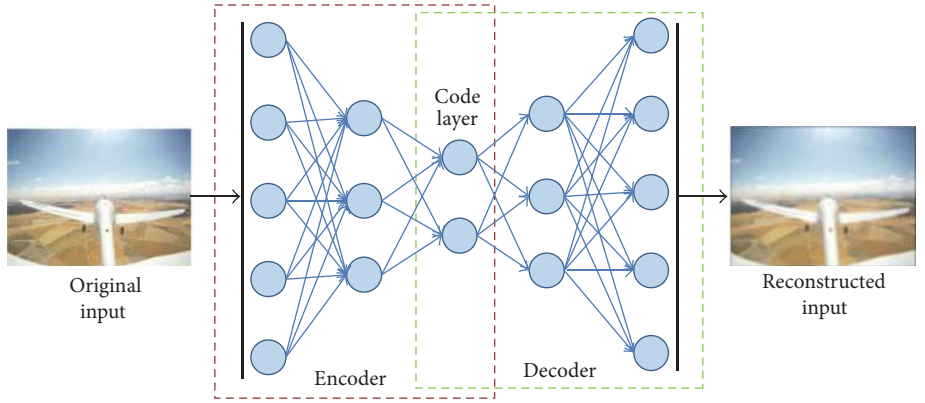

Figure 4: Deep autoencoder. An autoencoder consists of an encoder network, which transforms the original input data into a low-dimensional code, and a decoder network, which reconstructs the data from the code.

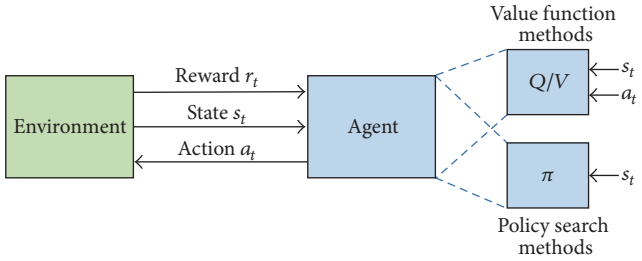

Figure 5: Generic structure of a reinforcement learning problem. The optimization methods to solve the reinforcement learning problem are mainly categorized into value function and policy search methods.

Figure 5). The agent must balance exploration and exploitation of the state space in order to find the optimal policy that maximizes the accumulated reward from the interaction with the environment. In this context, an agent modifies its behaviour or policy with the awareness of the states, actions taken, and rewards for every time step. Reinforcement learning composes an optimization process throughout the whole state space in order to maximize the accumulated reward. Robotic problems are often task-based with temporal structure. These types of problems are suitable to be solved by means of a reinforcement learning framework [28].

The standard reinforcement learning theory states that an agent is able to obtain a policy, which maps every state $s \in \mathbb{S}$ to an action $a \in \mathbb{A}$, where $\mathbb{S}$ is the state space (possible states of the agent in the environment) and $\mathbb{A}$ is the finite action space. The inner dynamics of the agent are represented by the transition probability model $p(s_{t+1} \mid s_t, a_t)$ at time t. The policy can be stochastic $\pi(a \mid s)$, with a probability associated with each possible action, or deterministic $\pi(s)$. In each time step, the policy determines the action to be chosen and the reward $r(s_t, a_t)$ is observed from the environment. The goal of the agent is to maximize the accumulated discounted reward $R_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i)$ from a state at time t to time T ($T = \infty$ for infinite horizon problems) [29]. The discount factor γ is defined to allocate different weights for the future rewards.

For a specific policy π, the value function V^π in (17) is a representation of the expectation of the accumulated discounted reward R_t for each state $s \in \mathbb{S}$ (assuming a deterministic policy $\pi(s_t)$):

$$V^\pi(s_t) = \mathbb{E}\left[R_t \mid s_t,\ a_t = \pi(s_t)\right]. \tag{17}$$

An equivalent of the value function is represented by the action-value function Q^π in (18) for every action-state pair (s_t, a_t):

$$Q^\pi(s_t, a_t) = r(s_t, a_t) + \gamma \sum_{s_{t+1}} p(s_{t+1} \mid s_t, a_t) V^\pi(s_{t+1}). \tag{18}$$

The optimal policy π^* shall be the one that maximizes the value function (or equivalently the action-value function), as in the following equation:

$$\pi^* = \arg\max_\pi V^\pi(s_t). \tag{19}$$

A general problem in real robotic applications is that the state and action spaces are often continuous spaces. A continuous state and/or action space can make the optimization problem intractable, due to the overwhelming set of different states and/or actions. As a general framework for representation, reinforcement learning methods are enhanced through deep learning to aid the design for feature representation, which is known as deep reinforcement learning. Reinforcement learning and optimal control aim at finding the optimal policy π^* by means of several methods. The optimal solution can be searched in this original primal problem, or the dual formulation V^*, Q^* can be the optimization objective. In this review, deep reinforcement learning methods are divided into two main categories: value function and policy search methods.

2.3.1. Value Function Methods. These methods seek to find optimal V^*, Q^*, from which the optimal policy π^* in (20) is directly derived. Q-learning approaches are based on the

optimization of the action-value function Q, based on the *Bellman Optimality Equation* [29] for Q (see (21)):

$$\pi^* = \arg\max_{a_t} Q^*(s_t, a_t), \tag{20}$$

$$Q^*(s_t, a_t) = \mathbb{E}\left[r(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})\right]. \tag{21}$$

Deep Q-Network (DQN) [30, 31] method estimates the action-value function (see (22)) by means of a CNN model with a set of weights θ as $Q^*(s, a) \approx Q(s, a; \theta)$:

$$\begin{aligned} Q_i^*(s_t, a_t) &= y_i \\ &= \mathbb{E}\left[r(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_{i-1}) \mid s_t, a_t\right]. \end{aligned} \tag{22}$$

The CNN can be trained by minimizing a sequence of loss functions $L_i(\theta_i)$ which are optimized in each iteration i as shown in the following equation:

$$L_i(\theta_i) = \mathbb{E}\left[(y_i - Q(s_t, a_t; \theta_i))^2\right]. \tag{23}$$

The state s of the DQN algorithm is the raw image and it has been widely tested with Atari games [31]. DQN is not designed for continuous tasks; thus this method may find difficulties approaching some robotics problems previously solved by continuous control. Continuous Q-learning with Normalized Advantage Functions (NAF) overcomes this issue by the use of a neural network that separately outputs a value function $V(x)$ and an advantage term $A(x, u)$, which is parametrized as a quadratic function of nonlinear features [32]. These two functions compose final $Q(x, u \mid \theta^Q)$, given by the following equation:

$$Q(x, u \mid \theta^Q) = A(x, u \mid \theta^A) + V(x \mid \theta^V) \tag{24}$$

with x being the state, u being the action, and θ^Q, θ^A, and θ^V being the sets of weights of Q, A, and V functions, respectively. This representation allows simplifying more standard actor-critic style algorithms, while preserving the benefits of nonlinear value function approximation [32]. NAF is valid for continuous control tasks and takes advantage of trained models to approximate the standard model-free value function.

2.3.2. Policy Search Methods. Policy-based reinforcement learning methods aim towards directly searching for the optimal policy π^*, which provides a feasible framework for continuous control. Deep Deterministic Policy Gradient (DDPG) [33] is based on the actor-critic paradigm [29], with two neural networks to approximate a greedy deterministic policy (actor) and Q function (critic). The actor network is updated by applying the chain rule to the expected return from the start distribution J with respect to the actor parameters (see (25)):

$$\nabla_{\theta^\mu} J \approx \mathbb{E}_{s_t \sim \rho^\beta}\left[\nabla_{\theta^\mu} Q(s, a \mid \theta^Q)\big|_{s=s_t, a=\mu(s_t \mid \theta^\mu)}\right]. \tag{25}$$

DDPG method learns with an average factor of 20 times fewer experience steps than DQN [33]. Both DDPG and DQN require large samples datasets, since they are model-free algorithms. Regarding DNN-based Guided Policy Search (DNN-based GPS) [34] method, it learns to map from the tuple raw visual information and joint states directly to joint torques. Compared to the previous works, it managed to perform high-dimensional control, even from imperfect sensor data. DNN-based GPS has been widely applied to robotic control, from manipulation to navigation tasks [35, 36].

3. Deep Learning for Feature Extraction

The main objective of feature extraction systems is to extract representative features from the raw measurements provided by sensors on board a UAV.

3.1. With Image Sensors. Deep learning techniques for feature extraction using image sensors have been applied over a wide range of applications using different imaging technologies (e.g., monocular RGB camera, RGB-D sensors, infrared, etc.). Despite the wide variety of sensors utilized for image processing, main deep learning feature extractors are based on CNNs [67]. As explained in Section 2.1, CNN models consist of several stacked convolution and pooling layers. The convolution layers are responsible for extracting features from the data by convolving the input image with learned filters, while pooling layers provide a dimensionality reduction over previous convolution layers.

In the robotics field, feature extraction systems based on CNN models have been mainly applied for object recognition [42–48] and scene classification [51–54]. Concerning the object recognition task, recent advances have integrated object detection solutions by means of bounding box regression and object classification capabilities within the same CNN model [42–44]. Unsupervised feature learning for object recognition was applied in [68], making fewer requirements on manually labeled training data, the obtainment of which can be an extremely time-consuming and costly process. Regarding the scene classification problem, recent advances have focused on learning efficient and global image representations from the convolutional and fully connected layers from pretrained CNNs in order to obtain representative image features [53]. In [52], it was also shown that the learned features obtained from pretrained CNN models were able to generalize properly even in substantially different domains for those in which they were trained, such as the classification of aerial images. Scene classification on board a Parrot AR.Drone quadrotor was also presented in [40], where a 10-layered CNN was utilized for classifying the input image of a forest trail into three classes, each of which represented the action to be taken in order to maintain the aerial robot on the trail (turn left, go straight, and turn right).

Nowadays, object recognition and scene classification from aerial imagery using deep learning techniques have also acquired a relevant role in agriculture applications. In these kinds of applications, UAVs provide a low-cost platform for aerial image acquisition, while deep learned features

are mainly utilized for plant counting and identification. Several applications have used deep learning techniques for this purpose [12, 49, 50, 55, 56], providing robust systems for monitoring the state of the crops in order to maximize their productivity. In [55], a sparse autoencoder was utilized for unsupervised feature learning in order to perform weed classification from images taken by a multirotor UAV. In [56], a hybrid neural network for crop classification amongst 23 classes was proposed. The hybrid network consisted of the combination of a Feedforward Neural Network for histogram information management and a CNN. In [49], the well-known AlexNet CNN architecture proposed in [69] was utilized in combination with a sliding window object proposal technique for palm tree detection and counting. Other similar approaches have focused on weed scouting using a CNN model for weed specifies classification [12].

Deep learning techniques applied on images taken from UAVs have also gained a lot of importance in monitoring and search and rescue applications, such as jellyfish monitoring [70], road traffic monitoring from UAVs [71], assisting avalanche search and rescue operations with UAV imagery [72], and terrorist identification [73]. In [72, 73], the use of pretrained CNN models for feature extraction is worth noting again. In both cases, the well-known Inception model [74] was used. In [72], the Inception model was utilized with a Support Vector Machine (SVM) classifier for detecting possible survivors, while in [73], a transfer-learning technique was used to fine-tune the Inception network in order to detect possible terrorists.

Most of the presented approaches, especially in the field of object recognition, require the use of GPUs for dealing with real-time constraints. In this sense, the state-of-the-art object recognition systems are based on the approaches presented in [46, 47], in which the object recognizer is able to run at rates from 40 to 90 frames per second on an Nvidia GeForce GTX Titan X.

Despite the good results provided by the aforementioned systems, UAV constraints such as endurance, weight, and payload require the development of specific hardware and software solutions for being embedded on board a UAV. Taking these limitations into account, only few systems in the literature have embedded feature extraction algorithms using deep learning processed by GPU technology on board a UAV. In [75], the problem of automatic detection, localization, and classification (ADLC) of plywood targets was addressed. The solution consisted of a cascade of classifiers based on CNN models trained on an Nvidia Titan X and applied over 24 M-pixel RGB images processed by an Nvidia Jetson TK1 mounted on board a fixed-wing UAV. The ADLC algorithm was processed by combining the CPU cores for the detection stage, allowing the GPU to focus on the classification tasks.

3.2. With Other Sensors. Most of the presented workload using deep learning in the literature has been applied to data capture by image sensors due to the consolidated results obtained using CNN models. However, deep learning techniques cover a wide range of applications and can be used in conjunction with sensors other than cameras, such as acoustic, radar, and laser sensors.

Deep learning techniques for UAVs have been utilized for acoustic data recognition [64, 65]. In [64], a Partially Shared Deep Neural Network (PS-DNN) was proposed to deal with the problem of sound source separation and identification using partially annotated data. For this purpose, the PS-DNN is composed of two partially overlapped subnetworks: one regression network for sound source separation and one classification network responsible for the sound identification. The objective of the regression network for sound source separation is to improve the network training for sound source classification by providing a cleaner sound signal. Results showed that PS-DNN model worked reasonably well for people's voice identification in disastrous situations. The data was collected using a microphone array on board a Parrot Bebop UAV.

In [65], the problem of UAVs identification based on their specific sound was addressed by using a bidirectional LSTM-RNN with 3 layers and 300 LSTM blocks. This model exhibited the best performance amongst other 2 preselected models, namely, Gaussian Mixture Models (GMM) and CNN.

Concerning the radar technology and despite the fact that radar data has not been widely addressed using deep learning techniques for UAVs in the literature, the recent advances presented in [62] are worth mentioning. In this paper, the spectral correlation function (SCF) was captured using a 2.4 GHz Doppler radar sensor that was utilized in order to detect and classify micro-UAVs amongst 3 predefined classes. The model utilized for this purpose was based on a semisupervised DBN trained with the SCF data.

Regarding laser technology, in [66], a novel strategy for detecting safe landing areas based on the point clouds captured from a LIDAR sensor mounted on a helicopter was proposed. In this paper, subvolumes of $1\,m^3$ from a volumetric density map constructed from the original point cloud were used as input to a 3D CNN which was trained to predict the probability of the evaluated area as being a safe landing zone. Several CNN models consisting of one or two convolutional layers were evaluated over synthetic and semisynthetic datasets, showing in both cases good results when using a 3D CNN model with two convolutional layers.

4. Deep Learning for Planning and Situational Awareness

Several deep learning developments have been reported for tasks related to UAV planning and situational awareness. Planning tasks refer to the generation of solutions for complex problems without having to hand-code the environment model or the robot's skills or strategies into a reactive controller. Planning is required in the presence of unstructured, dynamic environments or when there is diversity in the scope and/or the robot's tasks. Typical tasks include path, motion, navigation, or manipulation planning. Situational awareness tasks allow robots to have knowledge about their own state and their environment's state. Some examples of this kind of tasks are robot state estimation, self-localization, and mapping.

4.1. Planning. Path planning for collaborative search and rescue missions with deep learning-based exploration is presented in [57]. This work, where a UAV explores and maps the environment trying to find a traversable path for a ground robot, focuses on minimizing overall deployment time (i.e., both exploration and path traversal). In order to map the terrain and find a traversable path, a CNN is proposed for terrain classification. Instead of using a pretrained CNN, training is done on the spot, allowing training the classifier on demand with the terrain present at the disaster site [58]. However, the model takes around 15 minutes to train.

4.2. Situational Awareness. Cross-view localization of images is achieved with the help of deep learning in [59]. Although the work is presented as a solution for UAV localization, no UAVs were used for image collection and the experiments were based on ground-level images only. The approach is based on mining a library of raw image data to find nearest neighbor visual features (i.e., landmarks) which are then matched with the features extracted from an input query image. A pretrained CNN is used to extract features for matching verification purposes, and although the approach is said to have low computational complexity, authors do not provide details about retrieval time.

Ground-level query images are matched to a reference database of aerial images in [60]. Deep learning is applied here to reduce the wide baseline and appearance variations between both ground-level and aerial images. A pair-based network structure is proposed to learn deep representations from data for distinguishing matched and unmatched cross-view image pairs. Even though the training procedure in the reported experiments took 4 days, the use of fast algorithms such as locality-sensitive hashing allowed for real-time cross-view matching at city scale. The main limitation of their approach is the need to estimate scale, orientation, and dominant depth at test time for ground-level queries.

In [61], a CNN is proposed to generate control actions (the permitted turns for a UAV) given an image captured on board and a global motion plan. This global motion plan indicates the actions to take given a position on the map by means of a potential function. The purpose of the CNN is to learn the mapping from images to position-dependent actions. The process would be equivalent to perform image registration and then generate the control actions given the global motion plan but this behaviour is here learnt to be efficiently encoded in a CNN, demonstrating superior results to classical image registration techniques. However, no tests on real UAV were carried out and no information is provided about execution time, which might complicate the deployment for a real UAV application.

As seen from the presented works, developments in planning and situational awareness with deep learning for UAVs are still quite rudimentary. The path planning approach presented is limited to small-scale disaster sites and the different localization and mapping approaches are still slow and have little accuracy for real UAV applications.

5. Deep Learning for Motion Control

Deep learning techniques for motion control have been recently involved in several scientific researches. Classic control has solved diverse robotic control problems in a precise and analytic manner, allowing robots to perform complex maneuvers. Nevertheless, standard control theory only solves the problem for a specific case and for an approximated robot model, not being able to easily adapt to changes in the robot model and/or to hostile environments (e.g., a propeller on a UAV gets damaged, wind gusts, and rain). In this context, learning from experience is a matter of importance which can overcome numerous stated limitations.

As a key advantage, deep learning methods are able to properly generalize with certain sets of labelled input data. Deep learning allows inferring a pattern from raw inputs, such as images and LIDAR sensor data which can lead to proper behaviour even in unknown situations. Concerning the UAV indoor navigation task, recent advances have led to a successful application of CNNs in order to map images to high-level behaviour directives (e.g., turn left, turn right, rotate left, and rotate right) [38, 39]. In [38], Q function is estimated through a CNN, which is trained in simulation and successfully tested in real experiments. In [39], actions are directly mapped from raw images. In all stated methods, the learned model is run off board, usually taking advantage of a GPU in an external laptop.

With regard to UAV navigation in unstructured environments, some studies have focused on cluttered natural scenarios, such as dense forests or trails [40]. In [40], a DNN model was trained to map image to action probabilities (turn left, go straight, or turn right) with a final *softmax* layer and tested on board by means of an ODROID-U3 processor. The performance of two automated methods, SVM and the method proposed in [76], is latterly compared to that of two human observers.

In [37], navigable areas are predicted from a disparity image in the form of up to three bounding boxes. The center of the biggest bounding box found is selected as the next waypoint. Using this strategy, UAV flights are successfully performed. The main drawback is the requirement to send the disparity images to a host device where all computations are made. The whole pipeline for the UAV horizontal translation, disparity map generation, and waypoint selection takes about 1.3 seconds which makes navigation still quite slow for real applications. On the other hand, low-level motion control is challenging, since tackling with continuous and multi-variable action spaces can become an intractable problem. Nevertheless, recent works have proposed novel methods to learn low-level control policies from imperfect sensor data in simulation [41, 63]. In [63], a Model Predictive Controller (MPC) was used to generate data at training time in order to train a DNN policy, which was allowed to access only raw observations from the UAV onboard sensors. In testing time, the UAV was able to follow an obstacle-free trajectory even in unknown situations. In [41], the well-known Inception v3 model (pretrained CNN) was adapted in order to enable the final layer to provide six action nodes (three transitions and

Table 1: Deep learning-based UAV applications grouped by learning algorithms and application fields.

Learning type	Algorithm	Tasks	Field of application	References
Supervised	CNN	Outdoor navigation	Navigation	[37–39]
		Indoor navigation		[40, 41]
		Object recognition	Generic	[42–45]
				[46–48]
		Object recognition	Agriculture	[49, 50]
		Scene classification	Generic	[51–54]
		Scene classification	Agriculture	[55, 56]
		Path planning	Search & rescue	[57, 58]
		Image registration	Localization Navigation	[59–61]
Unsupervised	Autoencoder	Feature extraction	Agriculture	[55]
	DBN	Feature extraction	UAV identification	[62]
Reinforcement	DQN	—	—	—
	DDPG	—	—	—
	NAF	—	—	—
	GPS	Indoor navigation	Navigation	[63]

three orientations). After retraining, the UAV managed to cross a room filled with a few obstacles in random locations.

Deep learning techniques for robotic motion control can provide increasing benefits in order to infer complex behaviours from raw observation data. Deep learning approaches have the potential of generalization, with the limitations of current methods which have to overcome the difficulties of continuous state and action spaces, as well as issues related to the samples efficiency. Furthermore, novel deep learning models require the usage of GPUs in order to work in real time. In this context, onboard GPUs, Field Programmable Gate Arrays (FPGAs), or Application-Specific Integrated Circuits (ASICs) are a matter of importance which hardware manufacturers shall take into consideration.

6. Discussion

Deep learning has arisen as a promising set of technologies to the current demands for highly autonomous UAV operations, due to its excellent capabilities for learning high-level representations from raw sensor data. Multiple success cases have been reported (Tables 1 and 2) in a wide variety of applications.

A straightforward conclusion from the surveyed articles is that images acquired from UAVs are currently the prevailing type of information being exploited by deep learning, mainly due to the low cost, low weight, and low power consumption of image sensors. This noticeable fact explains the dominance of CNNs among the deep learning algorithms used in UAV applications, given the excellent capabilities of CNNs in extracting useful information from images.

However, deep learning techniques, UAV technology, and the combined use of both still present several challenges, which are preventing faster and further advances in this field.

Challenges in Deep Learning. Deep learning techniques are still facing several challenges, beginning with their own theoretical understanding. An example of this is the lack of knowledge about the geometry of the objective function in deep neural networks or why certain architectures work better than others. Furthermore, a lot of effort is currently being put in finding efficient ways to do unsupervised learning, since collecting large amounts of unlabeled data is nowadays becoming economically and technologically less expensive. Success in this objective will allow algorithms to learn how the world works by simply observing it, as we humans do.

Additionally, as mentioned in Section 2.3, real-world problems that usually involve high-dimensional continuous state spaces (large number of states and/or actions) can turn the problem intractable with current approaches, severely limiting the development of real applications. An efficient way for coping with these types of problems remains as an unsolved challenge.

Challenges in UAV Autonomy. UAV autonomous operations, enabling safe navigation with little or no human supervision, are currently key for the development of several civilian and military applications. However, UAV platforms still have important flight endurance limitations, restricting size, weight, and power consumption of the payload. These limitations arise mainly from the current state of sensor and battery technology and limit the required capabilities for autonomous operations. Undoubtedly, we will see developments in these areas in the forthcoming years.

Furthermore, onboard processing is desired for many UAV operations, especially those where communications can compromise performance, such as when large amounts of data have to be transmitted and/or when there is limited bandwidth available. Today, the design of powerful miniaturized computing devices with low-power consumption,

Table 2: Deep learning-based UAV applications grouped by the type of system within an unmanned aerial systems architecture, the sensor technologies, and the type of learning algorithms: supervised (*S*), unsupervised (*U*), and reinforcement (*R*).

Aerial robot systems	Sensing technologies	Learning algorithms	References
Feature extraction	Image	*S*	[42–45] [46–48] [51–54]
	Image	*S*, *U*	[55]
	Acoustic	*S*	[64, 65]
	Radar	*S*, *U*	[62]
	LIDAR	*S*	[66]
Planning	Image	*S*	[57, 58]
Situational awareness	Image	*S*	[59–61]
Motion control	Image	*S*	[38–41]
	LIDAR	*R*	[63]

particularly GPUs, is an active working field for embedded hardware developers.

Challenges in Deep Learning-Based UAV Applications. This review reveals that, within the architecture of an unmanned aerial system, feature extraction systems are the type of systems in which deep learning algorithms have been more widely applied. This is reasonable given the excellent abilities of deep learning to learn data representations from raw sensor data. Systems regarding higher-level abstractions, such as UAV supervision and planning systems, have so far obtained little regard from the research community. These systems implement complex behaviours that have to be learned and where the application of supervised learning (e.g., the generation of labelled datasets) is complex.

Nevertheless, systems operating at lower levels of abstraction, such as feature extraction systems, still demand great computational resources. These resources are still hard to integrate on board UAVs, requiring powerful communication capabilities and off-board processing. Furthermore, available computational resources are in most cases not compatible with online processing, limiting the applications where reactive behaviours are necessary. This again imposes the aforementioned challenge of developing embedded hardware technology advances but should also encourage researchers to design more efficient deep learning architectures.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Spanish Ministry of Science (Project DPI2014-60139-R). The LAL UPM and the MONCLOA Campus of International Excellence are also acknowledged for funding the predoctoral contract of one of the authors.

References

[1] A. G. Ivakhnenko, "Polynomial theory of complex systems," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 1, no. 4, pp. 364–378, 1971.

[2] K. Fukushima, "Neocognitron: a hierarchical neural network capable of visual pattern recognition," *Neural Networks*, vol. 1, no. 2, pp. 119–130, 1988.

[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[4] J. Deng, W. Dong, R. Socher et al., "ImageNet: a large-scale hierarchical image database," in *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, Miami, Fla, USA, June 2009.

[5] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: a review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[6] J. Schmidhuber, "Deep learning in neural networks: an overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[7] J. Gu, Z. Wang, J. Kuen et al., *Recent Advances in Convolutional Neural Networks*, https://arxiv.org/abs/1512.07108.

[8] L. Tai and M. Liu, "Deep-learning in mobile robotics - from perception to control systems: a survey on why and why not," CoRR abs/1612.07139. http://arxiv.org/abs/1612.07139.

[9] C. Martinez, C. Sampedro, A. Chauhan, and P. Campoy, "Towards autonomous detection and tracking of electric towers for aerial power line inspection," in *Proceedings of the 2014 International Conference on Unmanned Aircraft Systems, ICUAS 2014*, pp. 284–295, May 2014.

[10] M. A. Olivares-Mendez, C. Fu, P. Ludivig et al., "Towards an autonomous vision-based unmanned aerial system against wildlife poachers," *Sensors*, vol. 15, no. 12, pp. 31362–31391, 2015.

[11] A. Carrio, J. Pestana, J.-L. Sanchez-Lopez et al. et al., "Ubristes: uav-based building rehabilitation with visible and thermal infrared remote sensing," in *Proceedings of the Robot 2015: Second Iberian Robotics Conference*, pp. 245–256, Springer International Publishing, 2016.

[12] L. Li, Y. Fan, X. Huang, and L. Tian, "Real-time uav weed scout for selective weed control by adaptive robust control and

machine learning algorithm," in *Proceedings of the 2016 ASABE Annual International Meeting, American Society of Agricultural and Biological Engineers*, p. 1, 2016.

[13] J. L. Sanchez-Lopez, M. Molina, H. Bavle et al., "A multi-layered component-based approach for the development of aerial robotic systems: The aerostack framework," *Journal of Intelligent & Robotic Systems*, pp. 1–27, 2017.

[14] A. Graves, "Generating sequences with recurrent neural networks," arXiv preprint https://arxiv.org/abs/1308.0850.

[15] T. M. Mitchell, *Machine Learning*, vol. 45 (37), McGraw Hill, Burr Ridge, Ill, USA, 1997.

[16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, Cambridge, Mass, USA, 2016.

[17] S. Hochreiter and J. Schmidhuber, "LSTM can solve hard long time lag problems," in *Proceedings of the 10th Annual Conference on Neural Information Processing Systems, NIPS 1996*, pp. 473–479, December 1996.

[18] A. Gibson and J. Patterson, *Deep Learning*, O'Reilly, 2016.

[19] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[20] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle et al., "Greedy layer-wise training of deep networks," *Advances in Neural Information Processing Systems*, vol. 19, pp. 153–160, 2007.

[21] P. Smolensky, "Information processing in dynamical systems: foundations of harmony theory," Tech. Rep., DTIC Document, 1986.

[22] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[23] G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal, "The "wake-sleep" algorithm for unsupervised neural networks," *Science*, vol. 268, no. 5214, pp. 1158–1161, 1995.

[24] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *American Association for the Advancement of Science. Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[25] P. Vincent, H. Larochelle, I. Lajoie, and P. Manzagol, "Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[26] R. Salakhutdinov and G. Hinton, "Semantic hashing," *International Journal of Approximate Reasoning*, vol. 50, no. 7, pp. 969–978, 2009.

[27] A. Krizhevsky and G. E. Hinton, "Using very deep autoencoders for content-based image retrieval," in *Proceedings of the 19th European Symposium on Artificial Neural Networks (ESANN '11)*, Bruges, Belgium, April 2011.

[28] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[29] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, vol. 1, MIT Press, Cambridge, UK, 1998.

[30] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Playing atari with deep reinforcement learning," arXiv preprint https://arxiv.org/abs/1312.5602.

[31] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[32] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, vol. 48, pp. 2829–2838, New York, NY, USA, June 2016, preprint https://arxiv.org/abs/1603.00748.

[33] T. P. Lillicrap, J. J. Hunt, A. Pritzel et al., "Continuous control with deep reinforcement learning," preprint https://arxiv.org/abs/1509.02971.

[34] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016, preprint https://arxiv.org/abs/1504.00702.

[35] M. Zhang, Z. McCarthy, C. Finn, S. Levine, and P. Abbeel, "Learning deep neural network policies with continuous memory states," in *Proceedings of the 2016 IEEE International Conference on Robotics and Automation, ICRA 2016*, pp. 520–527, May 2016.

[36] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning deep control policies for autonomous aerial vehicles with MPC-guided policy search," in *Proceedings of the 2016 IEEE International Conference on Robotics and Automation, ICRA 2016*, pp. 528–535, May 2016.

[37] U. Shah, R. Khawad, and K. M. Krishna, "Deepfly: Towards complete autonomous navigation of mavs with monocular camera," in *Proceedings of the Tenth Indian Conference on Computer Vision, Graphics and Image Processing, ICVGIP 16*, pp. 59:1–59:8, New York, NY, USA, 2016.

[38] F. Sadeghi and S. Levine, "Real single-image flight without a single real image," preprint https://arxiv.org/pdf/1611.04201.pdf.

[39] D. K. Kim and T. Chen, "Deep neural network for real-time autonomous indoor navigation," preprint https://arxiv.org/abs/1511.04668.

[40] A. Giusti, J. Guzzi, D. C. Ciresan et al., "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 661–667, 2016.

[41] K. Kelchtermans and T. Tuytelaars, "How hard is it to cross the room? – training (recurrent) neural networks to steer a uav," preprint https://arxiv.org/abs/1702.07600.

[42] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the 27th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '14)*, pp. 580–587, Columbus, Ohio, USA, June 2014.

[43] R. Girshick, "Fast R-CNN," in *Proceedings of the 15th IEEE International Conference on Computer Vision (ICCV '15)*, pp. 1440–1448, December 2015.

[44] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, vol. 28, pp. 91–99, 2015.

[45] J. Lee, J. Wang, D. Crandall, S. Šabanovic, and G. Fox, "Real-time, cloud-based object detection for unmanned aerial vehicles," in *Proceedings of the 1st IEEE International Conference on Robotic Computing (IRC)*, pp. 36–43, Taichung, Taiwan, April 2017.

[46] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788, 2016, preprint https://arxiv.org/abs/1506.02640

[47] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," preprint https://arxiv.org/abs/1612.08242.

[48] W. Liu, D. Anguelov, D. Erhan et al., "Ssd: Single shot multibox detector," in *Proceedings of the European Conference on Computer Vision*, pp. 21–37, Springer, 2016.

[49] W. Li, H. Fu, L. Yu, and A. Cracknell, "Deep learning based oil palm tree detection and counting for high-resolution remote sensing images," *Remote Sensing*, vol. 9, no. 1, p. 22, 2017.

[50] S. W. Chen, S. S. Shivakumar, S. Dcunha et al., "Counting apples and oranges with deep learning: a data-driven approach," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 781–788, 2017.

[51] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Proceedings of the 28th Annual Conference on Neural Information Processing Systems 2014, NIPS 2014*, pp. 487–495, December 2014.

[52] O. A. B. Penatti, K. Nogueira, and J. A. Dos Santos, "Do deep features generalize from everyday objects to remote sensing and aerial scenes domains?" in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPRW 2015*, pp. 44–51, June 2015.

[53] F. Hu, G.-S. Xia, J. Hu, and L. Zhang, "Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery," *Remote Sensing*, vol. 7, no. 11, pp. 14680–14707, 2015.

[54] A. Gangopadhyay, S. M. Tripathi, I. Jindal, and S. Raman, "Sacnn: dynamic scene classification using convolutional neural networks," preprint https://arxiv.org/abs/1502.05243.

[55] C. Hung, Z. Xu, and S. Sukkarieh, "Feature learning based approach for weed classification using high resolution aerial images from a digital camera mounted on a UAV," *Remote Sensing*, vol. 6, no. 12, pp. 12037–12054, 2014.

[56] J. Rebetez, H. F. Satizábal, M. Mota et al., "Augmenting a convolutional neural network with local histograms-a case study in crop classification from high-resolution uav imagery," in *Proceedings of the European Symposium on Artificial Neural Networks*, 2016.

[57] J. Delmerico, E. Mueggler, J. Nitsch, and D. Scaramuzza, "Active autonomous aerial exploration for ground robot path planning," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 664–671, 2017.

[58] J. Delmerico, A. Giusti, E. Mueggler, L. M. Gambardella, and D. Scaramuzza, ""on-the-spot training" for terrain classification in autonomous air-ground collaborative teams," in *Proceedings of the International Symposium on Experimental Robotics (ISER)*, EPFL-CONF-221506, 2016.

[59] T. Taisho, L. Enfu, T. Kanji, and S. Naotoshi, "Mining visual experience for fast cross-view UAV localization," in *Proceedings of the 8th Annual IEEE/SICE International Symposium on System Integration, SII 2015*, pp. 375–380, December 2015.

[60] T.-Y. Lin, Y. Cui, S. Belongie, and J. Hays, "Learning deep representations for ground-to-aerial geolocalization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*, pp. 5007–5015, June 2015.

[61] F. Aznar, M. Pujol, and R. Rizo, "Visual Navigation for UAV with Map References Using ConvNets," in *Advances in Artificial Intelligence*, vol. 9868 of *Lecture Notes in Computer Science*, pp. 13–22, Springer, 2016.

[62] G. J. Mendis, T. Randeny, J. Wei, and A. Madanayake, "Deep learning based doppler radar for micro UAS detection and classification," in *Proceedings of the MILCOM 2016 - 2016 IEEE Military Communications Conference (MILCOM)*, pp. 924–929, Baltimore, Md, USA, November 2016.

[63] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning deep control policies for autonomous aerial vehicles with MPC-guided policy search," in *Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 528–535, Stockholm, Sweden, May 2016.

[64] T. Morito, O. Sugiyama, R. Kojima, and K. Nakadai, "Partially shared deep neural network in sound source separation and identification using a uav-embedded microphone array," in *Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2016*, pp. 1299–1304, October 2016.

[65] S. Jeon, J.-W. Shin, Y.-J. Lee, W.-H. Kim, Y. Kwon, and H.-Y. Yang, "Empirical study of drone sound detection in real-life environment with deep neural networks," preprint https://arxiv.org/abs/1701.05779.

[66] D. Maturana and S. Scherer, "3D convolutional neural networks for landing zone detection from LiDAR," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '15)*, pp. 3471–3478, IEEE, Washington, DC, USA, May 2015.

[67] Y. LeCun, B. E. Boser, J. S. Denker et al., "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems*, D. S. Touretzky, Ed., vol. 2, pp. 396–404, 1990.

[68] A. Ghaderi and V. Athitsos, "Selective unsupervised feature learning with convolutional neural network (S-CNN)," in *Proceedings of the 2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 2486–2490, December 2016.

[69] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS '12)*, pp. 1097–1105, Lake Tahoe, Nev, USA, December 2012.

[70] H. Kim, D. Kim, S. Jung, J. Koo, J.-U. Shin, and H. Myung, "Development of a UAV-type jellyfish monitoring system using deep learning," in *Proceedings of the 12th International Conference on Ubiquitous Robots and Ambient Intelligence, URAI 2015*, pp. 495–497, October 2015.

[71] N. V. Kim and M. A. Chervonenkis, "Situation control of unmanned aerial vehicles for road traffic monitoring," *Modern Applied Science*, vol. 9, no. 5, pp. 1–13, 2015.

[72] M. Bejiga, A. Zeggada, A. Nouffidj, and F. Melgani, "A convolutional neural network approach for assisting avalanche search and rescue operations with UAV imagery," *Remote Sensing*, vol. 9, no. 2, p. 100, 2017.

[73] A. Sawarkar, V. Chaudhari, R. Chavan, V. Zope, A. Budale, and F. Kazi, "HMD vision-based teleoperating UGV and UAV for hostile environment using deep learning," CoRR abs/1609.04147. URL http://arxiv.org/abs/1609.04147.

[74] C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '15)*, pp. 1–9, Boston, Mass, USA, June 2015.

[75] The Technion – Israel Institute of Technology, "Technion aerial systems 2016," in *Journal Paper for AUVSI Student UAS Competition*, 2016.

[76] P. Santana, L. Correia, R. Mendonça, N. Alves, and J. Barata, "Tracking natural trails with swarm-based visual saliency," *Journal of Field Robotics*, vol. 30, no. 1, pp. 64–86, 2013.