Abstract

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 ( to  connections and  to  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 related to perception, guidance, navigation, and control of unmanned rotorcraft systems. The purpose of referring to this architecture, depicted in Figure [1](https://www.hindawi.com/journals/js/2017/3296874/fig1/" \t "https://www.hindawi.com/journals/js/2017/3296874/_blank),+ 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

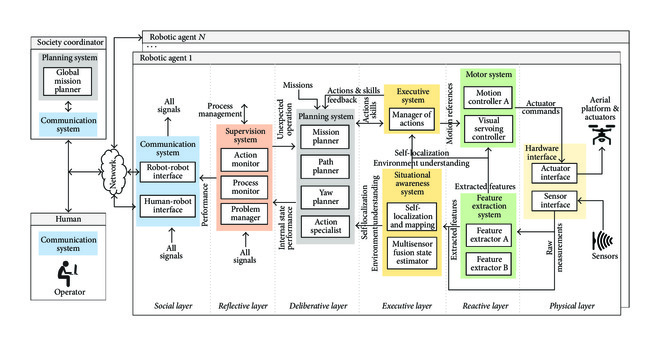


Figure 1

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.

The remainder of this paper is as follows: firstly, Section [2](https://www.hindawi.com/journals/js/2017/3296874/" \l "sec2) 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](https://www.hindawi.com/journals/js/2017/3296874/" \l "sec3) presents the state of the art in deep learning for feature extraction in UAV applications. Section [4](https://www.hindawi.com/journals/js/2017/3296874/" \l "sec4) 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](https://www.hindawi.com/journals/js/2017/3296874/" \l "sec5). Finally, a discussion of the main challenges for the application of deep learning for UAVs is covered in Section [6](https://www.hindawi.com/journals/js/2017/3296874/" \l "sec6).

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  with  features, a trained algorithm is expected to produce an output value or classification category  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  is given by the following equation:where  is a vector of  weights and  is an activation function that is usually chosen to be nonlinear. The activation of unit  in layer  given its  inputs (outputs of the previous layer ) is given by the following equation:

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](https://www.hindawi.com/journals/js/2017/3296874/fig2/" \t "https://www.hindawi.com/journals/js/2017/3296874/_blank), 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  with a 2-dimensional kernel , given by the following equation:

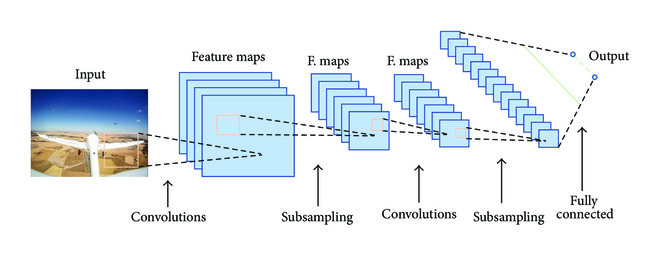
[](https://www.hindawi.com/journals/js/2017/3296874/fig2/" \t "https://www.hindawi.com/journals/js/2017/3296874/_blank)

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.

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 . 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:where  represents the hidden state at time step . The weight matrices  (input-to-hidden) and  (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  (hidden-to-output) as indicated by the following equation:

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.

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](https://www.hindawi.com/journals/js/2017/3296874/fig3/" \t "https://www.hindawi.com/journals/js/2017/3296874/_blank). The input, output, and forget gate vector activations in a standard LSTM are given as follows:

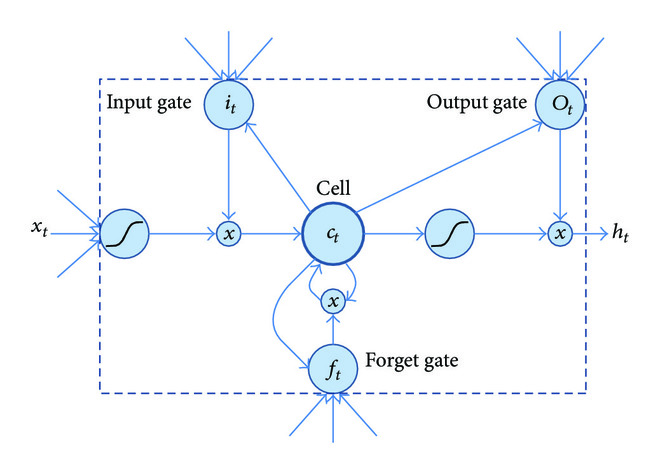
[](https://www.hindawi.com/journals/js/2017/3296874/fig3/" \t "https://www.hindawi.com/journals/js/2017/3296874/_blank)

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.

The cell state vector activation is given by the following equation:where  represents the Hadamard product. Finally, the output gate vector activation is given by the following equation:

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  can be calculated as follows:where  represent the model parameters.  are the “visible” stochastic binary units, which are connected to the “hidden” stochastic binary units . The bias terms are denoted by  for the visible units and  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](https://www.hindawi.com/journals/js/2017/3296874/" \l "EEq12)), where  represents the partition function (see ([11](https://www.hindawi.com/journals/js/2017/3296874/" \l "EEq12))):

The probability assigned by the model to a visible vector  can be computed as expressed in the following equation:

The conditional distributions over hidden variables  and visible variables  can be extracted using ([13](https://www.hindawi.com/journals/js/2017/3296874/" \l "EEq15)). 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](https://www.hindawi.com/journals/js/2017/3296874/" \l "EEq15)). 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](https://www.hindawi.com/journals/js/2017/3296874/" \l "EEq15)).where  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:where  is the learning rate,  represents the expected value of the product of visible and hidden states at thermal equilibrium, when training data is presented to the model, and  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](https://www.hindawi.com/journals/js/2017/3296874/fig4/" \t "https://www.hindawi.com/journals/js/2017/3296874/_blank)) 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.

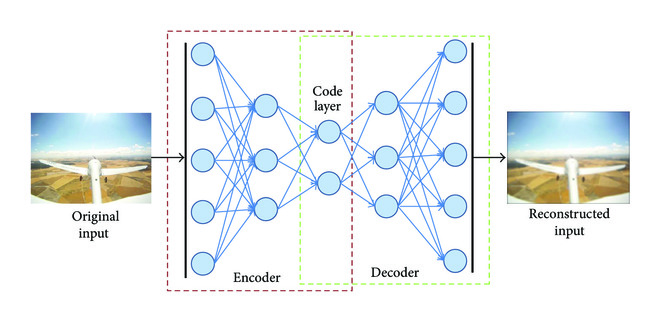
[](https://www.hindawi.com/journals/js/2017/3296874/fig4/" \t "https://www.hindawi.com/journals/js/2017/3296874/_blank)

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.

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 [5](https://www.hindawi.com/journals/js/2017/3296874/fig5/" \t "https://www.hindawi.com/journals/js/2017/3296874/_blank)). 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].

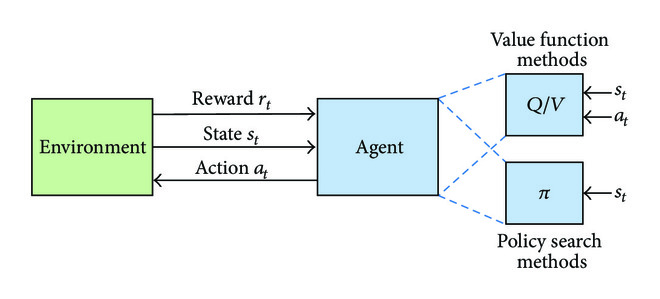
[](https://www.hindawi.com/journals/js/2017/3296874/fig5/" \t "https://www.hindawi.com/journals/js/2017/3296874/_blank)

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.

The standard reinforcement learning theory states that an agent is able to obtain a policy, which maps every state  to an action , where  is the state space (possible states of the agent in the environment) and  is the finite action space. The inner dynamics of the agent are represented by the transition probability model  at time . The policy can be stochastic , with a probability associated with each possible action, or deterministic . In each time step, the policy determines the action to be chosen and the reward  is observed from the environment. The goal of the agent is to maximize the accumulated discounted reward  from a state at time  to time  ( for infinite horizon problems) [29]. The discount factor  is defined to allocate different weights for the future rewards.

For a specific policy , the value function  in ([17](https://www.hindawi.com/journals/js/2017/3296874/" \l "EEq19)) is a representation of the expectation of the accumulated discounted reward  for each state  (assuming a deterministic policy ):

An equivalent of the value function is represented by the action-value function  in ([18](https://www.hindawi.com/journals/js/2017/3296874/" \l "EEq20)) for every action-state pair :

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

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  by means of several methods. The optimal solution can be searched in this original primal problem, or the dual formulation  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 , from which the optimal policy  in ([20](https://www.hindawi.com/journals/js/2017/3296874/" \l "EEq22)) is directly derived. -learning approaches are based on the optimization of the action-value function , based on the Bellman Optimality Equation [29] for  (see ([21](https://www.hindawi.com/journals/js/2017/3296874/" \l "EEq22))):

Deep -Network (DQN) [30, 31] method estimates the action-value function (see ([22](https://www.hindawi.com/journals/js/2017/3296874/" \l "EEq24))) by means of a CNN model with a set of weights  as :

The CNN can be trained by minimizing a sequence of loss functions  which are optimized in each iteration  as shown in the following equation:

The state  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 -learning with Normalized Advantage Functions (NAF) overcomes this issue by the use of a neural network that separately outputs a value function  and an advantage term , which is parametrized as a quadratic function of nonlinear features [32]. These two functions compose final , given by the following equation:with  being the state,  being the action, and , , and  being the sets of weights of , , and  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 , 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  function (critic). The actor network is updated by applying the chain rule to the expected return from the start distribution  with respect to the actor parameters (see ([25](https://www.hindawi.com/journals/js/2017/3296874/" \l "EEq27))):

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 -based Guided Policy Search (-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].