Review

Review of Deep Learning Methods in Robotic Grasp Detection

Shehan Caldera (shelessa@our.ecu.edu.au), Alexander Rassau (a.rassau@ecu.edu.au), and Douglas Chai (d.chai@ecu.edu.au)

School of Engineering, Edith Cowan University, Perth, Australia
* Correspondence: email to Shehan Caldera

Abstract: In order for robots to attain more general-purpose utility, grasping is a necessary skill to master. Such general-purpose robots may use their perception abilities in order to visually identify grasps for a given object. A grasp describes how a robotic end-effector can be arranged on top of an object to securely grab it between the robotic gripper and successfully lift it without slippage. Traditionally, grasp detection requires expert human knowledge to analytically form the task-specific algorithm, but this is an arduous and time-consuming approach. During the last five years, deep learning methods have enabled significant advancements in robotic vision, natural language processing, and automated driving applications. The successful results of these methods have driven robotics researchers to explore the application of deep learning methods in task generalised robotic applications. This paper reviews the current state-of-the-art in regards to the application of deep learning methods to generalised robotic grasping and discusses how each element of the deep learning approach has improved the overall performance of robotic grasp detection. A number of the most promising approaches are evaluated and the most successful for grasp detection is identified as the one-shot detection method. The availability of suitable volumes of appropriate training data is identified as a major obstacle for effective utilisation of the deep learning approaches, and the use of transfer learning techniques is identified as a potential mechanism to address this. Finally, current trends in the field and future potential research directions are discussed.

Keywords: deep learning; deep convolutional neural networks; dcnn; convolutional neural networks; cnn; robot learning; transfer learning; robotic grasping; robotic grasp detection; human-robot collaboration

1. Introduction

Recent advancements in robotics and automated systems have led to the expansion of autonomous capabilities and more intelligent machines being utilised in ever more varied applications [1,2]. The capability of adapting to changing environments is a necessary skill for task generalised robots [3,4]. Machine learning plays a key role in creating such general purpose robotic solutions. However, most robots are still developed analytically and based on expert knowledge of the application background. Even though this is considered an effective method, it is an arduous and a time-consuming approach, and has limitations for generalised applicability. Due to the recent successful results of deep learning methods in computer vision and robotics applications, many robotics researchers have started implementing deep learning methods in their research.

The type of the learning that is applied varies according to the feedback mechanism, the process used for training data generation, and the data formulation. The learning problem can vary from perception to state abstraction, through to decision making [5]. Deep Learning, a branch of machine learning, describes a set of modified machine learning techniques that, when applied to robotic systems, aims to enable robots to autonomously perform tasks that come naturally to humans.

Peer-reviewed version available at Multimodal Technologies Interact. 2018, 2, 57; doi:10.3390/mti2030057

Inspired by the biological nervous system, a network of parallel and simultaneous mathematical operations are performed directly on the available data to obtain a set of representational heuristics between the input and output data. These heuristics are then used in decision making. Deep learning models have proven effective in diverse classification and detection problems [6–8] and there is a great deal of interest in expanding their utilisation into other domains.

The grasp or grasping pose describes how a robotic end-effector can be arranged in a given image plane to successfully pick up an object. The grasping pose for any given object is determined through a grasp detection system. Any suitable perception sensors including cameras or depth sensors can be used to visually identify grasping poses in a given scene. Grasp planning relates to the path planning process that is required to securely grab the object and maintain the closed gripper contacts to hold and lift the object from its resting surface [9]. Planning usually involves the mapping of image plane coordinates to the robot world coordinates for the detected grasp candidate. The control system describes certain closed-loop control algorithms that are used to control the robotic joints or degrees of freedom (DOF) to reach the grasping pose while maintaining a smooth reach [10].

This paper reviews deep learning approaches for the detection of robotic grasping poses for a given object captured in an image. The paper is organised as follows; Section 2 provides some background information into robot grasping, Section 3 describes how robotic grasps have been represented in the literature, Section 4 introduces the main convolutional neural network approaches for detecting robotic grasps, Section 5 explores current trends for neural network architectures, Section 6 identifies popular datasets and training methodologies, and Section 7 provides some conclusions and recommendations for future research directions.

2. Background

Traditionally analytical approaches, also known as hard coding, involve manually programming the robots with the necessary instructions for performing a given task. These control algorithms are modelled based on the expert human knowledge of the robot and its environment during the specific task [4]. The outcome of this approach explains a kinematic relationship between the parameters of the robot and its world coordinates. Ju et al. [11] further suggests, the kinematic model helps in further optimising the control strategies. However direct mapping of results from a kinematic model to the robot joint controller is inherently open-loop and is identified to cause task space drifts. Therefore, Ju et al. [2] further suggests the use of closed loop control algorithms to address these drifts.

Even though such hard coded manual teaching is known to achieve efficient task performance, such an approach has limitations; in particular the program is restricted to the situations predicted by the programmer, but in cases where frequent changes of robot programming is required, due to changes in the environment or other factors, this approach becomes impractical [4]. According to Ju et al. [2], unstructured environments remain a large challenge for intelligent robots that would require a complex analytical approach to form the solution. While deriving of models requires a lot of data and knowledge of the physical parameters relating to the robotic task, use of more dynamic robotic actuators make it nearly impossible to model the physics, thus they conclude that manual teaching as an efficient but exhaustive approach [2]. In such cases, empirical methods will provide an increased cognitive and adaptive capability to the robots while reducing or completely removing the need to manually model a robotic solution [3]. Early work in empirical methods takes a classical form that explores the adaptive and cognitive capability of robots to learn tasks from demonstration. Various non-linear Regression techniques, Gaussian process, Gaussian mixture models, and support vector machines are some of the popular techniques related to this context [12]. Even though these techniques provided some cognition for the robots, the task replication is limited to the demonstrated tasks [12].

Deep learning has recently made significant developments in the application backgrounds of robotics vision, scene understanding, robotic arts and natural language processing [10,13]. Due

Peer-reviewed version available at Multimodal Technologies Interact. 2018, 2, 57; doi:10.3390/mti2030057

to the convincing results in the scope of computer vision there is an increasing trend towards implementation of deep learning methods in robotics applications. Many recent studies show that the unstructured nature of a generalised robotics task makes it significantly more challenging. However, in order to advance the state-of-the-art of robotic applications it is necessary to create a generalised robotic solution for various industries such as offshore oil rigs, remote mine sites, manufacturing assembly plants, and packaging systems where the work environments and scenarios can be highly dynamic. A desired primary ability for these general purpose robots is the capability to grasp and manipulate objects in order to interact with their work environment. Even though the visual identification and manipulation of objects is a trivial task for humans, it is a challenging task for robots that involves perception, planning, and control [10,14]. Grasping can help the robots to manipulate obstacles in the environment or to simply change the state of the environment if necessary. Early work such as [15,16] show how far researchers have advanced the research methods in robotic grasping. These studies discuss the early attempts of grasping novel objects using empirical methods.

Object grasping is challenging due to the wide range of factors such as different object shapes and unlimited object poses. Successful robotic grasping systems should be able to overcome this challenge to produce valuable results. Unlike robots, humans almost immediately know how to grasp a given object. Robotic grasping performs well below the human object grasping benchmarks but is being continually improved considering the high demand. A robotic grasping implementation has the following sub-systems [10]:

- Grasp detection system
- Grasp planning system
- Control system

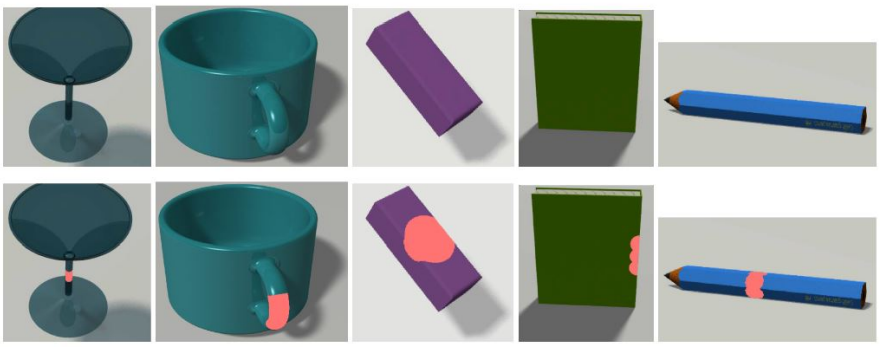
The authors identify the grasp detection sub-system as the key entry point for any robotic grasping research and aim to review current deep learning methods in grasp detection through the subsequent sections of this paper. A popular deep learning method that has been applied in most related literature is the Deep Convolutional Neural Network (DCNN) or sometimes referred to as the Convolutional Neural Network (CNN) due to the heavy involvement of convolutional layers in their architectures. It is evident that there are two approaches to apply a CNN to a problem:

1. Create an application specific CNN model
2. Use the complete or part of a pre-existing CNN model through transfer learning

Creating a proprietary CNN model requires a deeper level of understanding of the concepts and a moderate experience with CNNs. Therefore most researchers that implement CNNs in their grasp detection work have opted for transfer learning given the reduced number of parameters to be dealt with. Training of such a CNN requires a large volume of data [17]. The data can be labelled or unlabelled depending on whether a supervised or unsupervised training method is used. Training is the process of tuning the network parameters according to the training data. The studies [10,18–20] focus on simplifying the problem of grasp detection and build on the transfer learning model in order to improve the results. While there are several platforms to implement deep learning algorithms, most studies have used Tensorflow [21], Theano [22], or Matlab [6]. With the advancements of software applications and programming languages now there are more streamlined tools such as Keras [23], Caffe [24] or DarkNet [25] to implement the same functionality of former deep learning frameworks but in an efficient and easy way. Even though most recent deep learning approaches for robotic grasping follows purely supervised learning, software platforms such as NVIDIA ISAAC [26], encourage unsupervised learning methods with the support of virtual simulation capabilities.

3. Grasp Representation

Grasp detection is identified as being able to recognise the grasping point(s) or the grasping pose for an object in any given image [27]. A successful grasp describes how a robotic end-effector can be



ht (a) Martini glass (b) Mug (c) Eraser (d) Book (e) Pencil

Figure 1. Five object classes used for training with their grasping points marked on the images. Objects are a Martini glass, Mug, Eraser, Book, and a Pencil [16].

orientated on top of an object to securely hold the object between its gripper and pick the object up. As humans we use eye sight to visually identify objects in our vicinity and to find out how to approach them in order to pick them up. In a similar manner, the perception sensors on a robotic system can be used to produce the information on the environment in order to interpret them into a useful format [19]. Therefore a mapping technique is necessary to classify each pixel of the scene on the basis of belonging or not belonging to a successful grasp. Recent robotic grasping work has used different types of definitions to successful grasp poses [10,18–20].

The grasp detection has been approached in two methods. The conventional analytical method of robotic grasping is performed on the premise that certain criteria such as object geometry, physics models, and force analytics are known [9]. Whereas empirical methods or data-driven approaches rely on previously known successful results. Empirical methods are formed with the existing knowledge of grasping objects or using simulations on real robotic systems [28]. Since analytical methods perform under the assumption that the object parameters are known, for a generalised solution this will not work [28]. Empirical grasp detection can be defined in three steps:

1. Generating a dataset that consists of readings from a perception sensor along with the relevant grasp
2. Training the model on the labelled dataset considering it as the ground truth
3. Use the trained model to infer the grasp parameters for new readings from the perception sensor

In this regard, a definition of a good grasp is necessary. Different literature have used different definitions for objects in various formats. This section reviews some of the promising definitions.

Using a supervised learning approach, Saxena et al. [15] investigated a regression learning method to infer the 3-D location of the grasping point in a Cartesian coordinate system. They used a probabilistic model over possible grasping points while considering the uncertainty of the camera position [15]. To further in their investigation, they had discretised the the 3-D workspace in order to find the grasping point g, given by $g = (x, y, z)$.

To infer those three coordinates, they used two or more images of the same object [15]. As shown in Figure 1, furthering their investigation, Saxena et al. [16] defined graspable regions of objects as grasping points for their learning algorithm to infer a 3-D grasping point.

Most of these approaches were investigated with RGB colour images and RGB simulated images before the introduction of depth sensors, which has improved all detection work including the grasp detection since then. In another work that used Reinforcement Learning for robotic manipulation tasks, Zhang et al. [30] suggested that target point reaching is more important and proposed a method that used an approach similar to the one by Mnih et al. [31] to learn a Q value function to accurately predict the target points in the planar coordinates. Even though they did not further represent the grasp in a clear way, evidence suggested that their approach presented a grasp as a point.

Peer-reviewed version available at Multimodal Technologies Interact. 2018, 2, 57; doi:10.3390/mti2030057

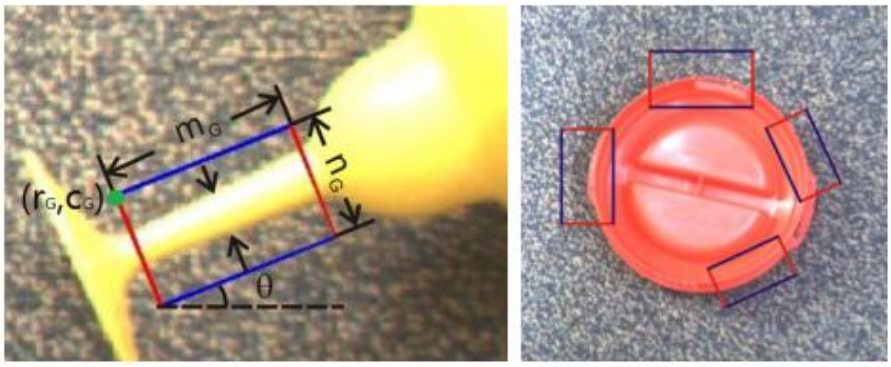


Figure 2. Grasping rectangle representation. The upper-left corner (r_G, c_G), length m_G, width n_G and its angle from the x-axis, θ_G. For some objects, like a red coloured lid in the right image, there can be multiple possible grasping rectangles [29].

Considering the challenge of grasping novel objects, one first has to define the problem space accurately to tackle the problem in a proper manner. Point defined grasps only suggest where to grasp an object. It does not determine how wide the gripper ends have to be opened nor the orientation of the gripper ends. Taking this into account Jiang et al. [29] proposed a method to detect robotic grasp configurations in 3D space using RGB-D images. According to Jiang et al. [29] a grasping configuration has a seven dimensional representation containing a **Grasping point**, **Grasping orientation**, and **Gripper opening width**. In 3D space the actual grasp representation, G can be stated as in Equation 1.

$$G = (x, y, z, \alpha, \beta, \gamma, l) \tag{1}$$

They represented a grasp as an oriented rectangle in the robot work space [29]. The rectangle edges shown in Figure 2 in blue colour represented the jaws of the gripper. The red coloured edges represented the opening or closing width of the gripper along with the direction of the motion.

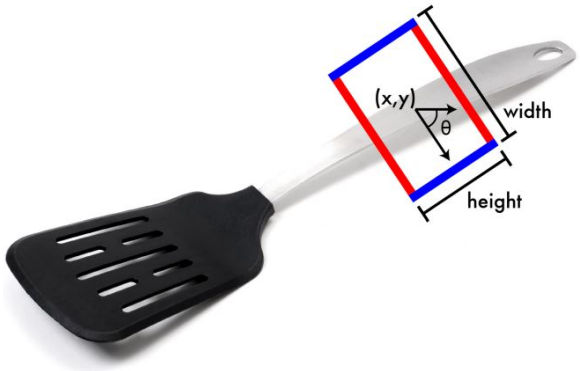


Figure 3. Grasp rectangle by Redmon et al. [19]. Where (x, y) is the center of the rectangle, θ is the orientation of the rectangle relative to the horizontal axis, h is the height, and w is the width [19].

Simplifying the seven dimensional grasp rectangle representation by Jiang et al. [29], Lenz et al. [18] proposed a five dimensional representation similar to the approach by Jiang et al. [29]. This was based on the assumption of a good 2D grasp being able to be projected back to the 3D space. While Lenz et al. [18] failed to evaluate their approach, Redmon et al. [19] confirmed the validity of the method with their own results. They supported the statements by Jiang et al. [29] and Lenz et al. [18], that detection of grasping points in this manner was analogous to object detection methods

Peer-reviewed version available at Multimodal Technologies Interact. 2018, 2, 57; doi:10.3390/mti2030057

in computer vision but with an added term for the gripper orientation [19]. Adapting the method of [18,29], they also presented a slightly updated representation of a grasp rectangle as shown in Figure 3 [19]. This modified rectangle grasp representation has been used in a number of later publications proving its usefulness [10,20,32]. In their work to use deep learning algorithms for robotic grasping detection, Kumra et al. [10] have used this grasp rectangle that was originally proposed by Redmon et al. [19]. A very recent online project page [20] has cited the same Redmon grasp rectangle in their work.

Inspired by the methods of Reinforcement Learning [31] Pinto et al. [33] presented a self-supervising algorithm for collecting data for learning to detect robotic grasps. They stated that manually labelled data would not be scalable for various applications therefore application specific data is needed for the individual application [33]. Considering the amount of data that might be required for such a method, Pinto et al. [33] presented a minimised representation of the grasp neglecting the gripper opening width. Given the grasping point $G = (x, y)$, they proposed a method to predict the grasp orientation θ [33]. According to Pinto et al. [33] a grasp, G can defined as $G = (x, y, \theta)$.

3.1. Grasp evaluation metric

Each grasp configuration that was predicted by the learning algorithms should go through an evaluation process to identify if it was in fact a valid grasp. There are two evaluation metrics for grasps: **rectangle metric** and **point metric** [10,19,20]. The point metric evaluates the distance between predicted grasp center and the actual grasp center relative to a threshold value, but the literature failed to provide further reasoning as to how to determine this threshold value. Furthermore, this metric did not consider the grasp angle which neglected the object orientation in the 2D image plane.

The rectangle metric defined a successful grasp under the following two conditions.

1. Difference between the grasp angles to be less than 30°
2. Jaccard index between the two grasps to be less than 25%

The Jaccard index is given by Equation 2:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{2}$$

These works demonstrate that there should be a consistent grasp representation method to start working with learning algorithms to detect robotic grasps. The ground truth labels should have the optimal number of parameters to represent a grasp while ensuring that it is not over-defined. Therefore the five dimensional grasp representation originally presented by Lenz et al. [18] for a 2D image should be further explored.

4. Grasp Detection using Convolutional Neural Networks

Most recent work in robotic grasp detection use different variations of convolutional neural networks to learn the optimal end-effector configuration for different object shapes and poses [10,18–20,27,33]. They do so by ranking multiple grasp configurations predicted for each object image instance. Ranking is done based on the learnt parameters from the representation learning capability of deep learning. As opposed to the manual feature design and extraction steps of classical learning approaches, deep learning can automatically learn how to identify and extract different application specific feature sets [17].

In analytical approaches, various grasping application specific parameters such as closure properties and force analysis are combined to successfully model the grasps [9]. Closure properties describes force and momentum exerted at the point of contact this is also known as a *Grasp Wrench*. Depending on the level of friction at each of these points, the point of contact could be further elaborated [9]. According to Bicchi et al. [9] the force analysis describes the required grasping force

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 May 2018 doi:10.20944/preprints201805.0484.v1

Peer-reviewed version available at Multimodal Technologies Interact. 2018, 2, 57; doi:10.3390/mti2030057

that should be applied by the robotic gripper on the object so as to grasp it securely without slipping or causing damage. Kinematic modelling between the contact points is a function that describes the relative motion between two different contact points [9]. Reviews such as [9] suggested how practically impossible, it would be to prepare a generalised grasping model using just analytical data. Given how well certain learning algorithms [10,18–20] performed in the past, however, it could be concluded that using visual presentation of successful grasps as training data with these learning algorithms would result in usable generalised solutions.

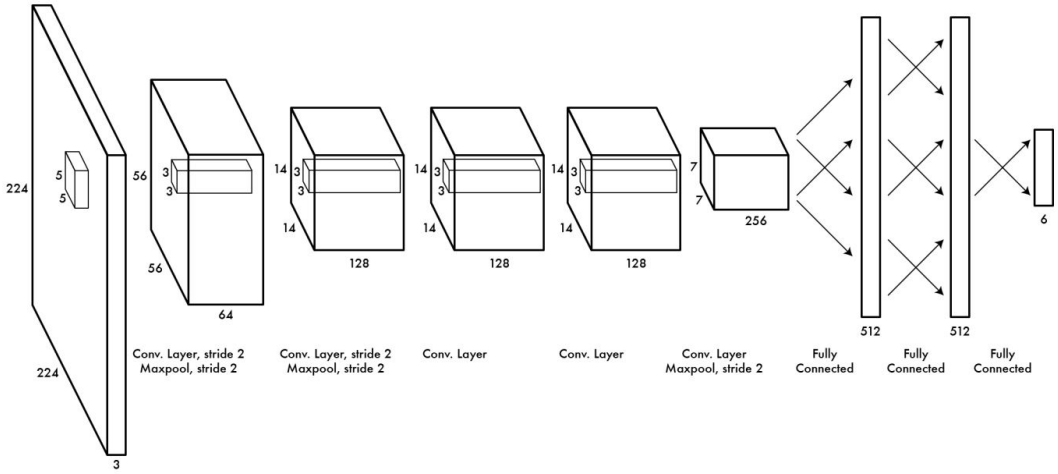


Figure 4. Neural network model proposed by Redmon et al. [19].

The most common deep learning method for detecting 2D robotic grasps is the sliding window approach. The sliding window method uses a small classifier to predict if a small patch of the image contains a potential grasp. The image is divided into a number of small patches and each patch is run through the classifier which makes it an iterative process. The patches that contain higher ranking grasps are considered as candidates and pushed as outputs. Lenz et al. [18] used a similar approach and it was later followed by Wang et al. [34] and Wei et al. [35]. In their approach, Lenz et al. [18] used RGB-D images for training while arguing that the use of multi-modal data for network training would help the robots to recognise the work environment in a more precise manner and generalise the learning to make sense of the various depths. They further supported the previous argument stating that deep learning could be used to learn the features automatically as opposed to the classical learning approaches [18]. Even though it has produced results with an accuracy of 75%, it takes about 13.5 seconds to deliver a prediction. It is theorised that the iterative scanning makes the process very slow. In addition, it requires a higher computational capacity, which always would not be practical on a robotic system. Also, such delays would not be tolerable in a practical human-robot collaborative workspace [19].

Redmon et al. [19] reasoned the iterative patch scanning requires a large block of time thus categorising sliding window as a very slow method. While arguing that one network would perform better than the two-cascaded system as in [18], they proposed a larger neural network model as shown in Figure 4 for a one-shot detection approach [19]. Using this method they produced a massive performance boost in results and delivery times [19]. Availability of cheaper depth sensors such as Microsoft Kinect made acquisition of multi-modal data possible [18]. Redmon et al. [19] presented three different architectures to detect robotic grasps from a given 2D image. The first architecture used a direct regression model with RGB images used as training data. The depth information was used for the rest of architectures. In order to avoid having to deal with multi-modal data processing, they replaced Blue channel information with depth values normalised between (0, 255) [19]. The RGD data was used with the other two network models: *Classification + Regression* and *Multi Grasp*

Peer-reviewed version available at Multimodal Technologies Interact. 2018, 2, 57; doi:10.3390/mti2030057

[19]. The *Classification + Regression* model first classified the object type and predicted the grasping rectangles accordingly [19]. The *Multi Grasp* model divided the image to 7x7 segments and predicted if each section contained its own grasping rectangle [19]. With Image-wise splitting, they achieved an accuracy of 84.4%, 85.5%, and 88% respectively for their models: Direct Regression, Regression+ Classification, and Multi Grasp respectively [19].

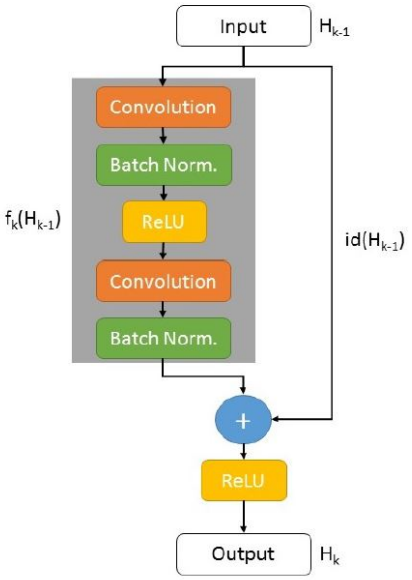


Figure 5. Example of a residual block [10].

In another approach for learning to detect robotic grasps, Kumra et al. [10] experimentally presented that increased depth of a convolutional network increased the prediction error even though the theory suggested otherwise. They criticised the frequently used optimiser function, Stochastic Gradient Decent (SGD) as it struggled to compute the identity map in an end-to-end learning approach [10]. As a solution they used the residual layers [36] in their work [10]. Residual layers reformulated the mapping function between each layer without having to compute an identity map [10]. Instead of the simplified AlexNet [37] model as in [19], they used the 50-layer ResNet model [10]. The ResNet architecture used the residual learning instead of learning an identity map, which had failed when more deep layers were used. The residual layers provided skip connections that bypassed a few layers at a time in a given feed-forward network processing stage [10]. At each skip, a residual block was called that was added to the input of each layer. The main objective was to skip convolutional and non-linear activation layers when necessary [10]. Figure 5 shows the residual block. They further analysed the uni-modal attempts such as all the models from [19] that only used three channels of data (RGB or RGD) [10]. Even though a multi-modal training data instance such as a RGB-D image that had four channels, transfer learning models restricted input to be three channels due to their original training configuration that was performed on RGB images. So attempts such as [19] converted four channel RGB-D information back into RGB, RGD, or depth only (DDD) to use with transfer learning [10]. They presented two network models [10]:

1. Uni-modal grasp predictor
2. Multi-modal grasp predictor

Peer-reviewed version available at Multimodal Technologies Interact. 2018, 2, 57; doi:10.3390/mti2030057

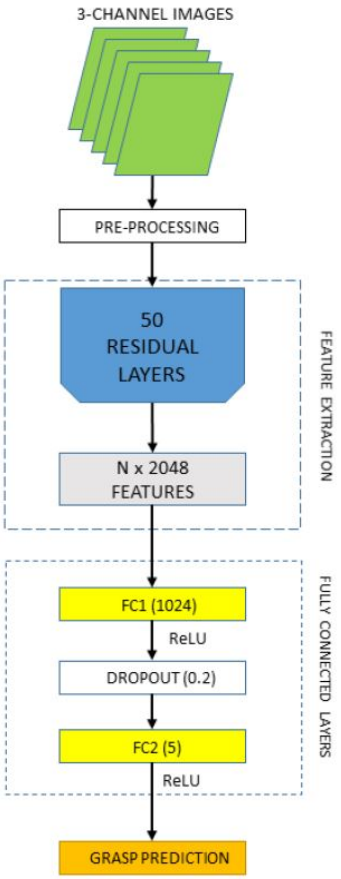


Figure 6. Uni-modal grasp predictor [10].

Most image datasets that were used for DCNN training contained only RGB images therefore the training was limited to learn only the RGB parameters [10]. The *Uni-modal* grasp predictor was trained using images with only three channels (RGB or RGD) [10]. The convolutional base of a ResNet-50 model was used to extract features from the images. The basic uni-modal predictor used a linear SVM as a classifier on these extracted features to predict the grasp [10]. This model is shown in Figure 6. The Multi-modal grasp predictor was designed in order to use the full potential of multi-modal training data. Depth values were converted to depth images in a similar manner of converting Gray images to RGB images. As illustrated in Figure 7, the RGB and Depth images were then fed into two different ResNet-50 network models to extract features [10]. These features were concatenated and fed into a shallow neural network with only three fully connected layers [10]. Using two DCNNs in parallel and merging the learnt features together it was possible to extract multi-modal features from the RGB-D dataset but a further experimentation would be necessary to evaluate this statement [10].

Peer-reviewed version available at Multimodal Technologies Interact. 2018, 2, 57; doi:10.3390/mti2030057

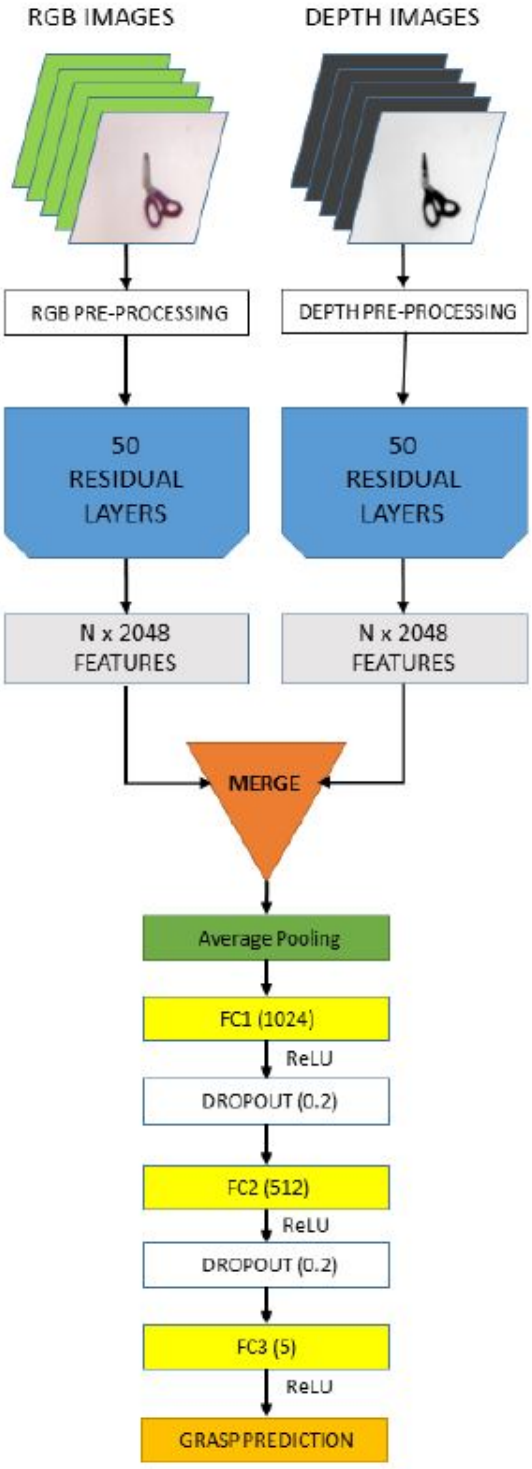


Figure 7. Multi-modal grasp predictor [10].

Pinto et al. [33] stated that regression models for predicting grasp configurations failed due to two main reasons: (a) Each object could have multiple grasping points and (b) convolutional networks were better at classification than regression. Therefore, they presented a two-step approach into predicting grasps [33]. They proposed a method to select a patch of the image and examine which angle produced a successful grasp [33]. They divided the angle space into 18 similar proportions

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 May 2018 doi:10.20944/preprints201805.0484.v1

Peer-reviewed version available at Multimodal Technologies Interact. 2018, 2, 57; doi:10.3390/mti2030057

making the problem an 18-way binary classification in their work [33]. The classification was conducted using a CNN model similar to Alexnet from [37] as shown in Figure 8 [33].

Redmon et al. [19] suggested that Network pre-training was a necessary step in any deep learning application that used transfer learning techniques. They further stated that pre-training could greatly improve training time and helped to avoid overfitting [19]. This statement was confirmed in later publication as well [10,20,32]. Kumra et al. [10] used a pre-trained ResNet-50 model for their learning application. Ebert et al. [20] also used a pre-trained Alexnet model for learning to detect grasps. In [10,19,20], authors suggested that pre-training the network models on datasets such as ImageNet [38] would rationally generalise the feature extraction making it more robust for domain specific data. Since the amount of domain specific data is limited pre-training would create more generalised filters while avoiding the overfitting during the training [10,19].

The sliding window and one-shot detection algorithms learn a certain heuristic that will be later used to infer candidate grasp(s) given an image of an object. These two methods are popular in this space due to their simplistic requirements and generalisation capability. However, some literature points to some other methods for robust grasping. Robust grasping also determines the most suitable grasp from a set of viable grasps for the given scene. It is based on a learnt function that involves different modalities of data or several types of perception sensor readings or a combination of both. As opposed to the previous method robustness learning requires lot of data, which might not always be valid for unstructured environments.

Mahler et al. [27] developed a deep learning architecture for robotic grasping known as Grasp Quality Convolutional Neural Network (GQ-CNN). It predicted an evaluation metric identified as the grasp robustness for each grasp candidate. The grasp robustness provided the valuable information required in grasping such as probability of grasp success, resistance to arbitrary forces and torques. They trained their GQ-CNN on the Dex-Net 2.0 dataset which comprised of 6.7 million point clouds, parallel-jaw grasps, and robust grasp metrics [27].

Mahler et al. [27] developed Dex-Net 2.0 for the purpose of reducing the training time with the use of cloud computing. They formed this large dataset with mesh models using physics-based models. While releasing the Dex-Net 2.0 cloud dataset Mahler individually explained in [39] that a probabilistic model is used by Dex-Net 2.0 to generate synthetic point clouds (from physics based mesh models), grasps, and grasp robustness labels from 3D object mesh models. The main insight behind the method had been that robust parallel-jaw grasps of an object were strongly correlated with the shape of the object. Further referring to their attempt in [27] Mahler [39] described that deep CNNs were able to learn the correlations among the data in Dex-Net 2.0 using a hierarchical set of filters that recognise geometric primitives.

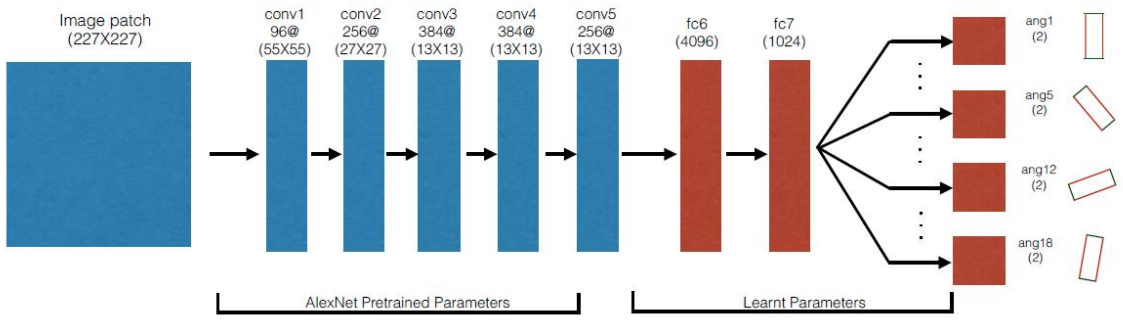


Figure 8. 18-way binary classifier by Pinto et al. [33].

In a similar approach to [27], Johns et al. [40] proposed a method for robust robotic grasp detection. Instead of detecting the single best grasp available the proposed method scored each possible grasp in the image so the grasp with the highest score would be selected. The function

Peer-reviewed version available at Multimodal Technologies Interact. 2018, 2, 57; doi:10.3390/mti2030057

that was used to score each grasp configuration was denoted the *grasp function* and it was trained by a CNN that took a depth image as input and returned scores for each grasp as outputs [40]. The proposed CNN had a similar network architecture to Alexnet [37]. The training data was generated by physics simulation and depth image simulation using 3D meshes [40]. Johns et al. [40] further stated that the higher volume of data requirement of convolutional networks remained a major challenge and proposed that simulation data could be helpful in the endeavour.

Viereck et al. [41] suggested that current methods could not dynamically adapt for environment changes during the grasp and thus proposed a CNN based closed-loop approach to react for unexpected disturbances of the object during grasping that would often cause the grasp to fail. The closed-loop approach calculated the distance to the grasp function that was trained using a CNN and successfully grasped objects that would change their poses mid grasping [41]. The neural network took a depth image, and a grasp (x, y, θ) as inputs and gave a distance value as the output. The grasp with the closest distance was always reached making closed-loop corrections along the way [41]. The training data was generated using simulations [41].

Image segmentation techniques were also being used for detecting graspable regions of a given scene. In an attempt to win the Amazon Picking Challenge 2016 (APC 2016) [42], Schwarz et al. [43] had designed a robotic system incorporating deep learning approaches to image segmentation. They used the DenseCap deep network [44] for its capability in captioning interesting areas of the scene using bounding boxes [43]. The CNN in DenseCap was pre-trained using image data from ImageNet [38]. Throughout the project, they had identified the importance of ideal performance of each individual network towards the success of a multiple deep network system. They concluded that they were able to produce a system that was performed well given the limited amount of training time allowed in the competition. And achieved the impressive end result of the second and the third places in the APC 2016 [43].

Detecting object grasping configurations from images is still accurately solved using analytical methods but the use of empirical methods is exponentially increasing due to successful results in recent publications. One commonly used method is to learn a visuomotor controller using deep learning that iteratively corrects the grasping point until the object is successfully grasped between the gripper jaws. The next best method is to learn a function that scores the possible grasps on an image and use it to select the highest scored grasp as the candidate. There are other methods that learn a certain heuristic and exhaustively search for possible grasps on the images. Training CNNs to detect grasps requires a higher volume of manually labelled data. As a solution, most researchers opt to use simulated training data as in [14,40,41]. On the other hand, data collection can be automated as in [33]. Ultimately recent approaches have suggested network pre-training would avoid the overfitting due to the limited availability of training data [10,19,20].

5. Convolutional Neural Network Architecture

A deep CNN is built with multiple layers to extract information representations [45]. In a manner inspired by the functioning of the human brain, a CNN learns a hierarchy of concepts that translates a combination of data points to an array of easily understandable values [17]. Goodfellow et al. [17] identifies the basic concept of neurons of such a system as similar to the ones in a human brain, and proceeds to explain how the same concept has been formalised in literature. In an attempt to standardise the theoretical approach of creating a deep learning network they explain that the neurons of a deep learning network are referred to as the features that have been extracted from the input data [17].

During the last five years there have been many active improvements for DCNN architectures. Most of these approaches use ImageNet [38] tests for benchmarking. Krizhevsky et al. [37] showed that using purely supervised learning, they were able to achieve remarkable results on identifying objects in the images from the highly challenging *Imagenet* image database [38]. They had created a deep convolutional neural network and improved the learning avoiding unsupervised approaches in

Peer-reviewed version available at Multimodal Technologies Interact. 2018, 2, 57; doi:10.3390/mti2030057

training. They had further studied how computational power helped them in increasing the network size and they concluded that longer training periods helped to improve the results [37]. They also stated that a higher computing capacity would be required to match the human processing speeds in image recognition. They hypothesised that by using video sequences with deeper neural networks it would be possible to obtain information that was missing or less obvious in static images [37]. With their CNN architecture AlexNet achieved a top error rate of 16.6% in the ImageNet challenge in 2012 exceeding the current state-of-the-art at the time [37].

When once again outperforming the then state-of-the-art approaches at the ImageNet Recognition Competition in 2014 [38] by achieving a top error rate of 6.8%, Szegedy et al. [46] stated that advances in the quality of image recognition had relied on newer ideas, algorithms, and improved network architectures as well as more powerful hardware, larger training data sets, and bigger learning models. They further commented that neither the deep networks nor the bigger models alone would result in such improvements but combining them in order to create a deeper architecture would suggest the same improvements over the classical theories [46]. They experimentally presented that by increasing the depth (the number of deeper levels) and the width (the number of units at each level) of a deep network would improve the overall network performance [46]. But it would require a larger set of training data that would result in the following drawbacks:

1. Larger data sets would result in increased features to be extracted while limited data sets would result in overfitting.
2. Larger data sets would require increased computational resources.

He et al. [36] stated that the computer vision research groups had theorised that deeper neural networks would succeed in image recognition problem [38]. But they further stated based on experimental results with increasing depth, that the network accuracy became saturated and started degrading [36]. There was evidence to prove that this was not due to overfitting [36]. As a solution to this degradation problem, they proposed a novel approach to fit the layers in a residual mapping instead of hoping that the stacked layers will directly fit a desired underlying mapping. Based on this, they managed to evaluate a residual neural network with up to 152 layers and to achieve an error rate of 3.57% [36].

In a further study, Szegedy et al. [47] praised the achievement by He et al. [36] in tackling the challenge of training deeper networks. Based on this, Szegedy et al. [47] proposed that by combining the residual network from [36] with their latest version of the inception network [48], the results could be improved. While highlighting the importance of residual connections in training deeper networks in [47], they experimentally showed that residual connections could replace the filter concatenation stage of the inception architecture. They further proposed that it would retain the efficiency of network training attribute of the inception architecture in a limited resources environment [47]. With this combination they managed to achieve an error rate of 3.08% in the ImageNet classification (CLS) challenge. The proposed improvement for the residual blocks is shown in Figure 9.

In an approach to interact with human hands, attempting to predict their future locations Lee and Ryoo [49] had proposed a convolutional neural network for future representation regression. They suggested that the hand locations at each time-step along with the robot positions could be considered as the inputs for the learning system. Therefore, it would allow prediction of the next hand location focusing on the time spent [49]. Lee and Ryoo [49] performed several experiments to prove their approach, achieving a successful outcome.

Polydoros et al. [50] identified the difficulties in manually identifying a dynamical model for a robot and proposed a neural network approach for learning inverse dynamics of manipulators using sensory data. They tested an initial hypothesis with several robotic platforms including a Baxter research robot [51]. Their algorithm was found to be better exploited when the model was continuously updated. They identified that the recurrent structure of their network allowed the exploitation improvement. They further identified that their method was more adaptable in object manipulation tasks: picking and releasing of objects.

Peer-reviewed version available at Multimodal Technologies Interact. 2018, 2, 57; doi:10.3390/mti2030057

Figure 9. Modified residual blocks. **Left**: residual block by He et al. [36], **Right**: improved residual block with 1x1 convolution by Szegedy et al. [47].

The inverse kinematics problem in robotics was classified by Xia and Wang [52] to be a time-varying quadratic optimisation problem. They further identified that neural networks had conveniently reduced the computational complexity of motion planning referring to various literature [35,53]. They jointly stated that most of the networks applied to kinematic control used feed-forward approaches, where they had identified the use of recurrent networks could avoid the need for off-line supervised learning and would be more suitable for unstructured environments. Therefore, they proposed the use of a single layer recurrent neural network reducing the network complexity and focusing on real-time learning. They had further proved the stability of their network [52].

6. Datasets for Deep Learning

The literature proves that deep learning requires a large volume of labelled data to learn the features during the training process [17]. The similar requirement that is apparent in supervised learning methods in robotic grasp detection supports this claim [10,18–20]. In recently published work, researchers either use training data from a third party or introduce their own application specific proprietary data sources or methods to automate the data generation [27,33]. Johns et al. [40] highlight that the major challenge with deep learning is it needs a very large volume of training data thus Johns et al. [40] opt to generate and use simulated data for the training process. Another challenge of training deep neural networks is that it lacks domain specific data as mentioned by Tobin et al. [14]. They proposed a method in [14], to generate generalised object simulations in order to address this challenge, although it is yet to be proven how effective the results can be [14]. For a real-time application, simulated data and the availability of object 3D models is not practically applicable [10,19]. However there are reports of network pre-training as a solution when there is limited domain specific data [10,19,20].

6.1. Training and testing data

Brownlee [54] identified that there were three different learning approaches in machine learning according to the availability of data and further categorised learning algorithms into three categories:

1. **Supervised Learning**: The input data was manually corresponded with the target labels and the system then tries to predict the mapping between input and output.
2. **Unsupervised Learning**: The system only has an input data set and tries to find the coherence between probable target labels thus making the learning without supervision.

Peer-reviewed version available at Multimodal Technologies Interact. 2018, 2, 57; doi:10.3390/mti2030057

3. **Semi-supervised Learning**: There is a partially arranged correspondence between input data and the target labels.

The effective accuracy of a learning algorithm relies heavily on how extensively it was trained. The outcome accuracy of any learning algorithm depends mostly on the following three factors [17, 31]:

- Data for training
- Architecture of the algorithm
- Self-exploration capacity of the algorithm

In an online article, Brownlee [55] explained the three different categories of training data. They broke down the training data into *training data*, *testing data*, and *validation data* and further explained the following terms as mentioned below [55]:

1. **Training data**: Sample of the data that was used to fit the neural network model.
2. **Validation data**: Sample of data that was used to evaluate the network model that was fit using the training data while tuning model hyper-parameters.
3. **Testing data**: Sample of previously unseen data during the training that was used to evaluate the model.

Goodfellow et al. [17] stated that the performance of a simple machine learning algorithm could rely on the input data representation. Furthering their statement they explained that for an AI system that was intended to diagnose a patient, the system would not have enough information to perform the diagnosis if the only input for the AI system is a MRI (Magnetic Resonance Imaging) scan image. There should also be enough guidelines to identify which type of data and what kind of data labels are represented by this particular input [17]. In similar terms Tobin et al. [14] stated that it is important to have a large volume of domain specific data to fit a model in any learning algorithm and proposed a method to use physics simulations to generate domain specific images of a particular set of generic object shapes to use in a generalised robotic grasping application. Preparing such a large database, however would require a tedious amount of human annotations. In this case, Mahler et al. [27] suggested to populate a dataset containing physics based analyses such as caging, grasp wrench space (GWS) analyses and grasp simulation data for different types of object shapes and poses. They further suggested that cloud computing could be leveraged to train a convolutional neural network with this dataset that would in turn, predict a robustness metric for a given grasp instead of directly predicting a grasp [27]. The proposed dataset was called *Dex-Net 2.0* and contained about 6.7 million point clouds and analytic grasp quality metrics with parallel-jaw grasps planned using robust quasi-static GWS analysis on a dataset of 1,500 3D object models [27].

6.2. Datasets for grasp detection

Most of the prior work used 3D simulations to find suitable grasp poses for objects, however, Kumra et al. [10] stated that even though those previous work had performed well they required a known 3D model of the object to calculate a grasp. This 3D model would not be known a priori and the complex modelling techniques of forming the 3D model was beyond the capacity for most of the general purpose robots as their desired primary function was the faster adaptation to dynamic scenarios [19]. In such cases, a learning algorithm would produce the necessary results provided that there were enough data instances for the training. As pointed out by Tobin et al. [14], most applications lack domain specific data. Mahler et al. [27] provided a fair reasoning for this statement, concluding that human annotation is a tedious process which requires months of work and the simulations alone would have to be run for numerous iterations on a robotic system. With the limited availability of domain specific data, Redmon et al. [19] proposed to use pre-training. Pre-training would assume by using the weights of the convolutional overhead of a CNN model that was trained on a large dataset such as Imagenet [38] would transfer the universal filtration to a smaller

Peer-reviewed version available at Multimodal Technologies Interact. 2018, 2, 57; doi:10.3390/mti2030057

dataset while providing better results compared to the usual training approach. The Cornell Grasp Dataset (CGD) at [56] was identified as such a small dataset that was compiled for most pre-training approaches in robotic grasping [10,19,20,32]. The CGD was created with grasp rectangle information for 240 different object types and it contained about 885 images, 885 point clouds and about 8019 labelled grasps including valid and invalid grasp rectangles. A sample of a set of images is shown in Figure 10. The grasps were specifically defined for the parallel plate gripper found on many robotic end-effectors. The CGD appeared in a number of research studies during the recent past, which might suggest that it has a reasonable diversity of examples for generalised grasps [19]. The recent trend of using RGB-D for learning to predict grasps was covered with the CGD dataset by including the point cloud data by its creators. Lenz et al. argued that having the depth information would benefit a better depth perception for an inference system that was trained on depth data [18]. A sense of good and bad grasps was also necessary to differentiate a better grasp from the alternatives [18,19]. Therefore, the CGD could be selected as a suitable dataset for its quality and adaptability.

Figure 10. Sample of Cornell Grasp dataset [56].

Most recent grasping work looked into grasp pose detection, feedback correction of the detected grasp, and servoing of the robotic grasp system. Most of these applications specifically relied on human annotated data [57]. While there are several of annotated datasets for grasping, the CGD stood out for its adaptability. In addition to the reasoning in [27], Pinto et al. [33] identified two other challenges with human labelling. First, most objects could be grasped in multiple ways which made exhaustive labelling impossible and second, human understanding of object grasping would always be biased on the semantics [33]. Therefore Pinto et al. [33] proposed a method to automate data collection using reinforcement learning techniques in which they ran their robotic system for a total of 700 hours combining about 50,000 different instances. With a similar purpose Levine et al. [57] proposed a method that avoided human intervention in data labelling. In their approach, Levine et al. [57] conducted two experiments involving about 6-14 robotic manipulations with 800,000 and 900,000 grasp attempts in two respective experiments.

Peer-reviewed version available at Multimodal Technologies Interact. 2018, 2, 57; doi:10.3390/mti2030057

6.3. Summary

In conclusion, the amount of training data play a key role in the outcome of the trained algorithm. Approaches such as [33,57,58] try to reduce or completely avoid the challenges of compiling such huge datasets. In cases, where an extended information set is necessary to produce a grasp prediction, a dataset such as the Dex-Net 2.0 [27] could be used. For most generalised grasp prediction networks, however, the CGD [56] would be an optimal starting point considering its adaptability for more generalised object shapes and poses with the added benefit of the inclusion of depth information.

7. Conclusions

Deep learning has been showing remarkable success in recent publications on applications of computer vision such as classification, detection and localisation. Thus far, however, it has not been adopted very extensively in robotic applications, but this is a rapidly growing area of research. Due to the requirements of higher computational power and large volumes of training data, it is still challenging to implement an end-to-end learning approach for the complete robotic grasping activity. Despite this, the detection of successful grasping poses for robotic systems using deep learning methods has been investigated in a number of recent publications. In this paper these emerging methods have been reviewed and the several key elements of deep learning based grasp detection identified. The key elements can be listed as:

1. Grasp representation
2. Approach to grasp detection
3. Architecture of the CNN used
4. Training datasets

There are several methods to represent successful grasps for a robotic system. A successful grasp is identified as the location within the work area that a robotic end-effector can be placed to securely grasp the target object between its parallel plate grippers and lift it without losing its grip. The information such as the coordinates of this location, the width the gripper needs to be opened, and the gripper orientation with reference to the horizontal axis are commonly included in this grasp representation.

Even today, the use of hand-engineered features in robotic grasp detection performs well. In recent work CNNs have been investigated for their effectiveness in learning to detect grasps. The most common deep learning approach has been identified as the sliding window approach which scans a patch of image for successful grasps in an iterative process. A major limitation of this approach, however, is that it is a very slow process considering the expectations for real-time operation. An alternate approach that has been suggested to overcome this is the one-shot detection method in which the complete image is scanned to check if it contains a successful grasp. Training a suitable neural network for this purpose usually requires a large amount of manually labelled data, such as the Cornell Grasps Dataset. But training a larger CNN requires even higher volumes of domain specific data which is not readily available at the time of writing. Therefore researchers have opted to predict optimal grasp orientations trained with data that was collected with self-supervised algorithms. While most of these grasp detection algorithms follow an open-loop approach, recent research trends show a shift into closed-loop methods. With closed-loop learning algorithms the robots can even grasp objects if the grasping pose for the object changes mid-grasping.

The most used deep convolutional neural network in one-shot grasp detection work is the AlexNet architecture. It follows a basic feed-forward neural network pattern, and it has five convolutional layers followed by two fully connected layers. The convolutional layers are interspersed with normalisation and maxpooling layers. In later work, researchers have used the ResNet architecture, which is a 50 layer deep convolutional neural network model arguing that it should provide improved results compared to the results from the AlexNet architecture. The increased depth of a ResNet DCNN has demonstrated increased accuracy in most cases. ResNet

Peer-reviewed version available at Multimodal Technologies Interact. 2018, 2, 57; doi:10.3390/mti2030057

uses residual learning in order to overcome the challenge of learning an identity map. This residual learning skips a number of layer connections while increasing training rates with increased depth. The Inception architecture has shown promise with advanced results in image recognition challenges in resource limited environments but it has never being employed in grasp detection. By combining the residual learning with the inception architecture a deep network can converge to accurate results faster while retaining the accuracy of the Resnet. This is worth exploring with grasp detection as it has proven to be successful in image recognition challenges achieving 3.08% error rate compared with the error rate of 15.3% for AlexNet at the time. However, there have been active improvements for DCNN architectures during the last five years. Most of these approaches use ImageNet [38] tests for benchmarking. Therefore the use of transfer learning techniques could be useful in creating a promising CNN model.

Cheap RGB-D sensors such as Microsoft Kinect are widely available and can be used to leverage capabilities of robotic systems in unstructured environments such as in [18]. Such sensors also encourage the community to develop useful datasets for CNN training. The point cloud data of the popular Cornell Grasp Dataset (CGD) [56] was generated using a Microsoft Kinect sensor. CGD includes 885 RGB images of about 240 objects along with their point cloud data. In addition to that, each image has their own valid and invalid grasp rectangles labelled. For each instance there are about five valid and four invalid grasp rectangles labelled. A major challenge in deep learning is the requirement of higher volumes of training data to avoid overfitting. Even though the CGD totals up to about 8019 ground truth instances, the amount is not sufficient for modern CNN architectures to be trained without experiencing overfitting. Therefore it has become the norm in the deep learning community to use pre-trained neural networks for transfer learning. In transfer learning, the convolutional overhead of the larger neural networks are used as fixed feature extractors for the specific application. This method prohibits the convolutional layers from training and allows the lower level fully connected layers to be trained. This approach also follows the concept that the convolutional base that is trained with larger image datasets is universal enough for many domains thus the learning is easily transferred to domain specific data [10].

Although the limited availability of training data is tackled with transfer learning, data augmentation is also a necessary step to avoid overfitting. While there are computational limitations of loading larger datasets during the training, modern deep learning tools provide real-time data augmentation tools for augmentation of images in mini-batches rather than augmenting the complete datasets. While [19] and [10] were able to achieve much better results in their own contexts, it is not evident how each of them would perform relative to each other under common rules. In addition to that, there have been many improvements since then to DCNN architectures. Therefore, the authors believe that there should be a comparative study between a set of pre-trained network models to clearly determine which performs better while monitoring how their differences build benefits for the original purpose.

The authors collectively acknowledge the recent developments of robotic gasp detection work such as [10,18–20,32]. Compared to the numerous iterations of a sliding window approach [18], the one-shot detection approach would produce much faster results as in [10,19]. One important assumption that is necessary for the one-shot detection method is that there is only one grasp per each image of the object. According to the previous sections a learning algorithm's effectiveness greatly relies on a large training dataset. Creating such a labelled dataset has few challenges, but two of them stands out according to [33]. They are: the objects can be grasped in multiple ways, which makes exhaustive labelling impossible, and human understanding of object grasping is always biased on the semantics [33]. Therefore reinforcement learning techniques could be implemented to automate certain elements of the grasp detection work. It is important to note that the computational time required for such attempts are higher. And if the robust grasping is necessary, it is important to use an extensive dataset such as the Dex-Net 2.0 [27]. Given the requirement of a generalised robotic solution for a grasp detection implementation, the CGD [56] stands out for its adaptability for initial

experimentation. Additionally, it is worthwhile to explore the methods to populate a dataset for grasp detection similar to the CGD, but using methods such as [33,57] so that much deeper DCNN networks could be used with training, with the potential to achieve much improved accuracies in the resulting system.

Author Contributions: S.C. is the first author and contributed more than 75% of content and material for the article. A.R. is the principal supervisor of S.C. and is the second author. D.C. is the collaborative supervisor of S.C. and the third author of the article. A.R. and D.C. both contributed substantively for content refining and editing.

Conflicts of Interest: The authors declare no conflict of interest as the work was not particularly sponsored by any government, organisation, nor an individual.

Abbreviations

The following abbreviations are used in this manuscript:

AI:	Artificial Intelligence
APC:	Amazon Picking Challenge
CGD:	Cornell Grasp Dataset
CNN:	Convolutional Neural Networks
DCNN:	Deep Convolutional Neural Networks
DDD:	Depth in 3 Channels (Depth Depth Depth) Image
Dex-Net:	Dexterity Network
DOF:	Degrees of Freedom
GQ-CNN:	Grasp Quality Convolutional Neural Networks
GWS:	Grasp Wrench Space
MRI:	Magnetic Resonance Imaging
RGB:	Red Green Blue Image
RGB-D:	Red Green Blue Depth Image
RGD:	Red Green Depth Image
SGD:	Stochastic gradient decent

References

1. Lopes, M.; Santos-Victor, J. Visual Learning by Imitation with Motor Representations. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **2005**, *35*, 438–449.
2. Ju, Z.; Yang, C.; Li, Z.; Cheng, L.; Ma, H. Teleoperation of Humanoid Baxter Robot Using Haptic Feedback. Multisensor Fusion and Information Integration for Intelligent Systems (MFI), 2014.
3. Konidaris, G.; Kuindersma, S.; Grupen, R.; Barto, A. Robot learning from demonstration by constructing skill trees. *International Journal of Robotics Research (IJRR)* **2011**, *31*, 360–375.
4. Kober, J.; Peters, J. Imitation and reinforcement learning. *IEEE Robotics & Automation Magazine* **2010**, *17*, 55–62.
5. Peters, J.; Lee, D.D.; Kober, J.; Nguyen-Tuong, D.; Bagnell, A.; Schaal, S. Robot Learning. *Springer Handbook of Robotics* 2^{nd} *Edition* **2017**, pp. 357–394.
6. Mathworks. Deep Learning in Matlab. https://au.mathworks.com/help/nnet/ug/deep-learning-in-matlab.html. Accessed: 2017-04-28.
7. Chen, Z.; Zhang, T.; Ouyang, C. End-to-End Airplane Detection Using Transfer Learning in Remote Sensing Images. *Remote Sensing* **2018**, *10*.
8. Rosenberg, I.; Sicard, G.; David, E.O. End-to-End Deep Neural Networks and Transfer Learning for Automatic Analysis of Nation-State Malware. *Entropy* **2018**, *20*.
9. Bicchi, A.; Kumar, V. Robotic grasping and contact: a review. Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), 2000, Vol. 1, pp. 348–353.

10. Kumra, S.; Kanan, C. Robotic Grasp Detection using Deep Convolutional Neural Networks. *CoRR* **2016**, *abs/1611.08036*, [1611.08036].
11. Ju, Z.; Yang, C.; Ma, H. Kinematics Modeling and Experimental Verification of Baxter Robot. Chinese Control Conference (CCC), 2014.
12. Billard, A.G.; Calinon, S.; Dillmann, R. Learning from Humans. *Springer Handbook of Robotics* 2^{nd} *Edition* **2017**, pp. 1995–2014.
13. Jeon, M. Robotic Arts: Current Practices, Potentials, and Implications. *Multimodal Technologies and Interaction* **2017**, *1*.
14. Tobin, J.; Zaremba, W.; Abbeel, P. Domain Randomization and Generative Models for Robotic Grasping. *CoRR* **2017**, *abs/1710.06425*, [1710.06425].
15. Saxena, A.; Driemeyer, J.; Kearns, J.; Ng, A.Y. Robotic Grasping of Novel Objects. In *Advances in Neural Information Processing Systems 19*; Schölkopf, B.; Platt, J.C.; Hoffman, T., Eds.; MIT Press, 2007; pp. 1209–1216.
16. Saxena, A.; Driemeyer, J.; Ng, A.Y. Robotic Grasping of Novel Objects using Vision. *The International Journal of Robotics Research* **2008**, *27*, 157–173, [https://doi.org/10.1177/0278364907087172].
17. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press, 2016. http://www.deeplearningbook.org.
18. Lenz, I.; Lee, H.; Saxena, A. Deep Learning for Detecting Robotic Grasps. *Int. J. Rob. Res.* **2015**, *34*, 705–724.
19. Redmon, J.; Angelova, A. Real-Time Grasp Detection Using Convolutional Neural Networks. *CoRR* **2014**, *abs/1412.3128*, [1412.3128].
20. Basalla, M.; Ebert, F.; Tebner, R.; Ke, W. Grasping for the real world (Greifen mit Deep Learning). https://www.frederikebert.de/abgeschlossene-projekte/greifen-mit-deep-learning/. Accessed: 2017-02-13.
21. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Jozefowicz, R.; Kaiser, L.; Kudlur, M.; Levenberg, J.; Mané, D.; Monga, R.; Moore, S.; Murray, D.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Talwar, K.; Tucker, P.; Vanhoucke, V.; Vasudevan, V.; Viégas, F.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y.; Zheng, X. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Software available from tensorflow.org.
22. Bergstra, J.; Breuleux, O.; Bastien, F.; Lamblin, P.; Pascanu, R.; Desjardins, G.; Turian, J.; Warde-Farley, D.; Bengio, Y. Theano: a CPU and GPU Math Expression Compiler. Proceedings of the Python for Scientific Computing Conference (SciPy), 2010. Oral Presentation.
23. Chollet, F.; others. Keras. https://keras.io, 2015.
24. Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; Darrell, T. Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093* **2014**.
25. Redmon, J. Darknet: Open Source Neural Networks in C. http://pjreddie.com/darknet/, 2013–2016.
26. NVIDIA. NVIDIA ISAAC Platform for Robotics. https://www.nvidia.com/en-us/deep-learning-ai/industries/robotics/. Accessed: 2017-12-23.
27. Mahler, J.; Liang, J.; Niyaz, S.; Laskey, M.; Doan, R.; Liu, X.; Ojea, J.A.; Goldberg, K. Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics. *CoRR* **2017**, *abs/1703.09312*.
28. Bohg, J.; Morales, A.; Asfour, T.; Kragic, D. Data-Driven Grasp Synthesis-A Survey. *IEEE Transactions on Robotics* **2014**, *30*, 289–309.
29. Jiang, Y.; Moseson, S.; Saxena, A. Efficient grasping from RGBD images: Learning using a new rectangle representation. 2011 IEEE International Conference on Robotics and Automation, 2011, pp. 3304–3311.
30. Zhang, F.; Leitner, J.; Milford, M.; Upcroft, B.; Corke, P. Towards Vision-Based Deep Reinforcement Learning for Robotic Motion Control. Australasian Conference on Robotics and Automation, 2015.
31. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; Hassabis, D. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533.

32. Watson, J.; Hughes, J.; Iida, F. Real-World, Real-Time Robotic Grasping with Convolutional Neural Networks. Towards Autonomous Robotic Systems; Gao, Y.; Fallah, S.; Jin, Y.; Lekakou, C., Eds.; Springer International Publishing: Cham, 2017; pp. 617–626.
33. Pinto, L.; Gupta, A. Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours. 2016 IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 3406–3413.
34. Wei, J.; Liu, H.; Yan, G.; Sun, F. Robotic grasping recognition using multi-modal deep extreme learning machine. *Multidimensional Systems and Signal Processing* **2017**, *28*, 817–833.
35. Wang, J.; Hu, Q.; Danchi. A Lagrangian network for kinematic control of redundant robot manipulators. *IEEE Transactions on Neural Networks* **1999**, *10*, 1123–1132.
36. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *CoRR* **2015**, *abs/1512.03385*, [1512.03385].
37. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*; Pereira, F.; Burges, C.J.C.; Bottou, L.; Weinberger, K.Q., Eds.; Curran Associates, Inc., 2012; pp. 1097–1105.
38. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.S.; Berg, A.C.; Li, F. ImageNet Large Scale Visual Recognition Challenge. *CoRR* **2014**, *abs/1409.0575*, [1409.0575].
39. Mahler, J. Releasing the Dexterity Network (Dex-Net) 2.0 Dataset for Deep Grasping. http://bair.berkeley.edu/blog/2017/06/27/dexnet-2.0/. Accessed: 2017-07-06.
40. Johns, E.; Leutenegger, S.; Davison, A.J. Deep Learning a Grasp Function for Grasping under Gripper Pose Uncertainty. *CoRR* **2016**, *abs/1608.02239*, [1608.02239].
41. Viereck, U.; ten Pas, A.; Saenko, K.; Jr., R.P. Learning a visuomotor controller for real world robotic grasping using easily simulated depth images. *CoRR* **2017**, *abs/1706.04652*, [1706.04652].
42. Amazon Robotics. Amazon Picking Challenge 2016. https://www.amazonrobotics.com/#/pickingchallenge. Accessed: 2017-07-06.
43. Schwarz, M.; Milan, A.; Lenz, C.; Muñoz, A.; Periyasamy, A.S.; Schreiber, M.; Schüller, S.; Behnke, S. Nimbro Picking: Versatile Part Handling for Warehouse Automation. IEEE International Conference on Robotics and Automation (ICRA), 2017.
44. Johnson, J.; Karpathy, A.; Fei-Fei, L. DenseCap: Fully Convolutional Localization Networks for Dense Captioning. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.
45. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature: Macmillan Publishers Limited* **2015**, pp. 436–444.
46. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.E.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going Deeper with Convolutions. *CoRR* **2014**, *abs/1409.4842*.
47. Szegedy, C.; Ioffe, S.; Vanhoucke, V. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *CoRR* **2016**, *abs/1602.07261*, [1602.07261].
48. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception Architecture for Computer Vision. *CoRR* **2015**, *abs/1512.00567*, [1512.00567].
49. Lee, J.; Ryoo, M.S. Learning Robot Activities from First-Person Human Videos Using Convolutional Future Regression. *CoRR* **2017**, *abs/1703.01040*.
50. Polydoros, A.S.; Nalpantidis, L.; Krüger, V. Real-time deep learning of robotic manipulator inverse dynamics. IEEE International Conference on Intelligent Robots and Systems (IROS), 2015.
51. Rethink Robotics. Baxter Collaborative Robots for Industrial Automation. http://www.rethinkrobotics.com/baxter/. Accessed: 2017-03-02.
52. Xia, Y.; Wang, J. A dual neural network for kinematic control of redundant robot manipulators. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **2001**, *31*, 147–154.
53. Ding, H.; Wang, J. Recurrent neural networks for minimum infinity-norm kinematic control of redundant manipulators. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* **1999**, *29*, 269–276.
54. Brownlee, J. Supervised and Unsupervised Machine Learning Algorithms. http://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/. Accessed: 2017-07-03.
55. Brownlee, J. What is the Difference Between Test and Validation Datasets? https://machinelearningmastery.com/difference-test-validation-datasets. Accessed: 2017-12-03.

56. Cornell University. Robot Learning Lab: Learning to Grasp. http://pr.cs.cornell.edu/grasping/rect_data/data.php. Accessed: 2017-04-12.
57. Levine, S.; Pastor, P.; Krizhevsky, A.; Quillen, D. Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection. *CoRR* **2016**, *abs/1603.02199*, [1603.02199].
58. Gandhi, D.; Pinto, L.; Gupta, A. Learning to Fly by Crashing. *CoRR* **2017**, *abs/1704.05588*, [1704.05588].