NeuroEvolution of Capsule Networks for Computer-Aided Laparoscopy

Muhammad Adil Raja ¹, Roisin Loughran ¹, and Fergal Mc Caffery ¹

¹Affiliation not available

December 7, 2023

Abstract

Of an order of 330 million surgeries are performed worldwide every year. Yet there is a backlog of around 150 million pending surgeries annually. Surgical robotics is becoming a lot more sophisticated by the day. Much of this increasing success is due to advances in Computer Vision (CV). CV allows the tracking of tools, detection of organs, and a description of the phase of surgery that enables a surgeon to perform the delicate art of surgery with much greater precision and efficiency. These advancements have also enabled remote robotic surgery in which a patient and the surgeon can be far apart from each other geographically.

In this paper, we report promising results of evolving the architecture of capsule networks for tool classification for Computer-Aided Laparoscopy (CAL).
NeuroEvolution of Capsule Networks for Computer-Aided Laparoscopy

Muhammad Adil Raja, Róisín Loughran, Fergal McCaffery
Regulated Software Research Center (RSRC)
Dundalk Institute of Technology (DkIT)
Dundalk, Ireland
adil.raja, roisin.loughran, fergal.mccaffery@dkit.ie

Abstract—Of an order of 330 million surgeries are performed worldwide every year. Yet there is a backlog of around 150 million pending surgeries annually. Surgical robotics is becoming a lot more sophisticated by the day. Much of this increasing success is due to advances in Computer Vision (CV). CV allows the tracking of tools, detection of organs, and a description of the phase of surgery that enables a surgeon to perform the delicate art of surgery with much greater precision and efficiency. These advancements have also enabled remote robotic surgery in which a patient and the surgeon can be far apart from each other geographically.

In this paper, we report promising results of evolving the architecture of capsule networks for tool classification for Computer-Aided Laparoscopy (CAL).

Keywords—Computer-aided laparoscopy, Capsule networks, Neural architecture search, Neuroevolution, Surgical tool classification.

I. INTRODUCTION

Computer-Aided Laparoscopy (CAL) has come a long way since its inception in the 1980s. Owing to advances in robotics, haptics and recently also to Computer Vision (CV) the field has grown into a multi-billion dollar enterprise. Nowadays, a surgeon sitting in their Operation Room (OR) in Europe can perform surgery over a patient lying in the USA. CAL tools, equipped with modern CV based object detection and tracking tools, can help in the real-time identification of surgical tools, parts of the human body they are acting on, and even the phase of the surgery. Both accuracy and computational efficiency are sought after while developing such tools. Owing to advances in High Performance Computing (HPC) hardware and Machine Learning (ML) algorithms for CV both the accuracy and efficiency of the resulting solutions have increased tremendously over the years. Until recently, the typical choice for ML applied to CV tasks has generally been Convolutional Neural Networks (CNNs).

Technical limitations of CNNs have been reported in recent years [1]. As a consequence, capsule networks have been proposed as a possible remedy [2]. Instead of giving a scalar output at each computational unit as in traditional Artificial Neural Networks (ANNs) and CNNs, capsule networks output a vector of values at every unit. Each vector of outputs is encapsulated in a so-called capsule. Each element of the encapsulated vector encodes in it a certain trait of the objects in the input image. Traits could be information about the position, shape, and orientation, as well as the scale, shear, and stretch along an axis. A central theory in devising capsule networks is that as objects are made of parts, a good model should capture the object-part relationship. This is to say that the parts that make up an object should be identified by the model first. Then that information about parts should be used to render the objects. Capsule networks do what is sometimes referred to as inverse rendering in computer graphics. Given the numerical code of a graphic scene, in terms of the scale, shape, orientation, position, etc. of various objects, a scene can be rendered on the screen. To the converse, capsule networks are provided with a visual scene and create the numerical code of the scene in terms of objects and the parts they are made of. Theoretically, capsule networks are considered to be more robust and fool-proof than conventional CNNs. However, practically they have revealed mixed results so far that have been, on occasions, quite promising.

Neuroevolution is the craft of creating and optimizing ANNs using an Evolutionary Algorithm (EA) [3]. A Genetic Algorithm (GA) is used to create the optimal structure and architecture of the target ANN. This technique has proven to be beneficial for such tasks as reported by various research studies [4]. In designing capsule networks, however, their application is quite recent [5]. The full potential of neuroevolution has still not been realized due to the stringent demands that the evolution of the capsules places on hardware resources. In this work, we have applied neuroevolution of capsule networks for developing models for CAL.

The rest of this paper is organized as follows: In section II we give a brief background about the history and practice of CAL. Section III presents our experimental details. In section IV results are presented. Finally, section ?? concludes this paper.

II. COMPUTER AIDED SURGERY AND LAPAROSCOPY

Laparoscopy is a surgical procedure that allows the surgeon to have access to the inside of a patient’s body without making large incisions. For many surgical operations, laparoscopy is a much preferable approach as compared to open surgery. It
requires a small incision, reduces the healing time, lessens blood loss, and affects the recovery positively [6]. The traumatic effects of surgery are lower due to it. The possibility of infections is also less. However, with its myriad of benefits also come some critical technical challenges, especially for surgeons. The main challenges among these are hand-eye coordination and a narrow field of view. Automatic tool detection, real-time body part identification, surgical phase information, and 3D pose estimation can assist the surgeon while performing CAL. These tasks can become highly cumbersome due to the presence of smoke, occlusion, blood, shadows, specularities, motion blur, cleaning gauze, and intricate background textures [7]. All of these challenges call for the development of more refined CV systems for tracking tools and identifying body parts and surgical phases in CAL.

The advent of deep learning and CNNs brought about a paradigm shift in CV during the past decade [8]. The wide availability of high-performance computing (HPC) infrastructure and ever-increasing graphical processing unit (GPU) power paved the way for rapid innovation of CV applications. Disciplines of medical imaging and robotic surgery were also beneficiaries of this. Initially capable of solving classification problems, CNNs were later adapted to achieve the capability of object detection [9]. In the former case, an algorithm is only able to predict what class of objects are present in an image. In the latter case, however, the algorithm is additionally capable of providing precise information about the position of various objects in the image [10]. It is due to the ability of object detection that robotic surgery and CAL have been taking technological strides. As algorithms get better, in addition to detecting objects with increasing accuracy, their computational efficiency also improves.

CNNs have led to tremendous advances in innovations and they have been used to develop CV applications for many problem domains [11]–[13]. In recent years, however, CNNs were criticized among the academic community for some of their critical technical shortcomings [2]. Professor Geoffrey Hinton, one of the initial proposers of CNNs has been among the chief critics. One of the problems identified was that CNNs cannot understand images in terms of objects and their parts. It is noted that CNNs are good at dealing with translations, meaning that they can detect objects irrespective of where they appear in an image. However, they are not good at dealing with objects with changing viewpoints [2].

Unlike traditional ANNs and CNNs, capsule networks are composed of encapsulated neurons at every level of data flow. The encapsulated neurons output a vector of values for every important part of the scene in the image. This vector of values encodes important information related to different aspects or traits of an object. These traits may include information related to the orientation and scale of the underlying object in the image. Capsules at the earlier layers of the networks learn to identify parts of bigger objects. Similarly, capsules at the latter layers of the network learn to identify larger objects. Parts are attributed to their respective wholes through a routing mechanism in which the part that fits most into a whole is ascribed to it.

It has been noted that the way capsule networks can learn to identify objects is more natural and closer to how the human visual system works in conjunction with the human mental ability to perceive visual percepts [1]. CNNs are inherently incapable of perceiving percepts according to humans. CNNs have only one percept. If the position of the object in the scene is changed, CNNs can detect the object at the new location. However, if the orientation of the object is changed, CNNs cannot detect the object as they are not designed to detect such changes. An example of this is to turn a chair upside down. CNNs are incapable to detect this object as a chair unless they are trained on a very large amount of images containing chairs at all sorts of orientations. Capsule networks, as they encode information related to the rendering of objects on screens, are supposed to be able to detect objects despite such changes in orientation and scale. Much inspiration for devising capsule networks was drawn from how rendering is achieved in computer graphics.

Designing and refining any ANNs in the traditional manner requires a huge amount of computational power. Neuroevolution is a technique to evolve ANNs using an EA [14]. The EA is employed to explore the design space of ANNs. As in any EA, an initial population of randomly drawn ANNs is used. The population is treated with evolutionary operators of selection, recombination, mutation, replacement and evaluation in a loop over a certain number of generations. Evolution is usually stopped after a certain number of generations when a more refined ANN has been found in the population from one of the later generations of evolution.

Evolution can be (and has been) used to exploit three aspects of an ANN’s design issues [15]. These are: researching the structure of the ANN, tuning of the hyper-parameters, as well as optimization of the coefficients of the neurons. In the past neuroevolution was usually only applied to the evolution of shallow ANNs [16]. Recent studies in the past decade have also observed the development and practical viability of deep learning CNNs [4], [17]. Eventually, neuroevolution of CNNs also became a reality. Impressive results have also been reported on the performance of deep neuroevolution for various engineering and scientific disciplines [18]–[20]. Further notable applications of neuroevolution include the development of a system for team games based on neuroevolution [21], autonomous robot navigation [20], evolution of controllers for Unmanned Aerial Vehicles (UAVs) [22]. Neuroevolution has also found applications in the field of medical device software innovation. Applications include the development of systems for X-Ray object detection [23], heart disease detection [24], detection of corona virus disease [25], breast cancer data analytics [26], pulmonary nodule detection [27], detection of motor impairment among children [28], and robotic heart surgery [29].

Quite recently an implementation of neuroevolution of capsule networks has also emerged [5]. This implementation is based on the TensorFlow library of ML tools and algorithms by Google [30]. It is known as NASCaps and it can be found
TABLE I: Hyperparameters of Training Experiments

<table>
<thead>
<tr>
<th>Parameter</th>
<th>CapsNet</th>
<th>NASCaps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epochs</td>
<td>25</td>
<td>15</td>
</tr>
<tr>
<td>Batch Size</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Timeout</td>
<td>600</td>
<td>600</td>
</tr>
<tr>
<td>GPUs</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

IV. RESULTS

In this section we report the results of the aforementioned experiments related to classifying surgical tools used in CAL of cholecystectomy. Fig. 2 shows the confusion matrix related to the randomly generated capsule network. More specifically, Figs. 2a, 2b and 2c show confusion matrices for training data, validation data used during training, and unseen test data used to validate the trained model. The results are reported in terms of percentages of correctly identified images with respect to the total number of images in that particular class or category. The benefit of showing results in terms of percentages is that it can be seen what proportion of the data is classified correctly and what proportion is misclassified as something else precisely. However, showing classification results in a proportionate sense has a problem that results cannot be seen in terms of how the overall data was classified.

Fig. 3 shows the same confusion matrices but in terms of absolute numbers of data instances used for training, validation, and testing. It would be pertinent here to explain the pictorial illustrations of a confusion matrix. Each row of the matrix tells us about what the actual tool was classified as by the model. Thus, for instance, if we look at the test data, 67 images containing the grasper were correctly classified as a grasper. However, the grasper was misclassified as a hook, scissors, or a clipper 0, 2, and 4 times respectively. The entire sum of all elements in a row is the total number of instances of that part in the data. It can’t be said with certainty that why classification accuracy is low for certain tools and high for others. However, a confusion matrix could be useful to gain certain insights about the data and the performance of the model. For instance, the tools that had lower numbers of samples in the overall data had poorer classification accuracy and vice versa. This can also be seen in Table II where exact number of instances of each tool in the test data are given, along with the proportion of each tool’s presence in the dataset as well as accuracy of the model at predicting that tool correctly. Scissors, for instance, have a substantially low
representation in the data in terms of number of instances. And the ability of the model to predict scissors precisely is also low. And when the scissors are misclassified, it happens almost equally badly as other tools.

TABLE II: Statistics Related to Test Data

<table>
<thead>
<tr>
<th>Tool</th>
<th>Number of instances</th>
<th>% of the total data</th>
<th>% accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grasper</td>
<td>210</td>
<td>35.06</td>
<td>93.33</td>
</tr>
<tr>
<td>Hook</td>
<td>229</td>
<td>38.23</td>
<td>89.96</td>
</tr>
<tr>
<td>Scissors</td>
<td>57</td>
<td>9.52</td>
<td>45.61</td>
</tr>
<tr>
<td>Clipper</td>
<td>103</td>
<td>17.2</td>
<td>77.67</td>
</tr>
</tbody>
</table>

Fig. 4 shows the results of the ground truth versus the predicted values of the tools for training, and test data respectively. The main purpose of this figure is to show some examples of how the model makes errors in its judgments about tool classification. The implementation of the capsule networks was based on an Auto-Encoder (AE) architecture. The figure also shows the output of the decoder for the curious reader.

V. CONCLUSIONS

In this paper, we have presented a novel technique to classify surgical tools in a CAL setting. Particularly, data related to cholecystectomy was used to evolve capsule networks using a simplistic GA. The software that we used for this purpose is based on the TensorFlow library, named NASCaps. A total of four different tools were present in the data, comprising four classes. These include a grasper, a clipper, a hook, and a pair of scissors. Results of classification models have been reported in this paper, which, although quite promising and interesting, are not very impressive. For one thing, they are inferior to the contemporary State-of-The-Art (SoTA) models. However, this is a well-known issue with capsule networks currently. It is expected that future versions of capsule networks will have improved accuracy. Moreover, nowadays, for models to be useful, they should additionally be able to do object detection in addition to classification.

The implementation of NASCaps, however, has some problems. Firstly, nowadays, for CV models to be useful, they should additionally be able to do object detection in addition to classification. Classification alone is not enough as it is not enough just to know the objects in a visual scene. In order to be useful, a model should be able to precisely tell where that object is in the image. Other problems with NASCaps relate to the implementation issues of the software and are not discussed here for the purpose of brevity. For instance, although it allows to evolve a capsule network using a GA with good classification accuracy and performance capabilities, the model cannot be reused conveniently after it has been trained. To this end, it allows us to save the model in the Hierarchical Data Format (Version 5) (HDF5). The way it can store the model is either the whole model is stored, or only the weights are stored. In the latter case, one has to know the structure of the model to retrieve its complete definition from the disk. It saves the model’s weights as it uses a few newer objects that are not part of TensorFlow. In order to retrieve the whole
model, one needs to know the integer-based genotype that was responsible for generating its particular structure. However, when it comes to that, the software gives an error message stating that the mapping of weights to the model could not be achieved. There was little to no support for the system as well.

An additional problem with NASCaps is that its implementation is based on a very simplistic GA having a minimalist design. As a matter of fact, it is merely a proof of concept implementation to show the viability of Neural Architecture Search (NAS) for the evolution of capsule networks. Although it performs well, the full potential of recent cutting-edge research in the Evolutionary Computing (EC) community has not been taken into account.

Given these and other shortcomings of NASCaps, we aspire to undertake a full-fledged implementation of neuroevolutionary software for capsule networks in the future. We shall implement software for capsule networks from scratch and augment it with a suitable library for EC. We also aspire to tailor capsule networks for object detection.

**REFERENCES**

Fig. 4: Examples of prediction errors made by the model. Predicted versus the actual labels of the tools, as well as the pictures of tools in reality. Alternate (even-numbered) rows show the output of the auto-encoder for the respective input images.


[32]