Applied Deep Learning in Orthopaedics

William Stewart Burton II
University of Denver

Applied Deep Learning in Orthopaedics

Abstract

The reemergence of deep learning in recent years has led to its successful application in a wide variety of fields. As a subfield of machine learning, deep learning offers an array of powerful algorithms for data-driven applications. Orthopaedics stands to benefit from the potential of deep learning for advancements in the field. This thesis investigated applications of deep learning for the field of orthopaedics through the development of three distinct projects.

First, algorithms were developed for the automatic segmentation of the structures in the knee from MRI. The resulting algorithms can be used to accurately segment full MRI scans in a matter of seconds. Reconstructed structures from predicted segmentation maps yielded on average submillimeter geometric errors when compared to geometries from ground truth segmentation maps on a test set. The resulting frameworks can further be applied to develop algorithms for automatic segmentation of other anatomies and modalities in the future.

Next, neural networks (NNs) were developed and evaluated for the prediction of muscle and joint reaction forces of patients performing activities of daily living (ADLs) in a gait lab environment. The performance of these models demonstrates the potential of NNs to supplement traditional gait lab data collection and has implications for the development of new gait lab workflows with less hardware and time requirements. Additionally, the models performed activity classification using standard gait lab data with near-perfect accuracy.

Lastly, a deep learning-based computer vision system was developed for the detection and 6-degree of freedom (6-DoF) pose estimation of two surgical tracking tools routinely used in total knee replacement (TKR). The resulting model demonstrated competitive object detection capabilities and translation error as little as a few centimeters for the pose estimation task. A preliminary evaluation of the system shows promise for its applications in skill assessment and operations research.

The development of these three projects represents a significant step towards the adoption of deep learning methodologies by the field of orthopaedics and shows potential for future additional applications.

Document Type
Thesis

Degree Name
M.S.

Department
Mechanical Engineering

First Advisor
Paul J. Rullkoetter, Ph.D.

Keywords
Computer vision, Convolutional neural networks, Deep learning, Machine learning, Orthopaedics

Subject Categories
Computer-Aided Engineering and Design | Engineering | Mechanical Engineering

Applied Deep Learning in Orthopaedics

A Thesis

Presented to

the Faculty of the Daniel Felix Ritchie School of Engineering and Computer Science

University of Denver

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

William S. Burton II

June 2019

Advisor: Paul J. Rullkoetter

Author: William S. Burton II
Title: Applied Deep Learning in Orthopaedics
Advisor: Paul J. Rullkoetter
Degree Date: June 2019

ABSTRACT

The reemergence of deep learning in recent years has led to its successful application in a wide variety of fields. As a subfield of machine learning, deep learning offers an array of powerful algorithms for data-driven applications. Orthopaedics stands to benefit from the potential of deep learning for advancements in the field. This thesis investigated applications of deep learning for the field of orthopaedics through the development of three distinct projects.

First, algorithms were developed for the automatic segmentation of the structures in the knee from MRI. The resulting algorithms can be used to accurately segment full MRI scans in a matter of seconds. Reconstructed structures from predicted segmentation maps yielded on average submillimeter geometric errors when compared to geometries from ground truth segmentation maps on a test set. The resulting frameworks can further be applied to develop algorithms for automatic segmentation of other anatomies and modalities in the future.

Next, neural networks (NNs) were developed and evaluated for the prediction of muscle and joint reaction forces of patients performing activities of daily living (ADLs) in a gait lab environment. The performance of these models demonstrates the potential of NNs to supplement traditional gait lab data collection and has implications for the development of new gait lab workflows with less hardware and time requirements. Additionally, the models performed activity classification using standard gait lab data with near-perfect accuracy.

Lastly, a deep learning-based computer vision system was developed for the detection and 6-degree of freedom (6-DoF) pose estimation of two surgical tracking tools routinely used in total knee replacement (TKR). The resulting model demonstrated competitive object detection capabilities and translation error as little as a few centimeters for the pose estimation task. A preliminary evaluation of the system shows promise for its applications in skill assessment and operations research.

The development of these three projects represents a significant step towards the adoption of deep learning methodologies by the field of orthopaedics and shows potential for future additional applications.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor, Dr. Paul Rullkoetter, for his support over the past two years and for his faith in my ability to excel in an area of research that is novel to the Center of Orthopaedic Biomechanics.

I’d like to thank Dr. Laz and Dr. Schneider for being generous with their time and serving on my thesis committee.

I must thank Dr. Casey Myers for his guidance and support of the projects that were undertaken in this thesis. I greatly appreciate the opportunity to work with him during the development of these projects.

I would additionally like to thank Dr. Laz for his guidance, both academic and otherwise, during the last six years of my time here at the University of Denver; and believing in my abilities to contribute to the Center as both an undergraduate and graduate student.

Also, I thank Dr. Chadd Clary for his support in the form of new hardware, without which this work would not have been feasible. I thank Dr. Donald Hume for his assistance in the set up associated with the hardware.

I would like to thank my parents for their unconditional support during my time as a graduate student as well as an undergraduate student; and for believing in my potential to succeed.

Lastly, I would like to thank Depuy Synthes and the National Science Foundation for supporting my role as a graduate research assistant.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

1.1 Introduction

“Remember, there will not be an ‘AI industry’. Instead, machine learning and AI will find their way into every problem in every industry.”

Francois Chollet, Google

In 2012, the introduction of AlexNet (Krizhevsky et al., 2012) sparked a renewed interest in deep learning by significantly outperforming state-of-the-art results on the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) (Deng et al., 2009) using a novel convolutional neural network (CNN) trained on multiple graphics processing units (GPUs). Since then, deep learning has made massive strides in computer vision (He et al., 2016a), natural language processing (Gehring et al., 2017; Kalchbrenner et al., 2016), and predictive modeling (Shi et al., 2015). This progress comes as a result of novel algorithms, new and powerful hardware, availability of data, and open-source frameworks that allow for efficient experimentation and implementation.

The success of deep learning has many implications for the field of orthopaedics in the context of solving complex computer vision problems and using predictive modeling to increase efficiency of common workflows. The work presented in this thesis explores three specific applications of deep learning in orthopaedics: automatic segmentation of medical imaging, predictive modeling of patient mechanics, and surgical tool tracking. Application of deep learning algorithms to automatic segmentation can result in faster medical imaging analysis and address the time-intensiveness associated with manual segmentation. Predictive modeling of patient mechanics can circumvent hardware and expertise requirements, as well as time-intensiveness of the standard gait lab workflow. Additionally, tracking of surgical tools has implications for robotic assisted surgery, augmented reality, and assessment of surgical skill and workflow as part of a broader goal of obtaining metrics around surgical operating room (OR) activity. For each of these applications, deep learning systems are developed and their performance is evaluated.

1.2 Objectives

The objectives of this thesis are to:

1. Develop CNNs for automatic segmentation of anatomical structures of the knee from MRI for applications in biomechanics research.
2. Develop NNs for predictive modeling of muscle and joint reaction forces based on patient kinematics, ground reaction forces, and anthropometrics.
3. Develop a deep learning-based computer vision system for detection and 6-Dof pose estimation of surgical tools in real-time.
4. Contribute to the general goal of accelerating the application of deep learning techniques into the field of orthopaedics.

1.3 Thesis Overview

Chapter 2 provides a brief technical review of NNs and relevant concepts.

Chapter 3 presents CNNs for automatic segmentation of medical imaging, in which CNNs were developed with semi-supervised learning methods and evaluated for the automatic segmentation of the structures of the knee from MRI.

Chapter 4 presents predictive modeling of patient mechanics, in which deep learning algorithms were developed to predict patient muscle and joint reaction forces from standard gait lab data.

Chapter 5 presents the development and evaluation of a computer vision system that leverages deep learning and traditional computer vision concepts for the tracking of surgical tools.

Chapter 6 briefly reviews each project along with significance.

CHAPTER 2. A BRIEF REVIEW OF NEURAL NETWORKS

2.1 The Structure of Neural Networks

A NN is a hierarchy of functions that can learn from data. In the field of machine learning, there are four main types of learning: unsupervised, supervised, semi-supervised, and reinforcement learning. The current work focuses on supervised and semi-supervised learning.

Supervised learning represents the situation in which there exists a set of data characterized by N inputs $X \in \mathbb{R}^{A\times N}$ and outputs $Y \in \mathbb{R}^{B\times N}$. In these matrices, $x^{(i)} \in \mathbb{R}^{A}$ and $y^{(i)} \in \mathbb{R}^{B}$ are the i^{th} columns of X and Y, and represent the inputs and outputs of a single instance in the data set. The objective of supervised learning is to develop a relationship that maps X to Y so that, given a new input $x \in \mathbb{R}^{A}$ this mapping can be used to infer the new $y \in \mathbb{R}^{B}$. Sometimes this problem can be solved with a model as simple as a linear regression, where the output Y can be described as a linear function of X:

$$Y = W^{T}X + b \tag{2.1}$$

Where $W \in \mathbb{R}^{A\times B}$ and $b \in \mathbb{R}^{B}$. Sometimes the data can't be described by a linear mapping and complex, non-linear functions are necessary. NNs offer a solution here.

NNs take as input some data $X \in \mathbb{R}^{A \times N}$ and map it to an output using a hierarchy of both linear and non-linear functions. These functions are developed with learnable parameters. For instance, consider, again, a linear regression. In a linear regression the goal is to learn ideal values for W and b by using the given data. In this scenario W and b are learnable parameters.

Fully-connected NNs represent the most fundamental type of NN and are composed of stacked layers that couple matrix transformations with non-linear activation functions. Given a vector $X \in \mathbb{R}^{A}$, the application of a single network layer to the input vector can be defined by:

$$Z = g(W^T X + b) \tag{2.2}$$

This structure is similar to the linear regression with one addition. The function g(.) is a non-linear activation function that is applied to the layer. Non-linear activation functions are important for giving the model the capacity to represent complex functions. Traditionally, the sigmoid function $g(x) = \frac{1}{1+e^{-x}}$ was the activation function of choice but many different functions are now used in practice (He et al., 2015). The matrix transformation $W^T X$ can be interpreted as a set of B nodes in a given network layer (Fig. 2.1). A node in a layer consists of a linear combination of each element of the input vector X, combined with the node's specific bias unit (before applying the non-linear activation). As a result, the layer yields a vector $Z \in \mathbb{R}^{B}$, that can be fed to another hidden layer. Layers are stacked together in a recursive fashion to create complex architectures and in each layer, W and b are parameters that must be tuned to transform the incoming vector in meaningful ways. In the final layer of a fully-connected network, the layer's input is mapped to an output whose dimensions correspond to the prediction task. In the simplest case, regression for predicting $Y \in \mathbb{R}$, the final layer will have dimensionality of 1 and may not be accompanied by the non-linear activation function.

The entire structure of a basic fully-connected NN can be described by recursively applying these layers. A network with 2 sigmoid-activated hidden layers and an output layer may look like this:

$$Z = W_3^T g(W_2^T g(W_1^T X + b_1) + b_2) + b_3 \tag{2.3}$$

As can be seen, the original input vector X is fed to an initial hidden layer, which is fed to a second hidden layer. After the second hidden layer has applied its activation function, the resulting vector is applied to a final output layer (without a non-linear activation) to complete the mapping from X to Y. This represents the structure of a fully-connected NN, which is the most basic class of NN. Realistically, there are many other forms of NNs. For instance, CNNs use a different set of functions that are more conducive for image data and recurrent neural networks (RNNs) are architectures that are conducive for sequence data.

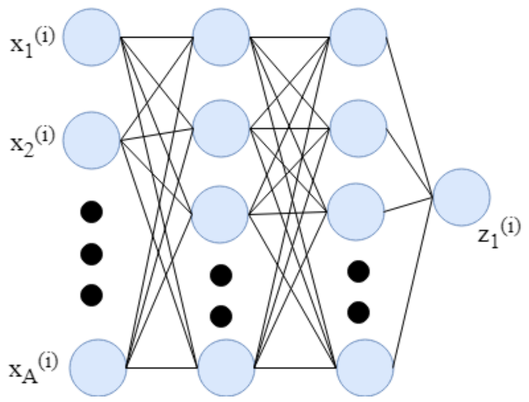


Figure 2.1. A pictorial representation of a fully-connected NN with two hidden layers and a 1D output. Each node in a layer aggregates context from all nodes from the previous layer.

2.2 Training Neural Networks

As a proxy for NNs, it helps to consider the linear regression. In simple linear regression, the goal is to learn the ideal parameters $W \in \mathbb{R}^{A \times B}$ and $b \in \mathbb{R}^{B}$. This goal can be represented as an optimization problem. In order for W and b to be optimal, the difference between $W^T X + b$ and Y should be minimal over the whole training set. This can take the form of the sum of the errors over all examples in the training set.

$$L(W, b) = \underset{W,b}{\operatorname{argmin}} \sum_{i=1}^{N} \left(W^T x^{(i)} + b - y^{(i)} \right)^2 \tag{2.4}$$

The objective function that needs to be minimized is called the loss function, because it measures the error between what the model's predictions are and what the actual ground truth is. The optimal parameters for a linear regression can be found with a closed-form solution using the classic least-squares approach, but can also be found by using gradient-based optimization. To do this, the derivatives of the loss function are computed with respect to each parameter and then the current parameter values are updated in the opposite direction of the gradient. The parameters W and b are consolidated into a single variable to make things easier to manage. Let $W' \in \mathbb{R}^{(A+1)\times B}$ denote this parameter. In order for this matrix to be compatible with the matrix X, a new variable $X' \in \mathbb{R}^{(A+1)\times N}$ is defined where the bottom row is a vector of ones. The new optimization problem is as follows:

$$L(W') = \underset{W'}{\operatorname{argmin}} \sum_{i=1}^{N} \left(W'^{T} x'^{(i)} - y^{(i)} \right)^{2} \tag{2.5}$$

W' is iteratively updated by calculating the current value of the gradient and then using this value to update the parameter based on:

$$W'_{t+1} = W'_{t} - \alpha \frac{\partial L(W')}{\partial W'} \tag{2.6}$$

Where α is a hyperparameter called the step size, or learning rate, that controls how much the parameter can be altered at once. This approach is called gradient descent and is a fundamental concept in optimization. The gradient is computed using calculus:

$$\frac{\partial L(W')}{\partial W'} = 2x'^{(1)}\left(W'^{T} x'^{(1)} - y^{(1)} \right) + \cdots + 2x'^{(N)}\left(W'^{T} x'^{(N)} - y^{(N)} \right) \tag{2.7}$$

$$\frac{\partial L(W')}{\partial W'} = 2\sum_{i=1}^{N} x'^{(i)}\left(W'^{T}x'^{(i)} - y^{(i)}\right) \tag{2.8}$$

The parameters for a linear regression are updated until the loss function ceases to improve. The resulting value of W' is taken as the trained model.

The training of NNs uses the same approach. The gradient of the loss function is used to change all trainable parameters of the model by a small amount at each training step. Applying this approach to NNs is more complicated because there are many parameters that need to be updated. Many times, NNs have millions of parameters. At each training step, the gradient of the loss function must be computed with respect to every trainable parameter before using each parameter's corresponding gradient to update its value. For the 3-layered NN from Equation 2.3, it is necessary to compute $\frac{\partial L(\theta)}{\partial W_1}, \frac{\partial L(\theta)}{\partial W_2}, \frac{\partial L(\theta)}{\partial W_3}, \frac{\partial L(\theta)}{\partial b_1}, \frac{\partial L(\theta)}{\partial b_2}$, and $\frac{\partial L(\theta)}{\partial b_3}$ at every training iteration. These gradients can be computed using the chain rule of calculus. To demonstrate this, the 3-layered NN is represented by the computational graph in Figure 2.2, where operations are nodes in the graph; and inputs and outputs to these operations are edges:

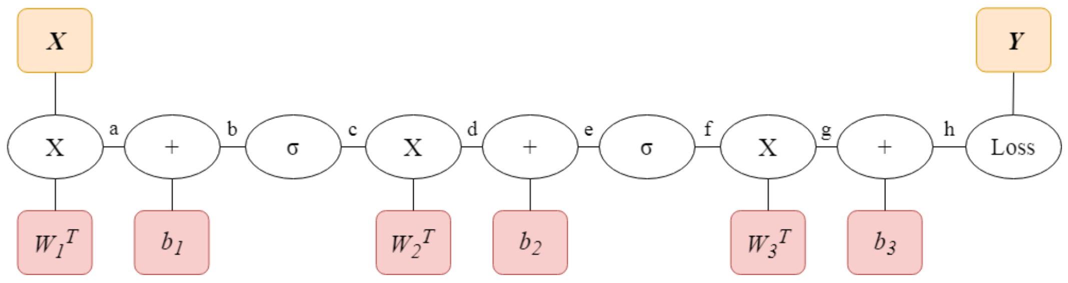


Figure 2.2. A computational graph representation of a 3-layered NN.

The gradient for $\frac{\partial L(\theta)}{\partial b_2}$ is computed by decomposing this value into a product of gradients around local nodes:

$$\frac{\partial L(\theta)}{\partial b_2} = \frac{\partial L(\theta)}{\partial h}\frac{\partial h}{\partial g}\frac{\partial g}{\partial f}\frac{\partial f}{\partial e}\frac{\partial e}{\partial b_2} \tag{2.9}$$

Once the gradients are determined for each parameter, the parameters are updated similar to Equation 2.6. The process of computing the gradients for all parameters in the network is known as backpropagation (Rumelhart et al., 1986).

In summary, it helps to conceptualize the training of NNs by comparing to the numerical training of linear regressions. Training instances are fed through the model (called the forward pass) and the outputs are compared to the ground truth using a cost function. The parameters of the model are then updated based on the gradient of the loss function with respect to each parameter. A single forward pass can be performed using the entire training set at once, a subset of the training set (called a "batch"), or even a single training instance. Much of the time, the batch size is determined based on computational constraints, because each instance in a batch is processed in parallel.

2.3 Convolutional Neural Networks

The NN presented in Equation 2.3 is called a fully-connected NN because every node of a layer is connected to every input to that layer via some weighting. For problems dealing with image data, a fully-connected structure may be extremely computationally expensive because of the number of pixels in images. Therefore, convolutional layers are used instead (Fig. 2.3) (LeCun et al., 1998). Convolutional layers replace the matrix transformation of a fully-connected layer with convolutions. In the case of a 2D image with depth of 1 (i.e. greyscale), a convolutional layer is implemented by sliding a small matrix (known as a convolutional filter) over the input image and taking the sum of elementwise products between the filter's and image's elements. The output of a convolutional filter A with dimensions $[m, n]$ applied at location (x, y) on an image I is defined by:

$$I(x, y) * A = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} A(i, j) \times I(x + i, y + j) \tag{2.8}$$

A NN that is composed of convolutional layers (but may also contain fully-connected layers) is a CNN. In a CNN, the convolution can be followed by the addition of a bias term, the application of a non-linear activation function, output normalization techniques, as well as pooling operations, which aggregate context into a more compressed dimensionality. A CNN that does not make use of any fully-connected layers is called fully-convolutional.

The work in chapers 3-5 of this thesis utilize CNNs. CNNs can be applied to data with any number of dimensions. For a 2D image, a 2D convolutional filter is applied over the height and width of the image. For a 3D image, a volumetric convolutional filter is applied the height, width, and depth of the volume. 1D convolutions can be applied to 1D data such as a time series.

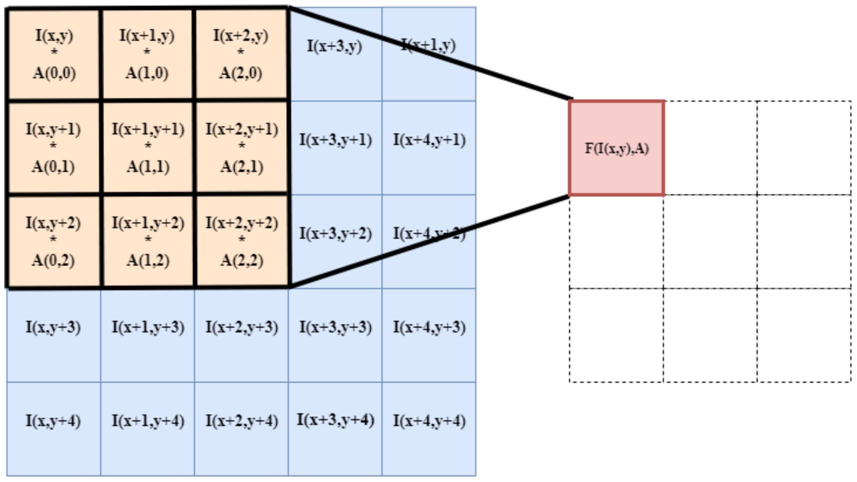


Figure 2.3. A 2D convolutional filter applied to a patch on a 2D image with depth of 1.

2.4 Recurrent Neural Networks

Recurrent neural networks are NNs that are designed for time series data (Hochreiter & Schmidhuber, 1997). Time series data is unique because it may be of variable dimensions. In contrast, fully-connected NNs are designed to only handle data of constant dimensionality.

At a time step t, an RNN cell stores compressed information from previous time steps $1\!:t-1$ in a vector called the hidden state. The hidden state can be used to inform predictions at the current time step, and can be used in subsequent time steps. The hidden state at the current time step is a function of the input to the NN at time t as well as the hidden state from the previous time step $t-1$. Suppose $h_t \in \mathbb{R}^N$ and $X_t \in \mathbb{R}^M$ are the hidden state and input to the NN at time t. For a basic “vanilla” RNN cell, the hidden state is updated based on:

$$h_t = \tanh(A^T X_t + B^T h_{t-1} + c) \tag{2.9}$$

Where A, B and c are parameters to be learned. These parameters are shared over all time steps. The hidden state can be passed to additional fully-connected layers to generate some output. RNNs are usually conceptualized as a computation graph "unrolled" through time with shared weights over each increment (Fig. 2.4).

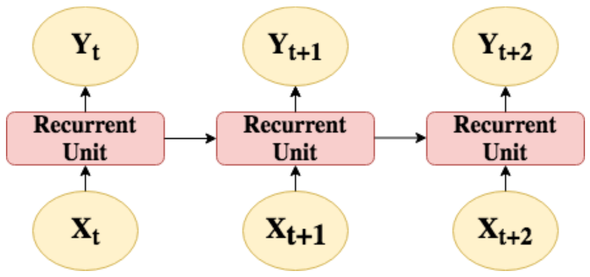


Figure 2.4. A representation of an RNN computation graph unrolled through time. Computations are made at each time step using the same parameters.

2.5 Implementing Neural Networks

The implementation of NNs is complicated because of backpropagation. A gradient value must be tabulated for every parameter in the NN during training and the manual computation of these gradients can become very tedious. Fortunately, current deep learning libraries (Abadi et al., 2016; Chollet, 2015; Paszke et al., 2017) compute these gradients automatically using a concept called auto-differentiation (Kucukelbir et al., 2017). As a result, training NNs is reduced to stacking together the layers that compose the architecture, defining a loss function and optimization scheme, and then feeding training batches to the computational graph. Computational capabilities of GPUs are leveraged in the case of heavy architectures. The advent of popular deep learning frameworks has allowed for convenient and scalable implementations of NNs.

CHAPTER 3. CONVOLUTIONAL NEURAL NETWORKS FOR AUTOMATIC SEGMENTATION OF THE KNEE FROM MAGNETIC RESONANCE IMAGING

3.1 Segmentation of Medical Imaging

Segmentation of medical imaging is an important task in orthopaedics workflows that rely on medical images to represent subject-specific geometries. Segmentation consists of annotating an image by assigning every pixel in the image with a semantic class. The image can be a three-dimensional volume such as a computed tomography (CT) scan. Segmentation of medical imaging is used to build computational models of anatomical structures. The resulting geometries are used in a variety of applications including statistical models to assess morphological variation through a population and finite element (FE) modeling to couple subject-specific kinematics and geometries.

Reconstructed geometries are used to develop statistical models to describe variations throughout a population. Smoger et al. (2015) used principal component analysis to characterize relationships between kinematics of the knee and the shape of the bones and cartilage of the knee. Sintini et al. (2018) developed a statistical shape model of the proximal humerus to assess anatomical differences throughout a population. Burton et al. (2019) used principal component analysis to describe the variation of morphology as well as material property distributions of healthy scapulae. The use of

SSMs is important in the context of informing implant designs to best fit a population. The development of these statistical models relies on computational geometries rebuilt from segmented medical images.

Segmented geometries are used to develop subject-specific FE models. Ali et al. (2016) developed and validated an FE model of the knee based on subject-specific geometries by segmenting the femur, patella, and tibia from CT scans; and the corresponding articular cartilages from magnetic resonance imaging (MRI).
Hume et al. (2019) validated a muscle-driven FE model of the knee developed from segmented CT and MRI scans.

Segmentation of medical imaging is also used for characterizing kinematics by registering and tracking geometries in fluoroscopy images (Ivester et al., 2015). In this workflow, a patient performs an ADL, such as a lunge, in front of a biplane fluoroscopy. A sequence of frames is captured throughout the activity. The bones of the joint of interest are captured using CT and segmented to recreate the geometries of the joint. Then, the geometries are semi-automatically registered to each pair of frames to locate the bones in a 3-dimensional coordinate system. The transformations of the bones throughout the ADL are used to quantify kinematics (Myers et al., 2011a, 2011b, 2012).

The applications of medical imaging segmentation extend beyond these examples, to quantitative anatomical studies (Yu et al., 2017) and pre-surgical planning. Segmentation of medical imaging is traditionally performed manually, which is time-intensive and requires significant expertise. For instance, segmentation of the femoral cartilage from an MRI scan consisting of 160 sagittal slices can require multiple hours to complete. Accordingly, methods for improving efficiency of medical imaging segmentation have been proposed. These methods range from semi-automatic (Vezhnevets & Konouchine, 2005; Zhu et al., 2014) to fully automatic (Atkins & Mackiewich, 2002).

Recently, CNNs have yielded state-of-the-art results in computer vision tasks, and have been successfully applied to automatic segmentation of images (Noh et al., 2015; Badrinarayanan et al., 2017; Shelhamer et al., 2017). These approaches have further been applied to the medical imaging domain (Ronneberger et al., 2015; Christ et al., 2016; Çiçek et al., 2016; Kamnitsas et al., 2016). Most of these works were performed in the domain of supervised learning, in which CNNs were trained with fully annotated data. A drawback associated with supervised learning is the cost of obtaining ground truth data. This limitation is especially prevalent in the context of segmentation, where annotation of a single MRI scan may take hours; and in the medical imaging domain, where medical imaging data may be hard to access.

Semi-supervised learning provides an alternative to supervised learning by leveraging both labeled and unlabeled data for training deep neural networks. As with other machine learning problems, the objective of semi-supervised learning is to develop a model using a set of training instances $X = \{x_1, x_2, \dots, x_l, x_{l+1}, \dots, x_n\}$. What makes semi-supervised learning unique is that the set of labels $Y = \{y_1, y_2, \dots, y_l\}$ is only available for a portion of the training instances. In segmentation of medical imaging, semi-supervised learning algorithms may be useful in a situation in which a dataset consists of N MRI scans, but segmentation maps are only available for the first L scans, where $L < N$. Semi-supervised learning algorithms leverage the unlabeled instances to achieve improved generalization. One approach to semi-supervised learning exploits unlabeled data by updating a model's parameters based on predictions made by a stronger model. This framework is referred to as the *student-teacher approach* (Laine & Aila, 2017; Tarvainen & Valpola, 2017; Perone & Cohen-Adad, 2018). Another semi-supervised framework, known as virtual adversarial training (VAT), applies an auxiliary loss function that evaluates the divergence between predictions based on an original instance and a virtual adversarial perturbed instance (Szegedy et al., 2014; Miyato et al., 2018). This approach forces a model to be robust to small perturbations to input data and can be applied with unlabeled data.

The objectives of this chapter are to develop convolutional neural networks for automatic segmentation of medical imaging using semi-supervised learning algorithms and to assess the performance gains of these methods compared to a fully-supervised baseline. The models are validated using an in-house dataset of MRI scans of the knee, as well as a publicly available unlabeled dataset (Nevitt et al., 2006).

3.2 Deep Learning Concepts for Automatic Segmentation

This section reviews CNNs for segmentation of the knee and two semi-supervised learning approaches, called mean teachers (MT) and virtual adversarial training (VAT). Semi-supervised learning is well-suited for segmentation applications given the difficulty of obtaining large volumes of labeled data.

Automatic Segmentation of the Knee from MRI

Use of CNNs for automatic segmentation of the anatomical structures of the knee has been previously explored. CNNs make use of 2D convolutions (Noh et al., 2015; Badrinarayanan et al., 2017; Shelhamer et al., 2017) or 3D convolutions (Çiçek et al., 2016; Milletari et al., 2016). 2D CNNs take as input single MRI slices (sometimes with multiple MRI weightings at once), whereas 3D CNNs are applied using 3D patches of an MRI volume. A third approach uses 2D slices from multiple views. This approach, known as 2.5D, leverages context in all three anatomical planes but may not be as computationally expensive as 3D CNNs. The ability to use inter-slice context for informing predictions improves performance for segmentation (Milletari et al., 2016).

CNNs were trained for segmentation of the knee and evaluated on two different datasets by Raj et al. (2018). The proposed architecture exercised extensive use of skip connections (He et al., 2016a, 2016b; Huang et al., 2017) and deep supervision (Lee et al., 2015). The model was trained and evaluated on the SKI10 dataset (Heimann et al., 2010) as well as on Osteoarthritis Initiative (OAI) data (Nevitt et al., 2006). The CNNs trained on these datasets only classified cartilage and menisci in each scan. The SKI10 dataset also includes annotations for femur and tibia but these annotations appear not to have been utilized in the study. Additionally, segmentation was applied after down sampling all scans to coarser voxel dimensions, which may reduce the accuracy of the output mask to below what is necessary for most orthopedic applications.

Bone and corresponding cartilage were initially combined into one class for segmentation using 2D CNNs by Lee et al. (2018). This approach identified structures of both bone and cartilage, as well as a structure of only bone, with the difference between these segmented structures representing predicted cartilage. Final predictions for an entire scan were taken from an ensemble of predictions from each anatomical plane. The models were trained and evaluated using the SKI10 dataset, which provides 60 training scans and 40 validation scans for segmentation of the femur, tibia, and femoral and tibial cartilage.

A multiview approach was developed by Prasoon et al. (2013) by using a triplanar CNN for segmentation of tibial cartilage. Their CNN took as input voxels in all three anatomical planes to predict the class of the voxel that is intersected by each plane. Extracted features from each plane were vectorized and concatenated for a fully-connected layer before final classification of the target voxel. This approach efficiently leveraged context from each anatomical plane while avoiding the computational expense associated with 3D CNNs.

Mean Teachers

The student-teacher approach is predicated on the intuition that model ensembles produce more accurate predictions than those by a single model. The goal of this approach is to train a model by jointly forcing predictions to be closer to provided ground truths (if available) and also closer to predictions made by the teacher model. As such, the student-teacher training framework can be described by two different aspects: conventional loss and consistency loss (Fig. 3.1). Conventional loss is the supervised loss applied to the labeled data regime, such as cross-entropy for classification or dice loss for segmentation. This loss can only be applied to labeled instances. Separately, the consistency loss is a cost that is computed based on two different predictions of a training instance; one from the student model and one from the teacher model. A training instance is fed to the student model and to the teacher model separately under distinct applications of noise. The objective of the student-teacher approach is to train the weights of the weaker student based on how its predictions differ from the teacher's predictions. In this way, the teacher model's predictions can be thought of as "pseudo-ground truths". The consistency loss is applied to both labeled and unlabeled training instances. Following Tarvainen & Valpola (2017), the consistency loss for an instance is defined as:

$$J_{C,i}(\theta) = \left|\left|f(x_i, \theta', \eta') - f(x_i, \theta, \eta)\right|\right|^2 \tag{3.1}$$

Where $f(x_i, \theta', \eta')$ is the prediction resulting from applying the teacher model with parameters θ' to a training instance x_i with applied noise η', and $f(x_i, \theta, \eta)$ is the student model's prediction for x_i under different noise. The application of noise allows the model to learn a function that is smooth in the space around the input. It is assumed that the ground truth distribution is invariant to the noise applied. When using the student-teacher approach, two separate loss functions can be defined for a labeled instance $L_{\iota,i}(\theta)$ and unlabelled instance $L_{\mu,j}(\theta)$.

$$L_{\iota,i}(\theta) = L(\, f(x_i, \theta, \eta), y_i \,) + \lambda J_{C,i}(\theta) \tag{3.2}$$

$$L_{\mu,j}(\theta) = \lambda J_{C,j}(\theta) \tag{3.3}$$

Where y_i is the ground truth corresponding to labeled training instance x_i, $L(f(x_i, \theta, \eta), y_i)$ is some conventional supervised loss function computed between ground truth and student model's prediction, and λ is a constant that controls the contribution of the consistency loss.

The teacher model has been presented in different forms. Two different dropout instances (Sutskever et al., 2014) differentiate the student and teacher predictions in the approach of Laine & Aila (2017), where dropout is applied under training conditions. This same work (Laine & Aila, 2017) also proposes to store the predictions of the training instance, taken at different points during training, and average over these predictions to obtain the teacher's prediction. Exponential moving averages of model weights during taken training are the teacher model proposed by Tarvainen & Valpola (2017), a framework called mean teachers (MT). These approaches are proposed in the context of image classification, in which the true class of an image is invariant to input noise.

Separately, Perone & Cohen-Adad (2018) propose to use MT for automatic segmentation. In segmentation, the ground truth of a training instance is not invariant to geometric transformations (translation, rotation, shear, etc.). Therefore, in order to preserve the validity of the consistency loss, any augmentation in the form of a geometric transformation applied to a training instance that is fed to the student model must also be applied to the teacher model's prediction.

Virtual Adversarial Training

It was found by Szegedy et al. (2014) that neural networks are not robust to adversarial perturbations. Given an input image, in practice the adversarial example is the input perturbed by some noise vector of a constrained magnitude that maximizes the difference between that image's ground truth and the CNN's prediction, x_i+r_{adv}, where r_{adv} is some noise applied to the input (Goodfellow et al., 2015):

$$x_{i,adv} = x_i + r_{adv} \tag{3.4}$$

$$r_{adv} = argmax_r(C(y_i, f(x_i + r, \theta))) \ \mid \ \left|\left|r\right|\right| \leq \epsilon \tag{3.5}$$

Where $C(y_i, f(x_i + r, \theta))$ is a cost function that penalizes the distance between ground truth and adversarial example prediction. The constant ϵ is some constraint on the magnitude of the noise vector. In order to address this problem Goodfellow et al. (2015) proposed to add a supplementary loss function that penalizes the difference between a training example's ground truth and the prediction based on the corresponding adversarial example. This loss was extended by Miyato et al. (2016, 2018) to unlabeled examples by penalizing the difference between a CNN's predictions from both the original training instance and the virtual adversarial perturbed training instance. This approach circumvents the need for ground truth and allows for application in semi-supervised settings. Similar to the student-teacher approach, VAT employs a supplementary loss function in addition to the conventional loss function. However, the virtual adversarial loss originates from a different motivation. This loss function effectively acts as a regularizer that forces a CNN's predictions to be smooth around training instances, specifically in the virtual adversarial direction. The computation of the virtual adversarial loss is straightforward once the virtual adversarial example is obtained. However, the calculation of the virtual adversarial direction is complicated and necessitates estimation (Miyato et al., 2016). In this chapter, the VAT method is applied in tandem to MT for segmentation:

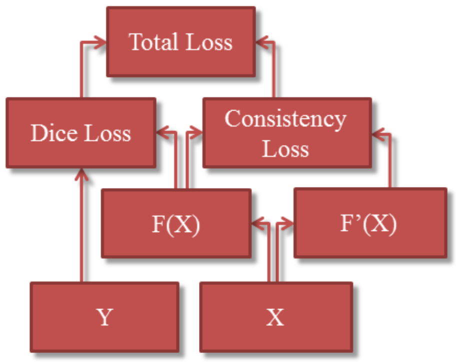
$$J_{VAT,i}(\theta) = \left|\left|(f(p_i,\theta) - f(p_i + r_{vadv},\theta))\right|\right|^2 \tag{3.6}$$

Where the L2 norm is taken over all pixels of the input image. Similar to MT, the loss function for labeled and unlabeled training instances can be defined separately:

$$L_{\iota,i}(\theta) = L(\, f(x_i,\theta), y_i\,) + \alpha J_{VAT,i}(\theta) \tag{3.7}$$

$$L_{\mu,j}(\theta) = \alpha J_{VAT,j}(\theta) \tag{3.8}$$

Where α is a constant that controls the contribution of the virtual adversarial loss.



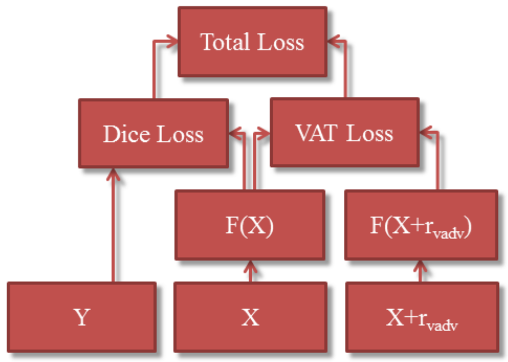


Figure 3.1. An overview of the MT (left) and VAT (right) frameworks

3.3 Semi-Supervised Learning for Automatic Segmentation of the Knee from MRI with Convolutional Neural Networks

Methods

Semi-supervised learning methods were applied to the problem of automatic segmentation of the knee from MRIs. Given an MRI, the trained CNNs classified each voxel of a scan into 1 of 7 classes. The models were trained and evaluated using a data set that consisted of both labeled and unlabeled training instances. The labeled training set comprised 29 subjects, which were previously used in the development of a statistical shape model of the knee (Smoger et al., 2015). This resulted in 3,864 labeled training images in the sagittal plane. An additional 2 subjects were withheld for validation and 5 subjects were used for evaluation. For the unlabeled data, 25,875 MRI slices from 51 scans were sampled from the OAI dataset (Nevitt et al., 2006). The data consisted of T2 and DESS MRIs with slice thicknesses of 0.7 or 1 mm and pixel dimensions of 0.23-0.46 mm in the slice plane.

2D CNNs were trained using different quantities of labeled data to explore the performance gains obtained from using semi-supervised learning algorithms. MT and VAT were applied separately to train CNNs with labeled cohorts of 500, 1,000, 2,000, and 3,864 slices. For each case, any excluded labeled instance was used in training as an unlabeled instance. All models were trained with an L2 regularization constant of 0.0001 and were implemented using TensorFlow (Abadi et al., 2016).

Each model was trained using the RMS-Prop optimization scheme. A 2D CNN architecture similar to U-Net was used (Fig. 3.2, Tab. 3.1) (Ronneberger et al., 2015). This architecture had an initial depth of 64 filters, and the number of filters was multiplied (divided) by 2 at each down sampling (up sampling). Additionally, two dropout layers were added to the network to be consistent with the dropout feature utilized in student-teacher approaches (Perone & Cohen-Adad, 2018; Tarvainen & Valpola, 2017). The dropout layers had a drop probability of 0.3. Skip connections were instantiated between equivalent scales of the encoder and decoder using concatenation of tensors, as opposed to residual connections.

MRI slice pixel values were thresholded at 2.5 standard deviations above a slice's mean value and then normalized to the range 0 to 255 and mean-centered. Random data augmentation was used during training in the form of contrast adjustment, shear, horizontal flipping, and pixel size resampling. The training cohort consisted of MRI slices with only a subset of pixel sizes. In order to make the models robust to a range of pixel dimensions, the data augmentation pipeline uniformly sampled from the range 0.19-0.45 mm, and then rescaled the input image, ground truth (if available), and teacher model prediction based on the ratio between the input image's current pixel size and the desired sampled size to obtain an augmented pixel spacing of the MRI slice. The CNNs were trained using random image crops of 368 x 368 pixels (which usually represented around 80% of the slice original area) and batches with 2 supervised instances and 2 unsupervised instances for semi-supervised training.

The conventional loss of choice was the weighted general dice loss similar to that presented by Sudre et al. (2017).

$$L(y'_i , y_i) = -2 \frac{(\sum_{c=1}^{C} w_{i,c} \sum_{n=1}^{N} (y'_{i,cn} \times y_{i,cn}))^2}{(\sum_{c=1}^{C} w_{i,c} \sum_{n=1}^{N} (y'_{i,cn} + y_{i,cn}))^2} \tag{3.9}$$

$$w_{i,c} = \frac{1}{(\sum_{n=1}^{N} y'_{i,cn})^2} \tag{3.10}$$

Where C is the pixel class, N sums over all pixels in training instance i, y_i is the one-hot encoded ground truth of the segmentation map, and y'_i is the softmax probability obtained by the student model. The weight $w_{i,c}$ increases loss for pixels with low frequency. It was found that the use of this weighting was important for segmentation given the infrequency of some classes. Without the class-wise weights, the model learned to ignore entire classes, even with aggressively small learning rates.

The mean-squared error over all pixels between the student model's and the teacher model's softmax probabilities were used for the MT consistency loss. Similar to Tarvainen & Valpola (2017), the value of λ was ramped up from 0 to 1 according to $\lambda = e^{-5(1-t/50000)^2}$ where t is the training step. The models were trained for 100,000 iterations.

2D CNNs were also trained using VAT similar to the methods of Miyato et al. (2016). L2 loss was implemented for the virtual adversarial loss instead of the original use of KL-Divergence. Models were trained for 100,000 iterations and a constant value ϵ = 5. The value of α from 0 to 1 was ramped up similar to the λ parameter in MT.

These 2D CNNs were trained to predict a segmentation map given a single MRI slice in the sagittal plane. One limitation of this approach is that voxel (e.g. pixel in three-dimensional space) classification predictions were obtained based on a single 2D image. This approach fails to leverage context in the transverse and coronal anatomical planes. A triplanar ensemble was developed to address this problem (Fig. 3.3) (Prasoon et al., 2013). Three separate CNNs with the same architecture as Table 2.1 were trained with MT using the full amount of data to segment scans in the sagittal, transverse, and coronal planes. The entire image of a slice was kept as input for each of the triplanar models instead of taking random crops.

3D CNNs were also trained for the automatic segmentation of the knee from MRI (Figure 3.4). Given a volumetric input patch of $x^{(i)} \in \mathbb{R}^{h \times w \times d \times 1}$, the 3D CNNs were trained to predict corresponding segmentation maps of $y^{(i)} \in \mathbb{R}^{h \times w \times d \times 7}$. The architecture used for the 2D CNNs (Table 3.1) was extended to the 3D scenario by replacing 2D convolutional filters with 3D filters. Also, the concatenation skip connections used for the 2D CNNs were replaced with residual skip connections for computational efficiency. A fully-supervised 3D CNN was trained on the labeled data set and an additional 3D CNN was trained with the MT framework. Each 3D CNN was trained with an initial training stage where training instances were sampled as a sequence of 4 slices and then down sampled in the slice plane dimensions to result in tensors $x^{(i)} \in \mathbb{R}^{96\times 96\times 4\times 1}$. The pretraining on down sampled training instances allowed for accelerated training. This stage lasted for 60,000 iterations. Afterwards, the 3D CNNs were trained for an additional 90,000 iterations with input tensors $x^{(i)} \in \mathbb{R}^{192\times 192\times 4\times 1}$. Here, the tensors were not down sampled versions of the original slices, but were instead random crops that were significantly smaller than the original slice size.

All trained models were evaluated on the 5 test subjects. Inference was performed with the 2D CNNs by obtaining segmentation maps for each slice, one at a time, before applying a connected components algorithm to filter out noisy clusters from the segmentation maps (Fiorio & Gustedt, 1996; Wu et al., 2005).

Predictions with the triplanar ensemble were computed by taking the class-wise probabilities for all voxels in a scan using models trained in each anatomical plane, and then averaging the predictions across the scan. Specifically, the final predicted segmentation map was a 4-dimensional tensor represented by $P_{final} \in \mathbb{R}^{h\times w\times d\times 7}$. The final prediction from a scan was obtained using:

$$P_{final} = \frac{1}{3} \times (P_{sagittal} + P_{coronal} + P_{transverse}) \tag{3.13}$$

Inference with the 3D CNNS was performed differently from the 2D CNNs. The 2D CNNs used an entire MRI slice as input, but the 3D CNNs sacrificed voxels in the slice plane for voxels in the depth dimension. As a result, it was hypothesized that voxel prediction accuracy was dependent on the location of the voxel relative to a sampled patch. That is, it was thought that voxels in the center of a patch enjoyed better accuracy because the 3D CNN had access to more context around voxels in the patch center. This perceived issue was addressed using a novel “Monte Carlo patch sampling” algorithm. First, patches were iteratively sampled from a test scan and predictions were obtained so that every voxel had an initial prediction associated with it. Then, patches were sampled by drawing from uniform distributions over the height, width, and depth of a test scan to produce a new sampled patch. Voxel predictions at a sampled patch were ensembled by taking the mean over all voxel predictions from any time that voxel had been included in a sampled patch. The mean was applied in a computationally efficient way to save on memory. That is, instead of storing predictions from every Monte Carlo iteration and then averaging at the end, voxel predictions were updated based on:

$$p^{(t+1)} = p^{(t)} + \frac{1}{t}(p^{(t+1)} - p^{(t)}) \tag{3.14}$$

Where the prediction at voxel p is being updated for the t^{th} time and the value of t is kept separately for all different voxels in a scan. Final predictions for the 3D CNNs were taken as the ensembled predictions after 2,000 Monte Carlo iterations.

Segmentation performance was evaluated using Intersection-over-Union (IoU) and Dice Similarity Coefficient (DSC) over each volume defined by:

$$IoU(X,Y) = \frac{|X \cap Y|}{|X \cup Y|} \tag{3.15}$$

$$DSC(X,Y) = \frac{2 \times |X \cap Y|}{|X| + |Y|} \tag{3.16}$$

The geometries of the anatomical structures of the knee were reconstructed using the predicted segmentation maps of the best-performing models to assess geometric quality. Specifically, the reconstructed geometries from predicted and ground truth segmentation maps were compared by representing each segmented voxel as a node in 3-dimensional space and using a k-nearest neighbors search to find the closest node on the manual geometry's surface for each predicted geometry surface node. This nodal error yielded intuition about the quality of the predicted geometries that is not provided by standard segmentation metrics such as IoU.

The geometries of all 5 test subjects as predicted by the best model were developed into an FE model to assess the feasibility of using the predicted geometries as part of a more complete workflow in biomechanics research methods. The bones were converted to a triangular surface mesh and the cartilage geometries were developed into meshes using the approach of Rodriguez-Vila et al. (2017). The resulting geometries were subjected to boundary conditions with Abaqus to check for FE-readiness.

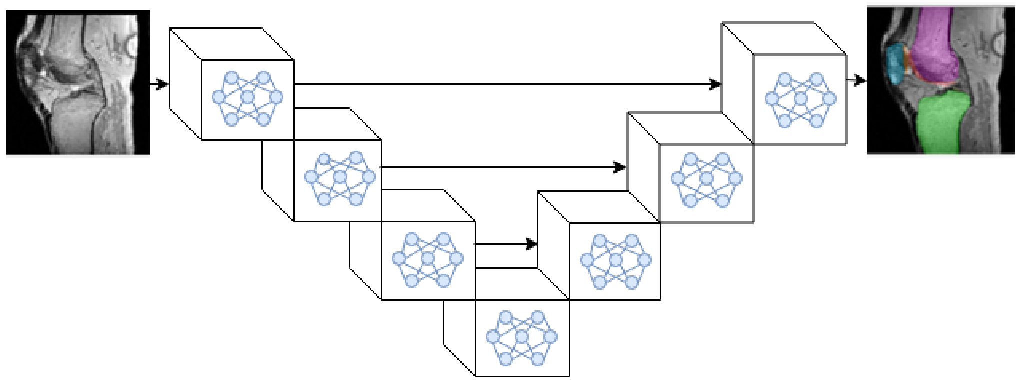


Figure 3.2. A pictorial representation of the U-Net Architecture. In the encoder, convolutional layers are applied and then down sampled. In the decoder, convolutional layers are applied, up sampled, and fused with feature maps from the encoder.

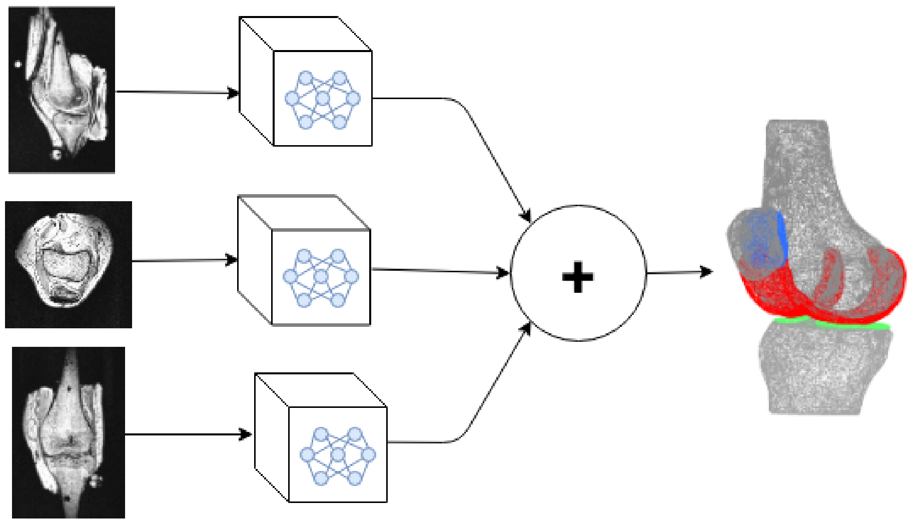


Figure 3.3. A pictorial representation of the triplanar ensemble. Images are used to predict segmentation maps in all three anatomical planes using distinct models. The predictions over the entire volume are fused to rebuild the full predicted geometries.

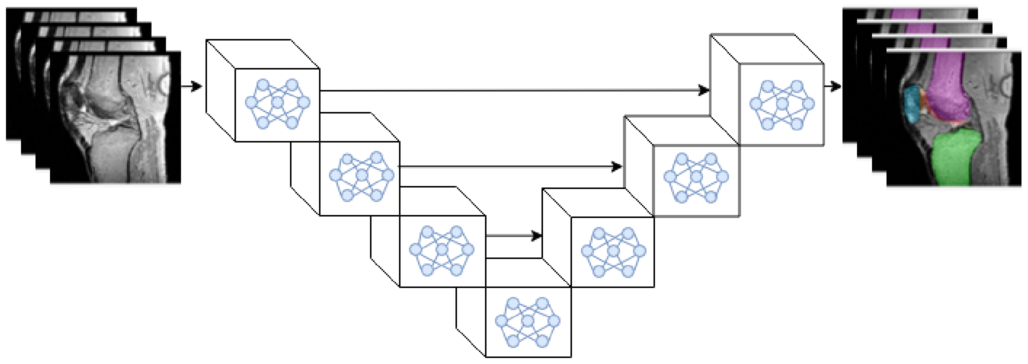


Figure 3.4. The 3D CNN differed from the 2D CNN in that 3D convolutional layers were applied to a volumetric patch of an MRI scan. The 3D CNN inherently leveraged inter-slice context as a result.

Table 3.1. A description of the CNN architecture used for automatic segmentation. Convolutional layers are accompanied by instance normalization (Huang & Belongie, 2017) and parametrized ReLU (PReLUs) (He et al., 2015) activation functions.

Layer	**Number of Filters**	**Features**
Conv 1a	64	
Conv 1b	64	Followed by max pool
Conv 2a	128	
Conv 2b	128	Followed by max pool
Conv 3a	256	
Conv 3b	256	Followed by max pool
Conv 4a	512	
Conv 4b	512	Followed by dropout layer; max pool
Conv 5a	1024	
Conv 5b	1024	
Conv 5c	1024	Followed by dropout layer
Upconv 1	512	Followed by fusion with Conv 4b
Conv 6a	512	
Conv 6b	512	
Upconv 2	256	Followed by fusion with Conv 3b
Conv 7a	256	
Conv 7b	256	
Upconv 3	128	Followed by fusion with Conv 2b
Conv 8a	128	
Conv 8b	128	
Upconv 4	64	Followed by fusion with Conv 1b
Conv 9a	64	
Conv 9b	64	
Conv 10	7	Followed by softmax function; no activation function

Results

The trained models were evaluated on 5 test subjects. Representative predicted segmentation maps are presented for the 2D semi-supervised models (Fig. 3.5). Mean IoU for the fully-supervised baseline was 0.948 (Tab. 3.2). The best-performing 2D MT and VAT models yielded IoU values of 0.964 and 0.967. Class-wise IoU as well as overall and class-wise DSC (Tab. 3.3) are also presented for all models. The performance of the MT and VAT models surpassed that of the fully-supervised model with only 1,000 labeled examples. The 2D fully-supervised baseline yielded an IoU of 0.948, compared to IoU values of 0.952 and 0.954 from the 2D MT and VAT models trained with only 1,000 labeled instances. This is roughly a fourth of the full amount of labeled data used to train the fully-supervised baseline.

The triplanar ensemble and 3D CNNs performed better than the models trained only in the sagittal plane. The triplanar ensemble returned an IoU value of 0.976 on the test set. This is the best performance of all models that were explored, including the 3D CNNs, which yield IoU's of 0.971 and .970 for the full-supervised and MT-trained models. The 3D CNN results are presented by ensembling the predictions of 2,000 Monte Carlo patch samples. IoU performance on the test set was shown to improve with the number of Monte Carlo samples (Fig. 3.16).

Geometries were reconstructed using predicted segmentation maps from the fully-supervised, MT, and VAT models as well as the triplanar ensemble and MT-3D CNN for comparison with geometries reconstructed from ground truth segmentation maps (Fig. 3.7-16). The median, mean, and standard deviation of nodal error between predicted and ground truth geometries are tabulated in Tables 3.4 and 3.5. Median nodal error ranged from 0.36 to 0.98 mm across all classes and models. Mean nodal error (Table 3.5) was higher for the femur and tibia than for the other structures of interest. No trend is evident for median error across models and structures. However, the models that leveraged inter-slice context yielded lower mean error for the femur and tibia than the standalone 2D CNNs. This demonstrates the advantage of using 3D CNNs or using 2D CNNs in multiple planes.

The FE models developed from the predicted geometries of all 5 test subjects were successfully subjected to boundary conditions. The meshes for one of the test subjects is illustrated in Figure 3.17.

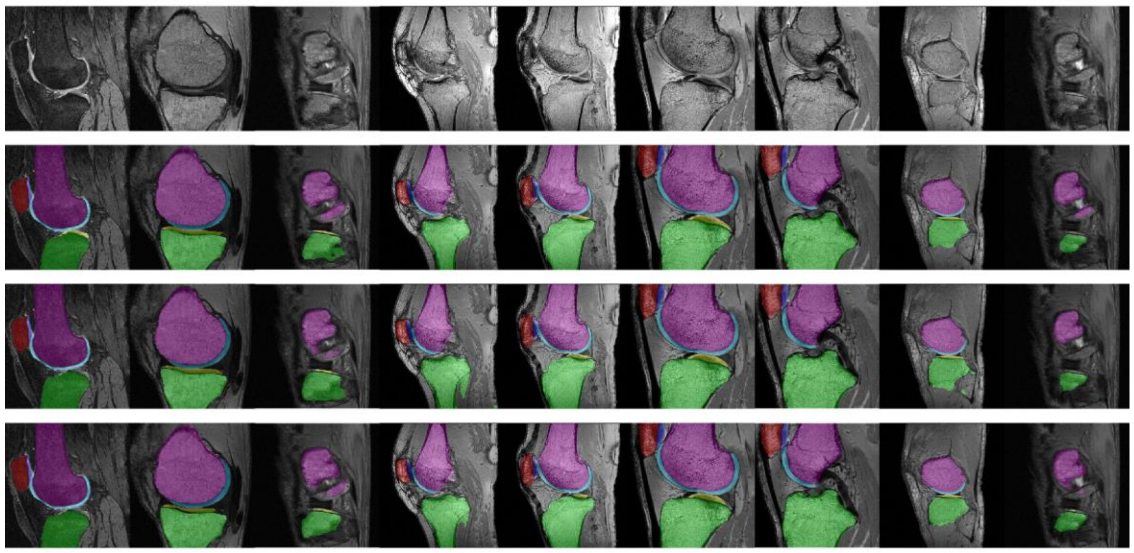


Figure 3.5. Predicted segmentation maps from the top fully-supervised (2nd row), MT (3rd), and VAT (4th) 2D CNNs with corresponding input images.

Table 3.2. IoU results for all models trained.

Model	Overall	Background	Femur	Femoral Cartilage	Patella	Patellar Cartilage	Tibia	Tibial Cartilage
Fully-Supervised	0.948	0.969	0.888	0.670	0.814	0.649	0.888	0.624
MT (500 labeled)	0.936	0.965	0.850	0.399	0.628	0.196	0.836	0.423
MT (1000 labeled)	0.952	0.973	0.893	0.516	0.826	0.545	0.892	0.508
MT (2000 labeled)	0.959	0.978	0.904	0.530	0.824	0.565	0.906	0.544
MT (3864 labeled)	0.964	0.980	0.927	0.676	0.831	**0.650**	0.896	0.659
VAT (500 labeled)	0.908	0.952	0.808	0.401	0.474	0.183	0.627	0.436
VAT (1000 labeled)	0.954	0.975	0.881	0.502	0.819	0.426	0.895	0.568
VAT (2000 labeled)	0.950	0.972	0.894	0.539	0.804	0.538	0.844	0.558
VAT (3864 labeled)	0.967	0.981	0.931	**0.690**	0.839	0.630	0.913	0.541
Sagittal MT	0.973	0985	0.950	0.676	0.831	0.592	0.936	0.623
Coronal MT	0.969	0.983	0.946	0.639	0.801	0.436	0.926	0.628
Transverse MT	0.971	0.984	0.946	0.625	0.824	0.576	0.935	0.622
Triplanar Ensemble	**0.976**	**0.986**	**0.955**	0.688	**0.855**	0.572	**0.943**	**0.675**
Fully-Supervised 3D CNN	0.971	0.983	0.947	**0.690**	0.804	0.571	0.927	0.660
MT 3D CNN	0.970	0.983	0.939	0.660	0.789	0.567	0.932	0.640

Table 3.3. DSC results for all models trained.

Model	Overall	Background	Femur	Femoral Cartilage	Patella	Patellar Cartilage	Tibia	Tibial Cartilage
Fully-Supervised	0.973	0.984	0.930	0.792	0.891	0.774	0.933	0.752
MT (500 labeled)	0.967	0.982	0.919	0.556	0.720	0.280	0.909	0.569
MT (1000 labeled)	0.975	0.986	0.943	0.670	0.904	0.702	0.941	0.656
MT (2000 labeled)	0.979	0.989	0.949	0.685	0.903	0.718	0.950	0.690
MT (3864 labeled)	0.981	0.990	0.962	0.805	0.907	0.783	0.944	0.789
VAT (500 labeled)	0.951	0.975	0.893	0.558	0.590	0.275	0.761	0.595
VAT (1000 labeled)	0.976	0.988	0.937	0.656	0.900	0.596	0.944	0.714
VAT (2000 labeled)	0.974	0.986	0.944	0.694	0.891	0.695	0.913	0.702
VAT (3864 labeled)	0.983	0.990	0.964	0.815	0.912	**0.771**	0.955	0.699
Sagittal MT	0.986	0.992	0.974	0.805	0.907	0.740	0.966	0.758
Coronal MT	0.985	0.991	0.972	0.779	0.889	0.578	0.962	0.627
Transverse MT	0.985	0.992	0.972	0.769	0.902	0.726	0.966	0.765
Triplanar Ensemble	**0.987**	**0.993**	**0.977**	0.814	**0.915**	0.720	**0.971**	**0.790**
Fully-Supervised 3D CNN	0.985	0.992	0.973	**0.816**	0.889	0.710	0.962	0.794
MT 3D CNN	0.984	0.991	0.969	0.793	0.886	0.708	0.964	0.779

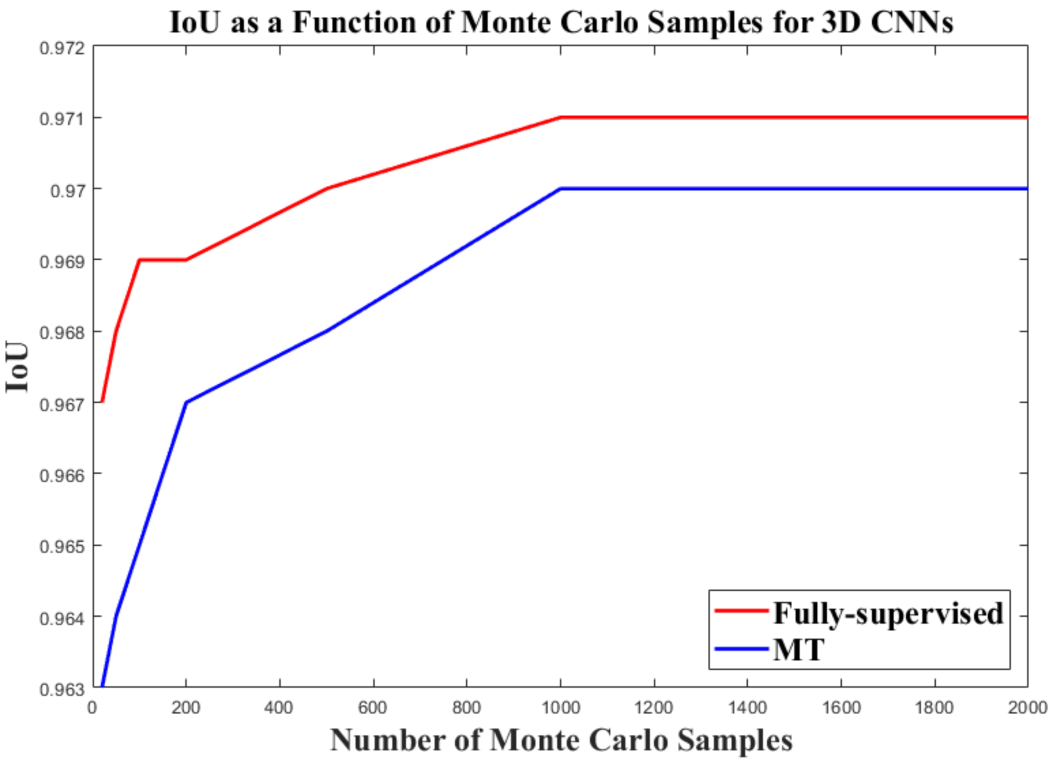


Figure 3.6. Test set IoU increased with the number of Monte Carlo patch samples used during inference and then leveled out.

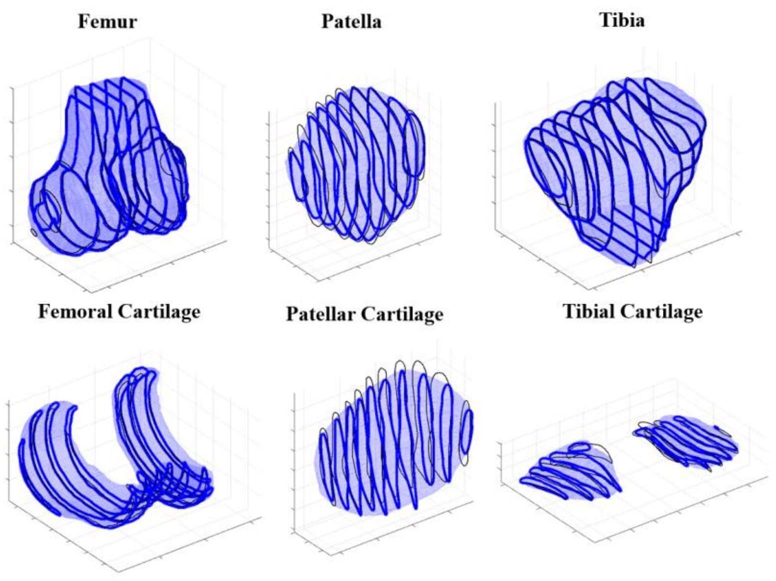


Figure 3.7. Reconstructed geometries from the 2D supervised model's predictions (blue) compared with contours of ground truth geometries (black).

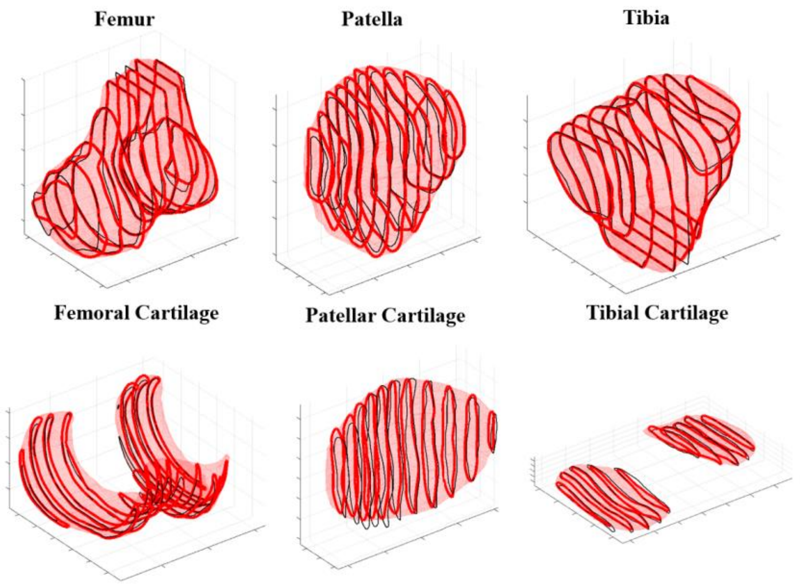


Figure 3.8. Reconstructed geometries from the 2D MT model's predictions (red) compared with contours of ground truth geometries (black).

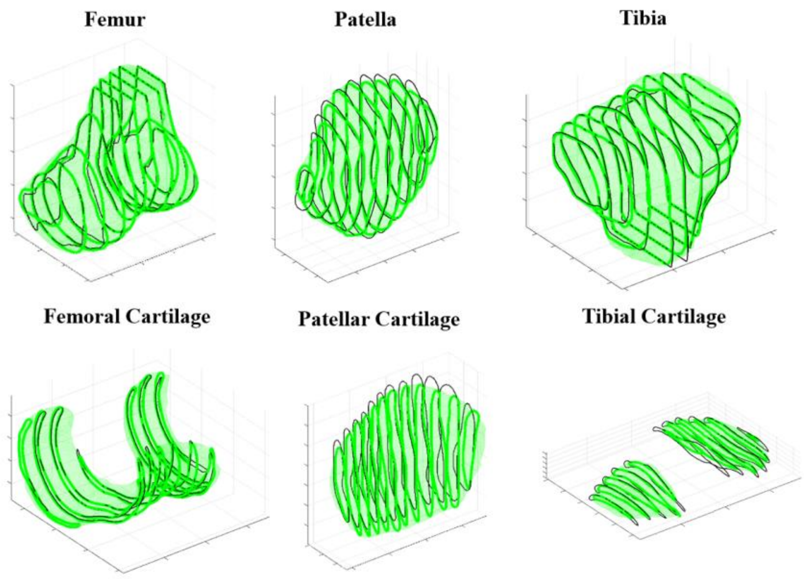


Figure 3.9. Reconstructed geometries from the 2D VAT model's predictions (green) compared with contours of ground truth geometries (black).

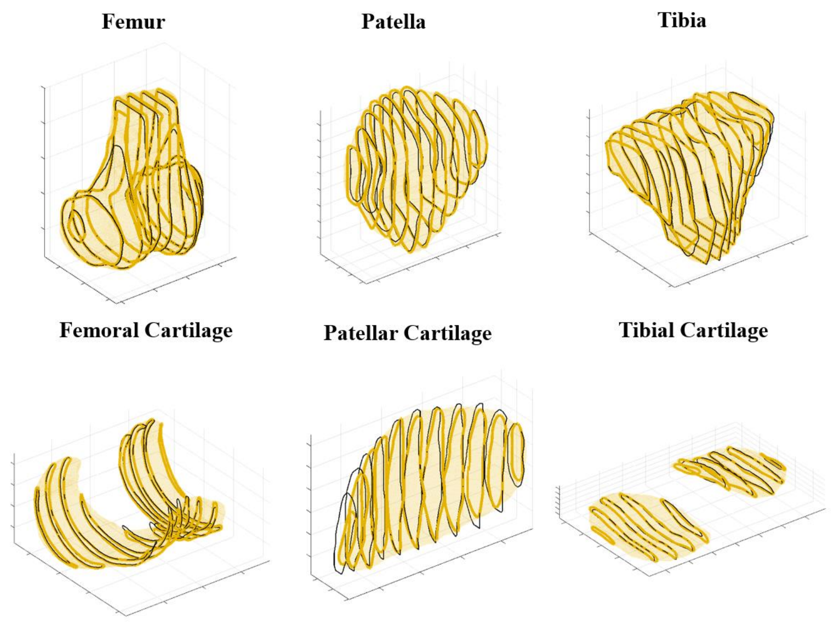


Figure 3.10. Reconstructed geometries from the triplanar ensemble’s predictions (gold) compared with contours of ground truth geometries (black).

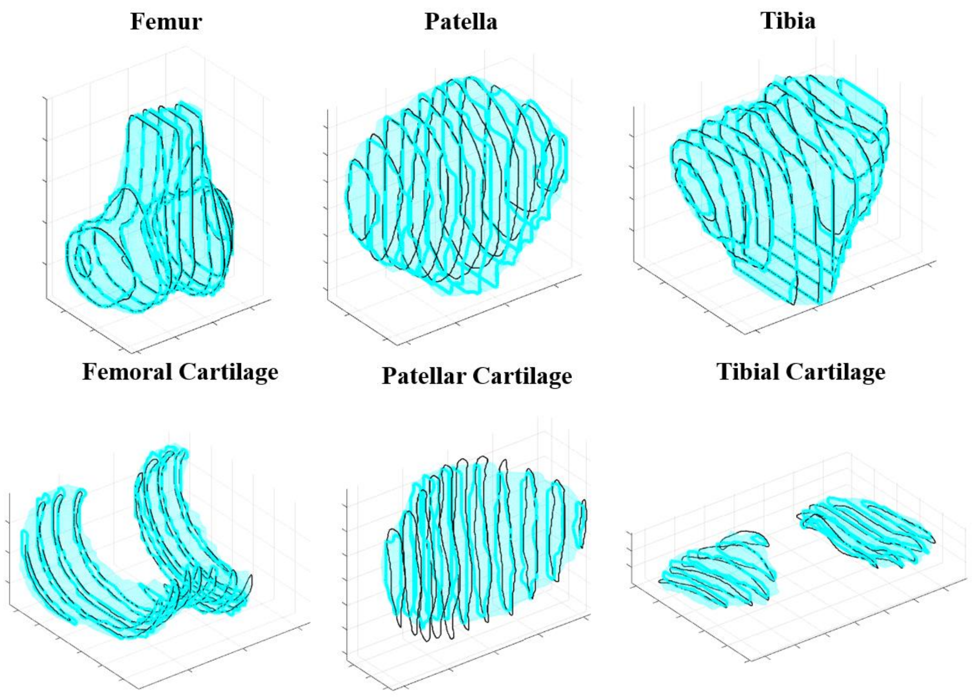


Figure 3.11. Reconstructed geometries from the MT-trained 3D CNN predictions (cyan) compared with contours of ground truth geometries (black).

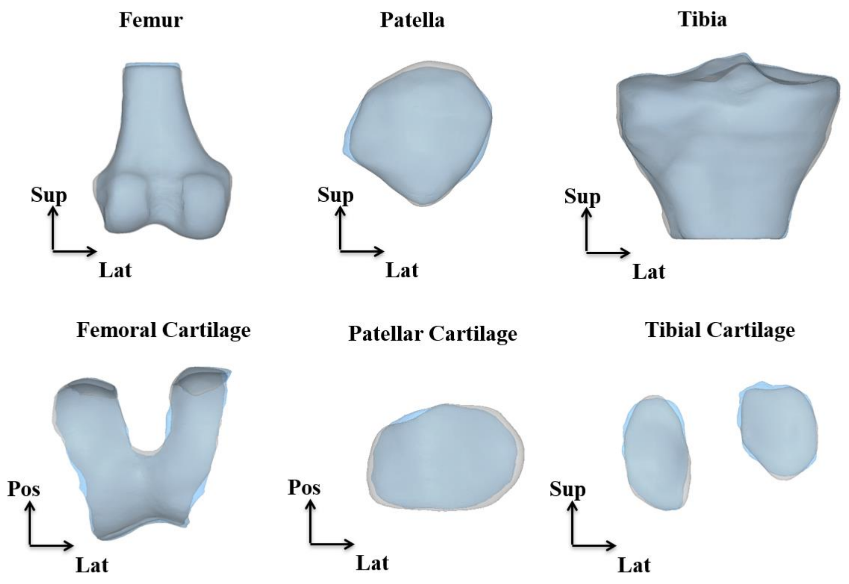


Figure 3.12. Reconstructed geometries from manual segmentations (gray) and predicted segmentations from the 2D fully-supervised model (blue).

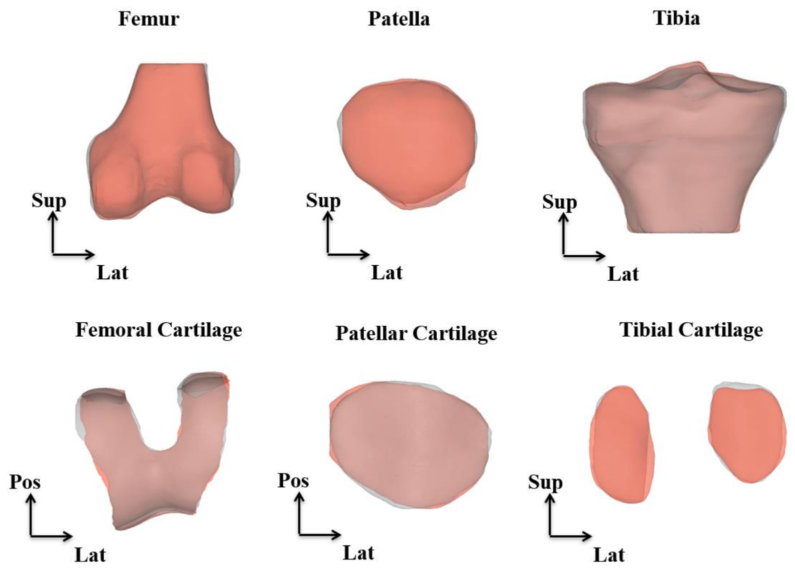


Figure 3.13. Reconstructed geometries from manual segmentations (gray) and predicted segmentations from the 2D model trained with MT (red).

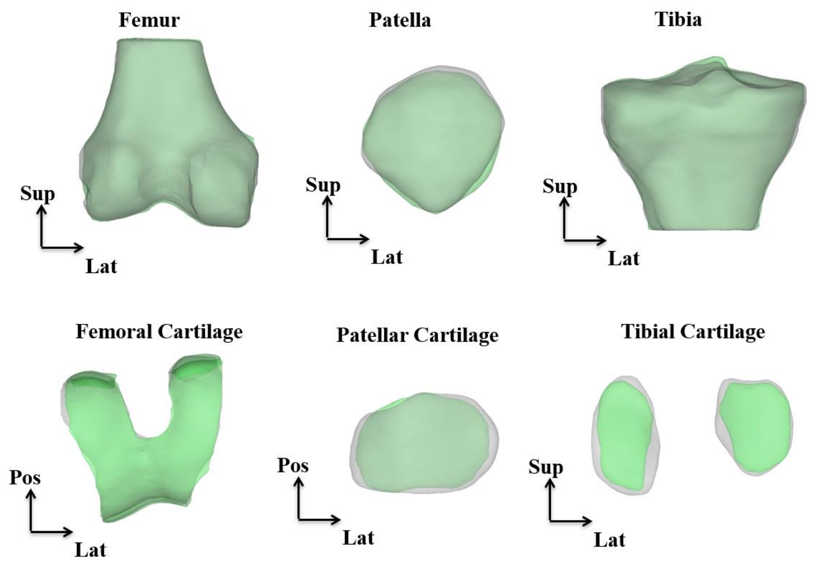


Figure 3.14. Reconstructed geometries from manual segmentations (gray) and predicted segmentations from the 2D model trained with VAT (green).

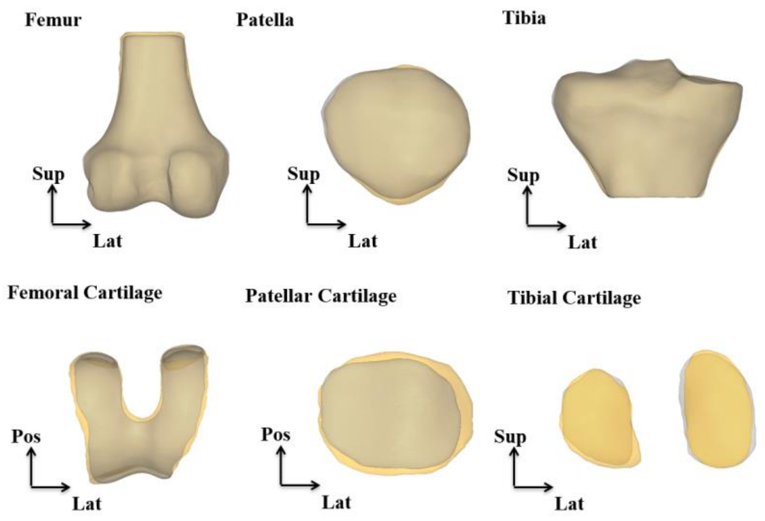


Figure 3.15. Reconstructed geometries from manual segmentations (gray) and predicted segmentations from the triplanar ensemble (gold).

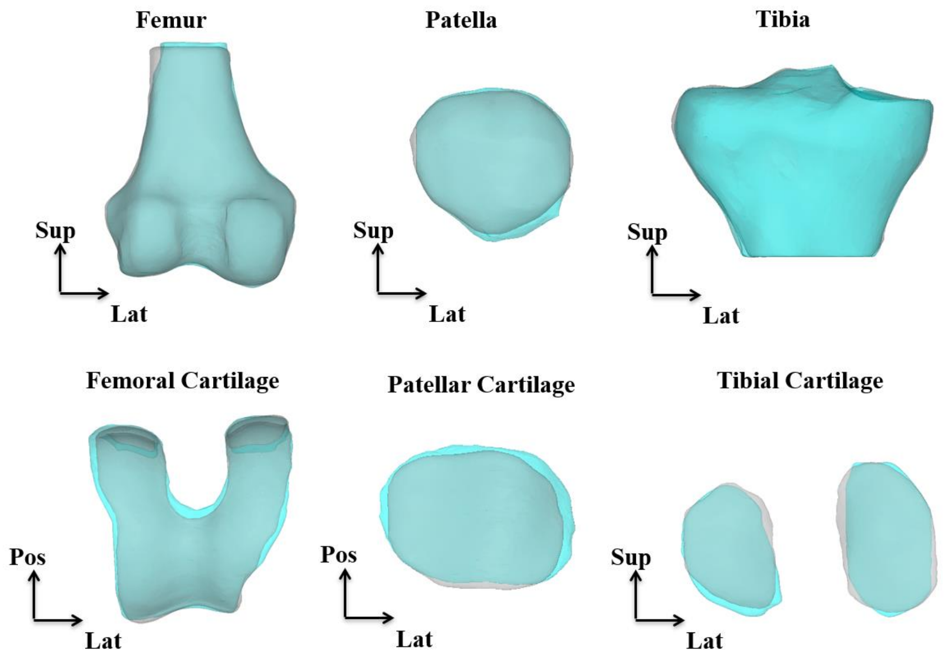


Figure 3.16. Reconstructed geometries from manual segmentations (gray) and predicted segmentations from the MT-trained 3D CNN (cyan).

Table 3.4. Median surface error between geometries from predicted and manual segmentation maps (mm)

Class	Supervised	MT	VAT	Triplanar Ensemble	Supervised 3D	MT 3D
Femur	0.65	0.64	0.59	0.36	0.52	0.60
Femoral Cartilage	0.51	0.59	0.53	0.46	0.64	0.60
Patella	0.58	0.78	0.63	0.58	0.66	0.67
Patellar Cartilage	0.58	0.68	0.65	0.58	0.66	0.64
Tibia	0.66	0.97	0.58	0.41	0.52	0.50
Tibial Cartilage	0.46	0.46	0.48	0.46	0.49	0.51

Table 3.5. Mean ± std surface error between geometries from predicted and manual segmentation maps (mm)

Class	Supervised	MT	VAT	Triplanar Ensemble	Supervised 3D	MT 3D
Femur	1.48 ± 2.50	1.19 ± 2.42	0.82 ± 0.84	0.46 ± 0.46	0.68 ± 0.69	0.75 ± 0.68
Femoral Cartilage	0.94 ± 1.50	0.93 ± 1.34	0.77 ± 0.77	0.61 ± 0.67	0.82 ± 0.72	0.74 ± 0.55
Patella	0.86 ± 0.94	1.01 ± 1.01	0.78 ± 0.60	0.71 ± 0.60	0.90 ± 0.74	0.90 ± 0.73
Patellar Cartilage	0.69 ± 0.53	0.81 ± 0.57	0.76 ± 0.54	0.76 ± 0.71	0.82 ± 0.63	0.80 ± 0.62
Tibia	1.41 ± 2.17	1.31 ± 1.23	1.32 ± 2.4	0.50 ± 0.56	0.66 ± 0.68	0.63 ± 0.61
Tibial Cartilage	0.53 ± 0.43	0.60 ± 0.48	0.6 ± 0.45	0.55 ± 0.53	0.70 ± 0.69	0.67 ± 0.59

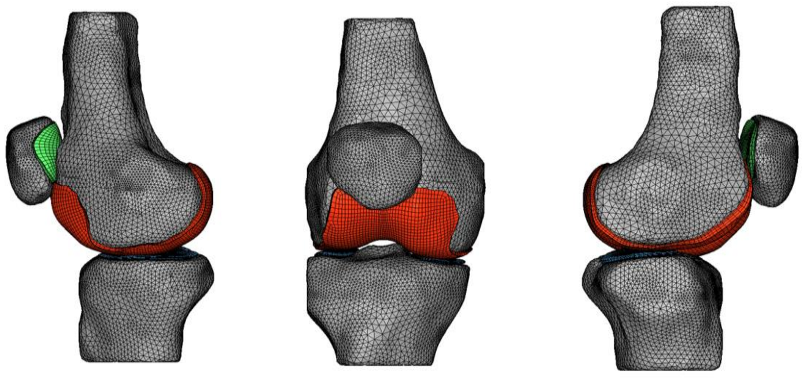


Figure 3.17. Automatically meshed FE model from predicted segmentation maps.

Discussion

The objectives of this chapter were to explore the performance gains obtained from using semi-supervised learning methods and to develop robust CNNs for automatic segmentation of the knee from MRIs with these frameworks. The 2D semi-supervised frameworks not only performed better than the 2D fully-supervised baseline, but they also exceeded the performance of the fully-supervised baseline with only a fourth of the labeled data. This finding yields intuition about the potential of semi-supervised learning for automatic segmentation in scenarios with small quantities of labeled data. The experiments in this chapter leveraged 25,000 unlabeled MRI slices in the sagittal plane. However, in practice it would be preferable to increase the amount of unlabeled data used, especially given the sheer size of publicly available datasets (Nevitt et al., 2006).

Automatic segmentation of the knee has previously been explored in literature. 3D CNNs were developed by Raj et al. (2018) to explore segmentation of ligaments and cartilage on two different datasets. They achieved DSC metrics of the femoral, tibial, and patellar cartilage of 0.849, 0.832, and 0.785 on scans from the OAI dataset. This contrasts with the current study’s best cartilage DSC values of 0.816, 0.794, and. 0.783. These differences can be explained by the use of less than half of the amount of labeled training data as compared to the referenced study. Additionally, the referenced study discusses the use of 2 scans per patient in the total data set (taken at different time points) and mentions the possibility of the 2 scans being split between training and testing sets during their cross validation. It is believed that this would lead to a significant boost in performance on the test set. In contrast, the subjects of the dataset used in this chapter are distinct between training, validation, and testing subjects. The triplanar ensembles of Lee et al. (2018) were trained on 60 labeled scans and resulted in DSC values of 0.973, 0.844, 0.981, and 0.838 for femur, femoral cartilage, tibia, and tibial cartilage. Only 29 labeled subjects are used to train the models in this study while still obtaining corresponding

DSC values 0.977, 0.816, 0.971, and 0.794. Although direct comparisons are not feasible between the distinct datasets, it is clear that using semi-supervised learning with 2D CNNs leads to performance competitive with larger studies.

The CNN-predicted geometries were evaluated for FE-readiness by developing FE models from the predicted geometries. The resulting FE models were subjected to boundary conditions and FE studies were successfully completed. This demonstrates the geometric quality of the predicted geometries and has implications for the potential of the CNNs to be incorporated into traditional biomechanics research methods.

The models trained in the sagittal plane inherently failed to leverage interslice context. Multiple methods have been proposed to address this fundamental issue (Prasoon et al., 2013, Milletari et al., 2016). Triplanar ensembles and 3D CNNs were explored in this chapter to address this. The 3D CNNs were sufficiently expensive that both training and inference were performed on a small 3D patch of the volume of interest (or a significantly down sampled version of the full volume) while still necessitating the use of 2 GPUs with 11 GB of memory each. Additionally, the batch size was limited and a single forward pass of the CNN is slower as well, leading to slower training and slower inference. Contrastingly, the triplanar ensemble was able to take as input entire slices for a single forward pass while still leveraging interslice context.

The triplanar ensemble method used in this chapter fused predictions from each anatomical plane with a simple voxel-wise average. This approach failed to consider that some views may provide stronger predictions for different locations in the 3D volume. For instance, the coronal or transverse plane may be more effective for predicting voxel-wise classes at the medial and lateral borders of the femoral cartilage. Ideally, when predicting segmentation maps for this area in the volume, it may be preferable to weight the sagittal view prediction lower than the other two views. Although this idea was not explored here, it was implemented by Wang et al. (2019) by using a form of the expectation-maximization (EM) algorithm. Future research directions for automatic segmentation will explore this method.

The triplanar ensemble and 3D CNN predictions yielded lower mean surface error than the single-plane models on the femur and tibia. The surface error is ultimately the metric that is more important from the perspective of biomechanics research.

The CNNs in this chapter were trained on a relatively small number of MRIs. Additionally, the training set constituted a limited number of MRI sequences; mostly DESS. In practice, these CNNs would not be robust to different MRI sequences without being trained on the sequence of interest in some capacity.

Training segmentation models from scratch (e.g. without transfer learning (Tan et al., 2018)) is difficult. Specifically, the biggest issue encountered during training was the tendency for a CNN to learn to ignore entire classes during training. Once a CNN ceased to predict a specific class altogether, it rarely recovered from this local minimum. Listed below are some strategies that seemed to be effective in avoiding this issue:

1. Reduce the learning rate – The CNNs in this chapter were trained with learning rates from 1e-5 to 1e-8 when using the RMS Prop Optimizer.

2. Increase batch size – It was observed that increasing batch size helped to avoid the class-ignoring issue. This is presumably because as the batch size increases, the probability of all classes occurring in the batch also increases. If computational constraints limit the batch size, simply down sample the resolutions of the training instances to increase batch size initially. Then reduce batch size later and return the training instances to their preferred resolution.
3. Strategic training instance sampling – Early in training it may be beneficial to manually choose which training instances to include in the batch. Specifically, ensure that all classes are present in each batch.
4. Annealing the dice loss weight coefficient – The denominator of the dice loss weight (Eq. 3.10) is traditionally squared. It may be useful to tune this exponent. If the CNN ignores classes of small frequency (cartilage), increase this exponent. If the CNN ignores classes of high frequency (femur, tibia), reduce this exponent.
5. Dropout layers – Conventional dropout (as opposed to Monte Carlo dropout (Gal & Ghahramani, 2016)). is more often used in fully-connected layers. However, the use of conventional dropout layers with drop probabilities ranging from 0.4-0.2 seemed to address the class-ignoring problem.

3.4 Conclusion

This chapter evaluated 2D and 3D CNNs trained with fully-supervised and semi-supervised techniques for automatic segmentation of 6 structures of the knee. The semi-supervised methods were able to successfully leverage unlabeled data and achieved performance that is on par with existing literature despite using significantly less labeled data. Triplanar ensembles and 3D CNNs were developed to leverage context from all three anatomical planes for informing predictions. It was found that 3D CNNs yield superior performance to 2D CNNs, which verifies findings in other literature.

Given that reconstructed geometries from predicted segmentation maps are of sufficient quality, it would be interesting to explore how surface error of reconstructed geometries propagates through to results of downstream biomechanics applications. For instance, comparing differences between FE study results obtained from predicted and ground truth (e.g. manually obtained) geometries would serve to further validate the use of CNNs for automatic segmentation as part of more complete workflows in biomechanics research methods.

CHAPTER 4. TOWARD REAL-TIME MUSCULOSKELETAL MODELING FOR ESTIMATING PATIENT MECHANICS USING DEEP LEARNING

4.1 Gait Lab Analysis in Orthopaedics

Gait lab measurement of whole-body kinematic data and ground reaction forces is utilized in clinical settings for patient diagnosis and monitoring; as well as in research settings to obtain metrics needed to drive computational models. These data are commonly processed in musculoskeletal modeling platforms such as OpenSim (Delp et al., 2007) and Anybody (Damsgaard et al., 2006) to estimate muscle forces and joint reaction forces during activity.

Berchuck et al. (1990) used gait lab data to assess kinematic differences between patients with deficient anterior cruciate ligaments and a control population. Myers et al. (2018) used gait lab data to inform musculoskeletal models that were used to simulate the effects of hip implant alignment on muscle and joint loads. Hume et al. (2019) used gait lab data to tune joint loading and muscle forces for FE modeling of the knee. The gait lab workflow is also applied within the context of joint replacement evaluation and results in valuable output metrics for use in implant design and surgical decision making, as well as characterizing overall patient function (Ngai & Wimmer, 2009).

However, the processing required to obtain musculoskeletal modeling estimates can be time consuming, requires expertise and hardware, and thus limits the patient populations studied. The expense of mapping kinematics and ground reaction forces to metrics of interest may benefit from data-driven machine learning techniques.

Deep learning has found applications in predictive modeling of sequence data in which neural networks are used to predict an output given a sequence of input data. This class of problems is called sequence-to-sequence learning. Sutskever et al. (2014) used RNNs to translate sentences between different languages; a process called neural machine translation. Ping et al. (2018) applied deep learning techniques generate speech. Gehring et al. (2017) and Kalchbrenner et al. (2016) developed CNNs for neural machine translation instead of RNNs.

Machine learning techniques provide potential for supplementing or even replacing the tedious workflow for processing joint reaction or muscle forces from standard gate lab data by learning mappings from data. The fast inference time of these algorithms could allow for real-time mapping. Accordingly, the objective of this chapter was to explore the potential of deep learning techniques for learning to map standard gait lab data to joint reaction and muscle forces. This was performed by applying two different neural network architectures to an in-house dataset and evaluating the performance of each on a test set.

4.2 Deep Learning Concepts for Sequence-to-Sequence Modeling

Long Short-Term Memory Cells

Neural networks have been successfully applied to sequence-to-sequence problems. Examples include neural machine translation (Sutskever, et al., 2014) and generating spoken voice (Ping et al., 2018). Traditionally, RNNs are the architecture of choice for this problem class (Graves, 2012; Hochreiter & Schmidhuber, 1997). One known issue with the vanilla RNN unit formulation presented in Chapter 2 is known as the vanishing gradient problem. This problem is characterized by a gradient that is reduced to an insignificant magnitude as it is backpropagated through time steps during training of the RNN. The vanishing gradient problem limits an RNN’s ability to learn dependencies over large time scales. As a result, it is common to use a specialized RNN unit called a long short-term memory cell (LSTM). While there are numerous definitions for the LSTM cell that slightly differ from each other (e.g. with or without peepholes (Sak et al., 2014)) the LSTMs used in this chapter are formulated as follows:

$$i_t = \sigma(W_1^T X_t + W_2^T H_{t-1} + b_1) \tag{4.1}$$

$$f_t = \sigma(W_3^T X_t + W_4^T H_{t-1} + b_2) \tag{4.2}$$

$$C_t = f_t \circ C_{t-1} + i_t \circ \tanh(W_5^T X_t + W_6^T H_{t-1} + b_3) \tag{4.3}$$

$$o_t = \sigma(W_7^T X_t + W_8^T H_{t-1} + b_2 + b_4) \tag{4.4}$$

$$H_t = o_t \circ \tanh(C_t) \tag{4.5}$$

Where X_t is the input at the current time step, H_t is the familiar hidden state, and C_t is called the cell state. Similar to vanilla RNNs, the hidden state is used to make predictions at the current time step and also passed to the future time step to inform future predictions. Additionally, $\circ$ is known as the Hadamard product.

Temporal Convolutional Neural Networks

A disadvantage of RNNs is that they have an inherent linear sequential structure. The computational graph of an RNN is essentially unrolled for t time steps and can only be evaluated one step at a time. The computational speed of evaluating an input sequence becomes an issue as the input sequence length becomes larger. As a result, 1D CNNs have also been applied to sequence-to-sequence modeling (Kalchbrenner et al., 2016), in which a hierarchy of 1D convolutional layers are applied to an input sequence to map it to an output. This format of NN is referred to as a temporal convolutional network (TCN). Given an input sequence $X \in \mathbb{R}^{T\times N}$ with T time steps and N features per time step, a 1D convolutional filter $A \in \mathbb{R}^{M\times N}$ is applied to time step t of the sequence based on:

$$X(t) * A = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} A(i,j) \times X(t+i,j) \tag{4.6}$$

For sequence-to-sequence learning, successive 1D convolutional layers apply convolutions with different dilations between convolutional filter elements along the time dimension. For a dilation rate d, a dilated convolution is applied at a single location by:

$$X(t) * A = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} A(i,j) \times X(\, t + (d \times i)\, , j\,) \tag{4.7}$$

Notice that a dilation rate of $d = 1$ corresponds to a standard convolution. Introducing dilations allows a convolution with a fixed number of elements to capture larger global context of sequence data. Using TCNs for predictive modeling with time series data allows for each time step in a series to be processed in parallel and thus the inference speed of a TCN does not scale with input sequence length at the same rate as it does for an RNN.

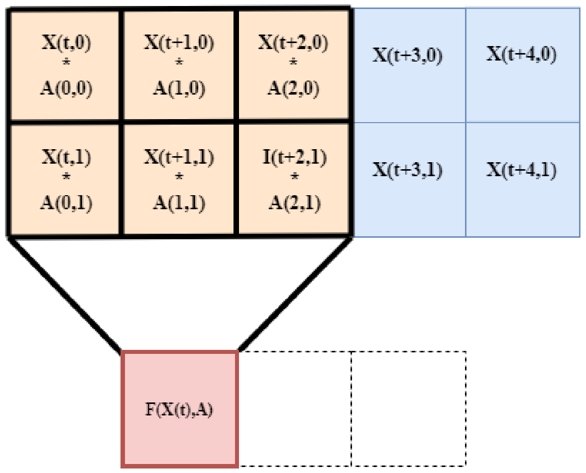


Figure 4.1. A 1D convolutional filter applied to a time series with multiple input features.

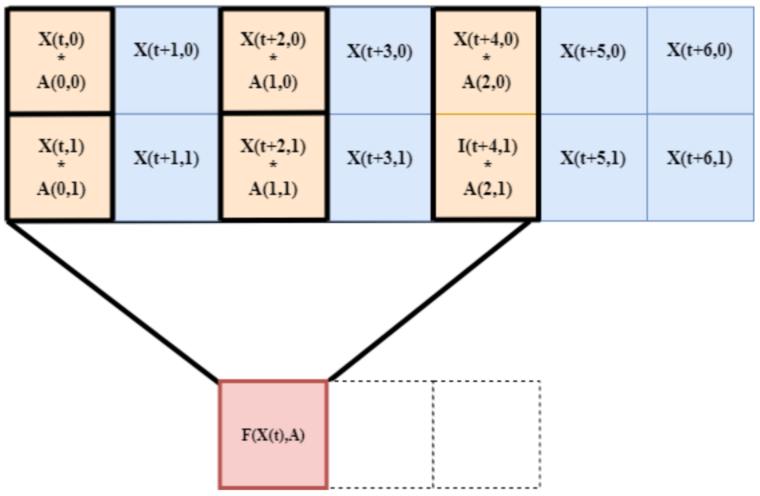


Figure 4.2. A dilated 1D convolutional filter applied to a time series. The location at which the convolutional filter elements are applied is controlled by the dilation hyperparameter.

4.3 Neural Networks for Classification and Predictive Modeling of Patient Mechanics

Methods

Two experiments were performed around the use of NNs for predictive modeling of patient mechanics. 70 TKR patients were fitted with 32 reflective markers used to define anatomical landmarks for 3D motion capture. Patients were instructed to perform multiple tasks including sit-to-stand, right and left sided step down, and gait. Tasks were performed onto a Bertec force platform embedded in the floor while force data was collected at 2000 Hz and an 8 camera Vicon motion capture system collected at 100 Hz. Step down was performed off of a stool with height 8” and sit-to-stand was performed with a chair of height 43”. The resulting data was processed in OpenSim (Fig. 4.3) to acquire kinematics, as well as joint reaction and muscle forces in the hip and knee, similar to the methods of Myers et al. (2018). All data was normalized to 100 time steps.

The full set of data consisted of 135 instances from 70 patients with 63 sit-to-stands, 15 right sided step downs, 14 left sided step downs, and 43 gait sequences. The resulting data was used in the development and evaluation of NNs for predictive modeling around gait lab data.

For the first experiment, NNs were trained to classify the ADL based on joint angle, ground reaction force, and anthropometrics (Figure 4.4). The data was split into 100 training instances, 15 validation instances, and 20 testing instances. This task was explored using TCNs as well as RNNs. The architectures used for training are presented in Tables 4.1 and 4.2. The RNN classification was taken by applying a softmax layer only at the final time step of the sequence, taking the form of a “many-to-one” configuration. Losses during training were backpropagated through all time steps from only the final time step. The TCN consisted of a series of convolutional layers and max pooling operations with a final fully-connected layer followed by a softmax layer for classification.

Training was performed using a learning rate of 1E-4 and stochastic gradient descent with a momentum of 0.9. A L1 regularization constant of 3E-3 was used to constrain the magnitude of the model’s parameters to address overfitting. This was found to be crucial given the small size of the training set. Early stopping was implemented using the validation instances by terminating the training process after the model’s performance on validation subjects ceased to improve for 20 epochs.

The second experiment explored the sequence-to-sequence learning problem with RNNs and TCNs. Each algorithm was given the input sequential data consisting of the kinematics, ground reaction forces, height, and weight and trained to predict hip and knee reaction forces as well as muscle forces (Figure 4.5). NNs were trained to predict hip and knee joint reaction forces, as well as vastus medialis and vastus lateralis muscle forces, throughout the ADL for both left and right sides. All 8 output sequences were predicted at once by a single model.

The RNN architecture utilized fully-connected layers with sigmoid activation functions, 3 LSTM cells, and skip connections (Table 4.3). For the RNN architecture, Training settings were consistent with the first experiment. Backpropagation-through-time was implemented over all time steps during a single training step. That is, the RNNs were fed all 100 time steps of a training instance before performing a parameter update.

The TCN architecture is detailed in Table 4.4. Convolutional layers used filters with length of 5 and were augmented with layer normalization and exponential linear units. After training, each NN was evaluated on the test set using correlation coefficient, median absolute error, and root-mean-square-error (RMSE).

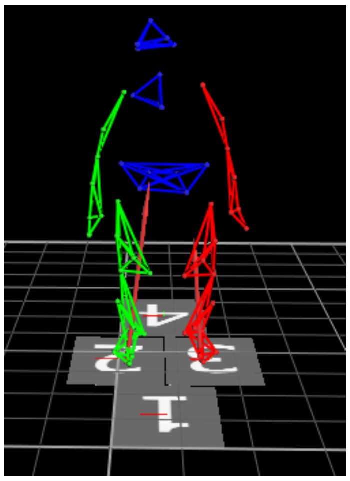


Figure 4.3. Marker data and ground reaction force data obtained from gait lab studies were processed in OpenSim to obtain patient mechanics.

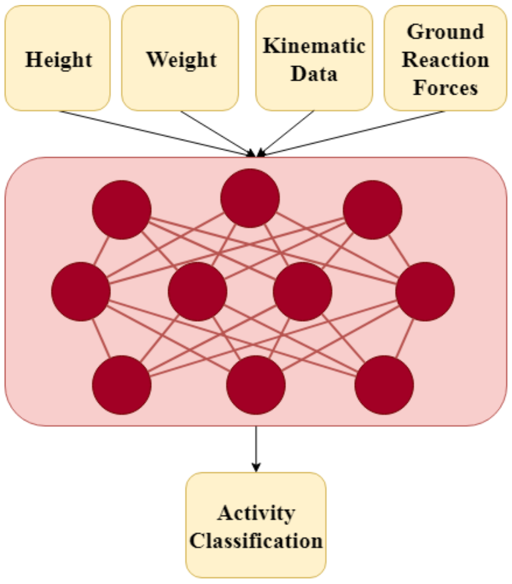


Figure 4.4. NNs were trained to classify an ADL based on patient kinematics, ground reaction forces, and anthropometrics.

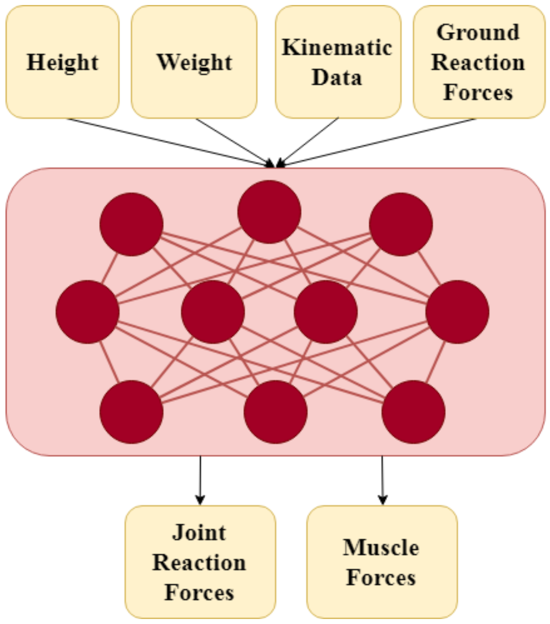


Figure 4.5. NNs were trained to predict muscle and joint reaction forces over time as a function of patient kinematics, ground reaction forces, and anthropometrics.

Table 4.1. RNN Architecture for Classification

Layer	Features
Layer 1	Fully-connected layer; 150 nodes
LSTM 1	State and cell sizes of 150; hidden state concatenated with layer 1
LSTM 2	State and cell sizes of 150; hidden state concatenated with layer 1 and LSTM 1
LSTM 3	State and cell sizes of 150; hidden state concatenated with layer 1 and LSTM 1,2
Layer 2	Fully-connected layer; 150 nodes
Layer 3	Fully-connected softmax classification layer, applied only at final time step.

Table 4.2. TCN Architecture for Classification

Layer	Features
Conv 1	32 filters, dilation rate of 1, followed by max pooling layer
Conv 2	64 filters, dilation rate of 2, followed by max pooling layer
Conv 3	128 filters, dilation rate of 4, followed by max pooling layer
Conv 4	256 filters, dilation rate of 8, followed by max pooling layer
Layer 1	Fully-connected layer, softmax output

Table 4.3. RNN Architecture for Prediction of Patient Mechanics

Layer	Features
Layer 1	Fully-connected layer; 150 nodes
LSTM 1	State and cell sizes of 150; hidden state concatenated with layer 1
LSTM 2	State and cell sizes of 150; hidden state concatenated with layer 1 and LSTM 1
LSTM 3	State and cell sizes of 150; hidden state concatenated with layer 1 and LSTM 1,2
Layer 2	Fully-connected layer; 150 nodes
Layer 3	Fully-connected 1D regression output layer

Table 4.4. TCN Architecture for Prediction of Patient Mechanics

Layer	Number of Filters	Dilation Rate
Conv 1	32	1
Conv 2	64	2
Conv 3	128	4
Conv 4	256	8
Conv 5	256	8
Conv 6	128	4
Conv 7	64	2
Conv 8	32	1
Conv 9	1	1

Results

Performance of the RNNs and TCNs for activity classification were evaluated on a test set of 20 patients performing different ADLs (Tables 4.5-4.6). The RNN predicted activity classification with 90% accuracy, only misclassifying two different step-downs as gait. The TCN achieved 100% accuracy on the classification task.

Correlation coefficient, median absolute error, and RMSE are presented in Table 4.7 for the evaluation of the second task. Qualitative results for each ADL class are presented in Figures 4.6-4.9. Both classes of NN were able to effectively estimate the trends of the output metrics over time. Correlation coefficients ranged from 0.786-0.926 for the RNN and 0.650-0.950 for the TCN across prediction tasks. Median absolute error for left and right knee reaction forces were 0.306 and 0.195 kN for the RNN and 0.438 and 0.166 kN for the TCN. Median absolute error for left and right hip reaction forces were 0.394 and 0.332 kN for the RNN and 0.365 and 0.307 kN for the TCN.

Table 4.5. Confusion matrix for activity classification with RNN. The RNN yielded an overall accuracy of 90%.

	Ground Truth Sit-to-Stand	Ground Truth Right Step Down	Ground Truth Left Step Down	Ground Truth Gait
Predicted Sit-to-Stand	11	0	0	0
Predicted Right Step Down	0	1	0	0
Predicted Left Step Down	0	0	0	0
Predicted Truth Gait	0	0	2	6

Table 4.6. Confusion matrix for activity classification with TCN. The TCN yielded an overall accuracy of 100%.

	Ground Truth Sit-to-Stand	Ground Truth Right Step Down	Ground Truth Left Step Down	Ground Truth Gait
Predicted Sit-to-Stand	11	0	0	0
Predicted Right Step Down	0	1	0	0
Predicted Left Step Down	0	0	2	0
Predicted Truth Gait	0	0	0	6

Table 4.7. Quantitative Results for Prediction of Patient Mechanics on a Test Subject

	RNN Corr. Coef.	TCN Corr. Coef.	RNN Med. Abs. Error (kN)	TCN Med. Abs. Error (kN)	RNN RMSE (kN)	TCN RMSE (kN)
Hip (R)	0.858	0.894	0.332	0.307	0.669	0.532
Knee (R)	0.910	0.881	0.195	0.166	0.540	1.101
Vastus Med. (R)	0.926	0.950	0.059	0.126	0.159	0.367
Vastus Lat. (R)	0.786	0.808	0.086	0.119	0.290	0.404
Hip (L)	0.815	0.650	0.394	0.365	0.914	0.947
Knee (L)	0.872	0.767	0.306	0.438	0.794	1.036
Vastus Med. (L)	0.894	0.793	0.071	0.155	0.171	0.326
Vastus Lat. (L)	0.807	0.814	0.097	0.152	0.278	0.409

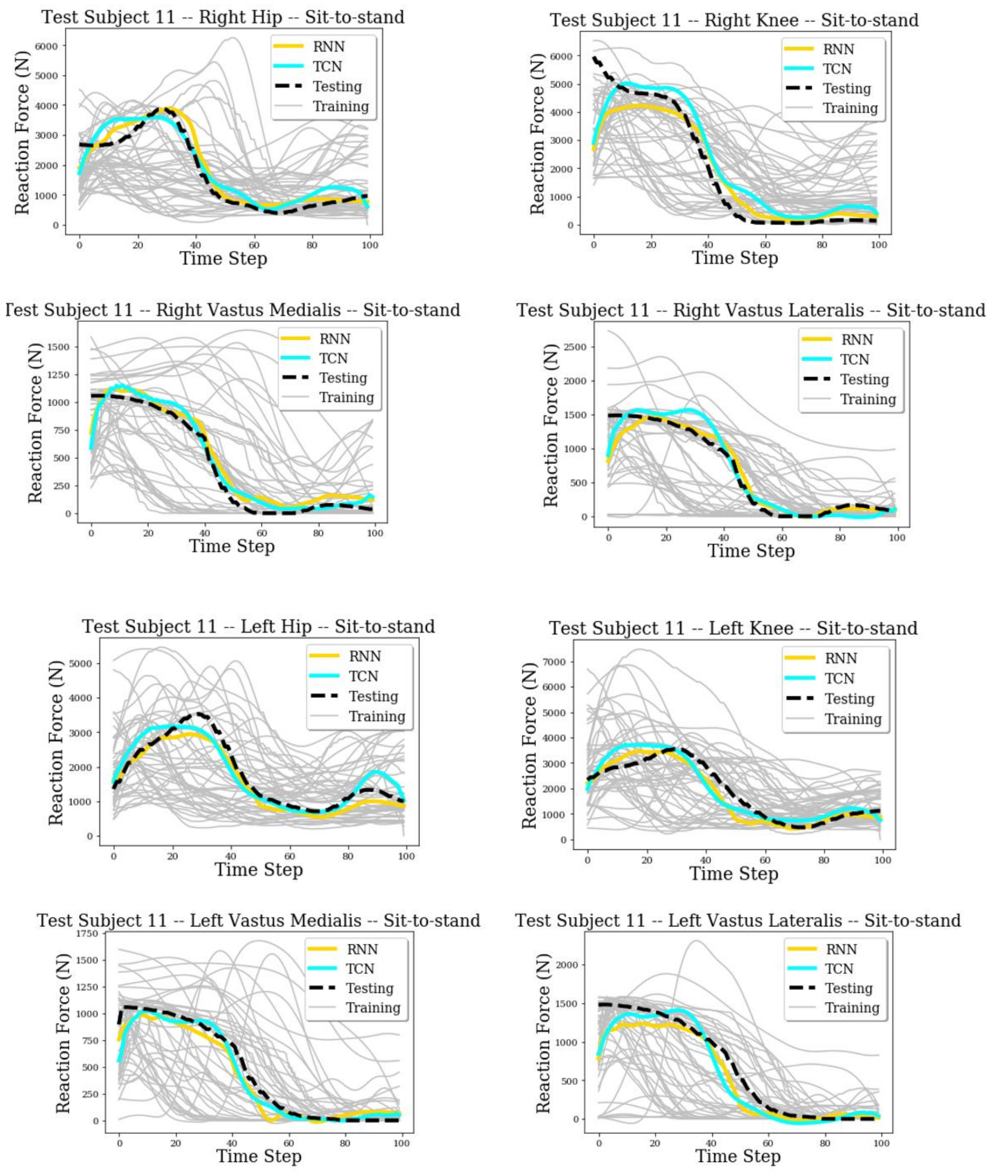


Figure 4.6. Predicted patient mechanics for a sit-to-stand ADL

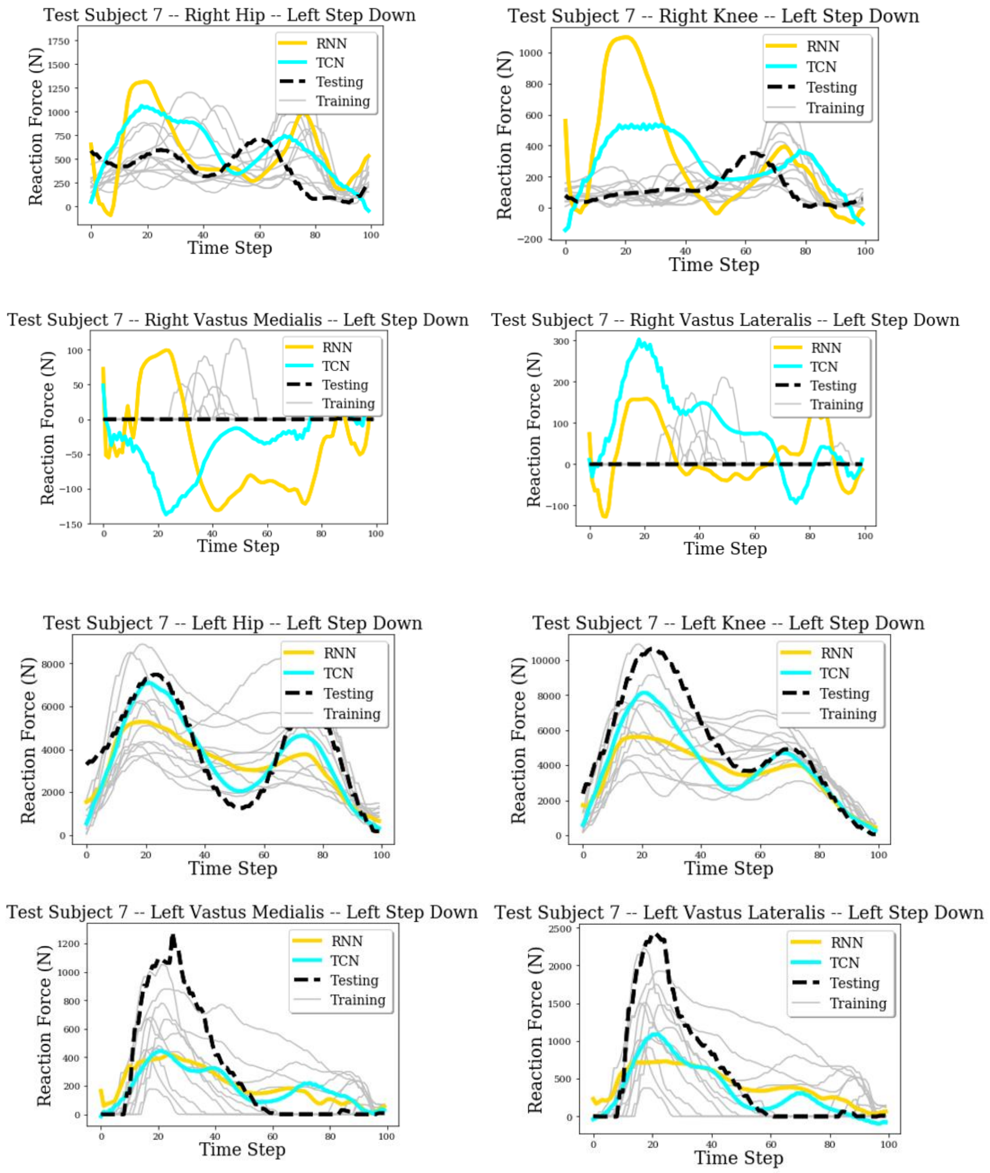


Figure 4.7. Predicted patient mechanics for a left step down ADL

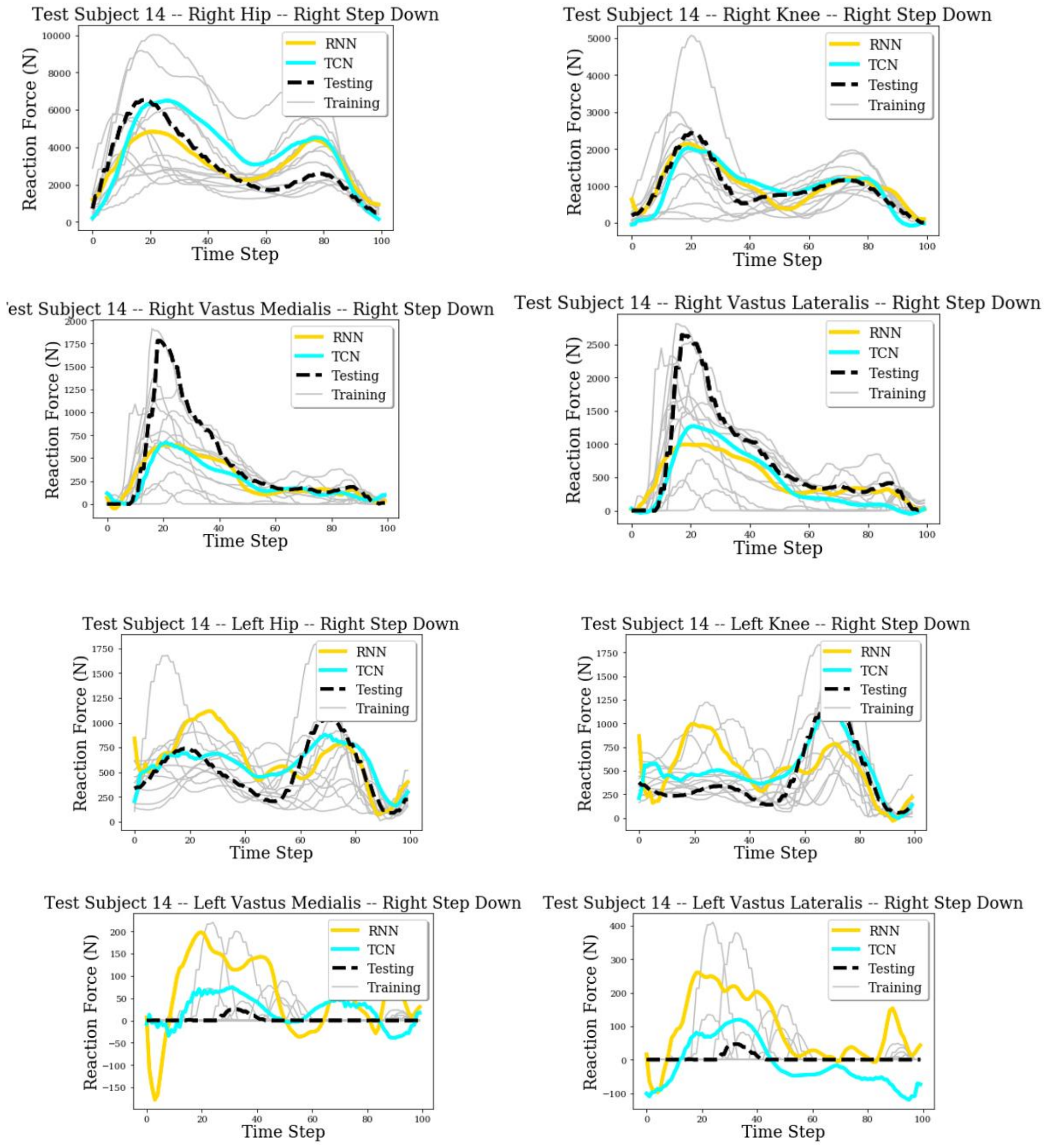


Figure 4.8. Predicted patient mechanics for a right step down ADL

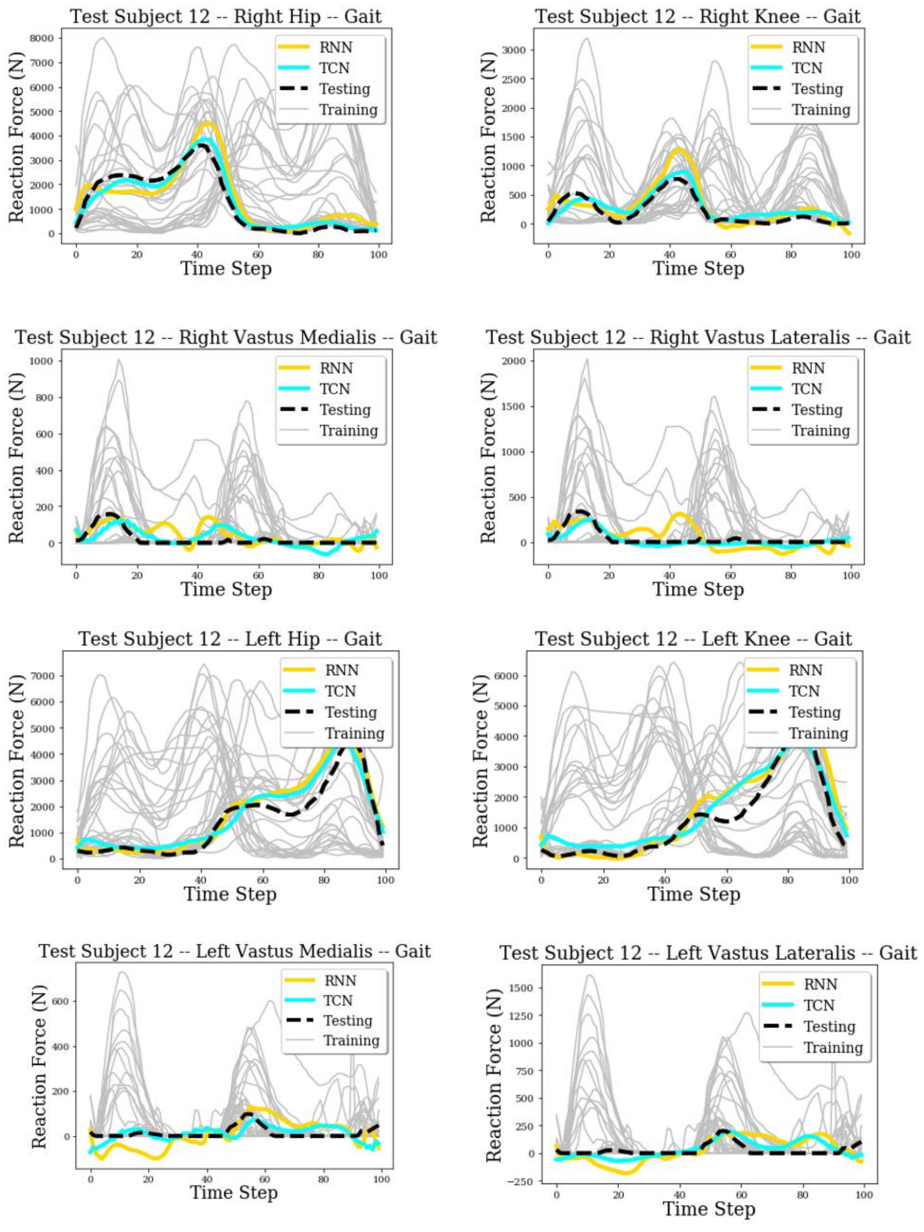


Figure 4.9. Predicted patient mechanics for gait

Discussion

The objective of these experiments was to assess the ability of deep learning techniques for the application of predictive modeling of patient mechanics. The performance of the models demonstrates the potential for deep learning to be used in a clinical system for estimating metrics of interest from standard gait lab data.

The applied NNs effectively bypass a single computational step in the gait lab workflow by estimating the output metrics of interest that would normally be obtained by performing a series of processing steps in specialized software. However, the model still requires ground reaction forces as input and joint angles, which are derived from marker data. As such, the models presented here would cut out only a single step. Future research will explore which other steps in the gait lab workflow can be substituted by data-driven methods with the overall goal of reducing hardware constraints, expertise requirements, and processing time for obtaining metrics of interest for patients performing ADLs. An idealized system would sufficiently predict joint reaction and muscle forces for patients given only a sequence of video frames and anthropometrics.

The lack of data presents a significant bottleneck for further development of these models. The current study was performed using a data set with only 135 instances that represented a limited number of tasks. More data is needed to develop truly robust models that can be trusted to the point of deployment.

4.4 Conclusion

Two different classes of NNs were developed for the classification and estimation of patient mechanics from standard gait lab data. Both classes of NN effectively estimated metrics of interest on a test set despite being trained with small quantities of data. The small quantities of available data as well as the cost of obtaining more present the largest obstacle to the development of a robust data-driven approach for estimation of mechanics for patients performing ADLs.

CHAPTER 5. A DEEP LEARNING-BASED COMPUTER VISION SYSTEM FOR DETECTION AND POSE ESTIMATION OF SURGICAL TOOLS

5.1 Surgical Tool Tracking

Real-time tracking of surgical tools has implications for assessment of surgical skill and workflow. Accordingly, efforts have been devoted to the development of systems that track the location of surgical tools in real-time.

Jin et al. (2018) proposed a convolutional neural network to classify different laparoscopic surgical tools and predict their spatial bounds in the image plane. Rieke et al. (2018) proposed to combine a template tracking approach with regression forests to predict bounding boxes and semantic key points of surgical tools used for retinal microsurgery. Du et al. (2016) used an algorithm based on SIFT (Lowe, 1999) features to predict the pose of surgical instruments from RGB frames. Laina et al. (2017) implemented a CNN to segment and localize laparoscopic surgical tools. These works proposed algorithms that returned spatial information about surgical instruments on the image plane. This 2D information alone may not sufficiently characterize surgical tool pose, depending on application. 3D information can subsequently be recovered using depth cameras or stereo vision.

Marker-based approaches have also been applied to surgical tool tracking. Fan al. (2018) developed an algorithm for registering markers on surgical tools for minimally-invasive surgery. A patterned cylindrical marker was attached to tools for convenient registration by Zhang et al. (2017). These approaches can accurately localize surgical tools in 3D space, which may be necessary for certain applications. However, this comes at the cost of bulky additions to the surgical tool, which can affect ergonomics, performance, and maneuverability.

A computer vision system that can detect and localize surgical tools in 3D space without significantly altering the shape of the tool would be beneficial for OR assessment. One potential approach to this is based on the estimation of objects in 3D space using only RGB images. An approach for estimating the 6-DoF pose of an object from RGB images was proposed by Pavlakos et al. (2017). The workflow consists of three steps: object detection, key point localization, and an optimization step. They used a CNN for object detection, a separate CNN for key point localization, and an optimization step to place an object of interest in 3D space by leveraging predicted key points, known object geometries, and camera geometry. This chapter expands upon this work and applies a similar approach to the detection and pose estimation of surgical tools used in total knee replacement (TKR).

An alternative to estimating the 6-DoF position of an object from RGB images is to directly use depth cameras or RGB-D cameras. Indeed, the use of depth cameras has shown success in various computer vision tasks (Shotton et al., 2013). This work focuses only on the use of RGB cameras because the use of color cameras is conducive to the proposed data synthetization pipeline, which is crucial for the training of robust CNNs. Additionally, the proposed system results in less expensive hardware. However, the use of RGB-D or depth cameras is a future research direction.

The objective of this chapter is to develop and evaluate a deep learning-based computer vision system for the detection and pose estimation of 2 types of surgical tools routinely used in TKR. The performance of the resulting system demonstrates the potential of the system to be used for operations research applications.

5.2 Deep Learning and Computer Vision

This section introduces concepts used in the development of the proposed system. The system relies on CNNs for object detection, CNNs for key point detection, and optimization concepts for estimating 6-DoF pose from RGB pixels.

Object Detection

The first step of the proposed workflow uses computer vision algorithms for detecting objects of interest. Object detection is more complicated than simple image classification because it does not assume there to be only a single object in an entire image. The objective of object detection is to predict the occurrence and class of potentially multiple instances in an image, and to predict the spatial bounds of each detected instance using a bounding box. State-of-the-art object detection is performed using deep learning (Girshick et al., 2015; Liu et al., 2016; Redmon & Farhadi, 2017; Redmon et al., 2016; He et al., 2017). The current work focuses on the class of object detectors known as single-shot detectors. An approach similar to the Single-Shot Detector (SSD) (Liu et al., 2016) is used for object detection.

The SSD approach uses a single forward pass of a fully-convolutional CNN for the detection, classification, and localization of objects in an image frame. An image frame is passed through convolutional and down sampling layers to aggregate global features from the image and obtain an $[N, M]$ feature map, where N and M are generally smaller than the original image dimensions. The CNN predicts bounding boxes for each cell in the $[N, M]$ grid. To do this, the CNN must predict the location of the instance centroid, bounding box dimensions, objectness score (i.e. confidence that there is an object), and classification of the detected instance. A box's centroid is encoded as its relative position within a given grid cell. It is constrained to the range [0,1] using a sigmoid function. Box dimensions are predicted as offsets from prior bounding box dimensions, known as anchor boxes. The objectness score is a confidence value for each anchor constrained to the range [0,1]. The classification prediction is a vector of length C for each anchor box with class-wise probabilities, conditioned on the presence of an object (Redmon et al., 2016).

It has been shown that training a single-shot object detector to leverage multiple anchor boxes leads to stronger performance on benchmark datasets (Liu et al., 2016). Accordingly, for A anchor boxes used in the model, the CNN outputs an $[N, M]$ feature map with depth of $A \times (5 + C)$, where C is the number of classes that the model is trained on.

Suppose there is an object detection CNN trained with only a single anchor box to detect C different classes. In this case, the CNN would output a tensor of shape $[N, M, (5 + C)]$. The values at position (c_x, c_y) of this tensor can be represented as $L = \{l_1, l_2, l_3, l_4, l_5, l_6 \dots l_{5+N}\}$. The bounding box prediction can be obtained for this location by:

$$b'_x = l'_1 + c_x/M \tag{5.1}$$

$$b'_y = l'_2 + c_y/N \tag{5.2}$$

$$b'_w = p_w e^{l'_3} \tag{5.3}$$

$$b'_h = p_h e^{l'_4} \tag{5.4}$$

$$b'_o = \ l'_5 \tag{5.5}$$

Here it is assumed that the values of l'_1, l'_2, and l'_5 have already been passed through the sigmoid function $\sigma(.)$. The values c_x and c_y are the cell coordinates of the predictions (indexing starting from 0), and p_w and p_h are the anchor box dimensions represented as percentage of original image dimensions. These equations return the location of a predicted box's centroid (b'_x, b'_y) and dimensions (b'_w, b'_h) in percentage of the original image's dimensions. The value b_o is the objectness score.

During inference, objectness scores greater than some threshold (0.5 for instance) are taken as detected instances. The other outputs are then used to infer the bounding box and instance class. Additionally, in practice, a CNN may be responsible for detecting objects based on multiple anchor boxes and may also use multiple scales to detect objects. Figure 5.1 depicts a bounding box prediction module for a single anchor box with 2 classes.

The model is trained by taking bounding box annotations from a known training set and encoding them into the format required by the CNN. Then, a loss function is used to tune each component of the model. Thus, the loss function for training a CNN for object detection is a linear combination of the errors of the predicted objectness score, bounding box dimensions and locations, and classification. This error is computed over all scales and all anchor boxes at each scale for which there are bounding box prediction modules. The following loss function is used in this chapter, and slightly differs from the YOLO object detection literature (Redmon et al., 2016):

$$L_{box}(\theta) = \sum_{m=0}^{S} \sum_{i=0}^{N\times M} \sum_{j=0}^{B_m} \lambda_{obj} \mathbb{I}_{i,j}^{obj} [(l_{1,i,j} - l'_{1,i,j})^2 + (l_{2,i,j} - l'_{2,i,j})^2] + \lambda_{obj} \mathbb{I}_{i,j}^{obj} [(l_{3,i,j} - l'_{3,i,j})^2 + (l_{4,i,j} - l'_{4,i,j})^2] + \lambda_{obj} \mathbb{I}_{i,j}^{obj} (l_{5,i,j} - l'_{o,i,j})^2 + \lambda_{no-obj} \mathbb{I}_{i,j}^{no-obj} (l_{5,i,j} - l'_{5,i,j})^2 + \lambda_{obj} \mathbb{I}_{i,j}^{obj} \sum_{n=1}^{C} (l_{5+n,i,j} - l'_{5+n,i,j})^2 \tag{5.6}$$

Where the loss function sums over S prediction modules. In this equation, notation is slightly abused by assuming that each value is specific to its appropriate scale. As can be seen, the bounding box dimensions and locations, as well as the class predictions, are only penalized given that there is a ground truth box at that location. Also, locations with a ground truth bounding box are penalized differently than those without. This is because the number of locations without instances greatly outnumbers the occurrence of instances during training. To address this, $\lambda_{obj} = 20$ and $\lambda_{no-obj} = 0.1$ are used.

The fact that each bounding box prediction module in a CNN uses multiple anchor boxes necessitates a strategy for choosing which anchor box should be "responsible" for detecting an object during training. The method of SSD is adopted for this. Specifically, given the ground truth bounding box of an object in the training set, the anchor box whose dimensions have the highest IoU with the object's bounding box is assigned that instance for training. Additionally, any anchor box whose dimensions result in a sufficient overlap with the ground truth box is also assigned this instance for training. An IoU threshold of 0.5 is used to control this. As a result, an object of interest will always be paired with an anchor box, but also any anchor box that is deemed to be similar enough to the ground truth box in question will also be paired with it.

During inference, box predictions are made by deriving bounding box characteristics when a certain confidence threshold is reached. It is possible that multiple anchor box predictions at multiple grid cells and anchor boxes will detect the same object instance. Therefore, non-max suppression (NMS) is used to filter out redundant predictions. Predicted bounding boxes that have an IoU greater than 0.5 with other predictions of a similar class are kept in a greedy fashion based on the confidence of each box.

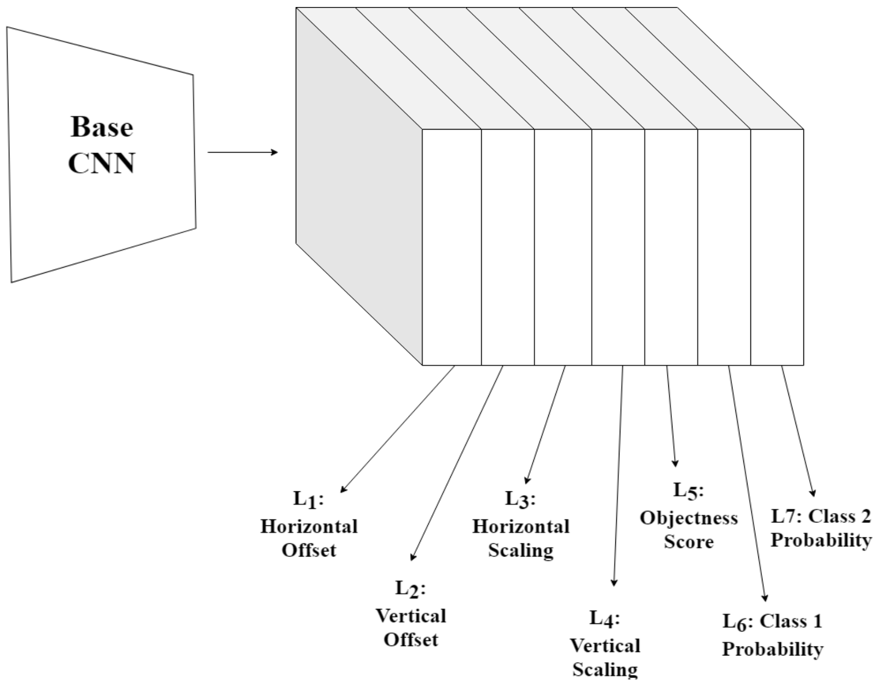


Figure 5.1. A bounding box prediction module from a CNN with a single anchor box. Each feature map in the depth direction is responsible for a different variable of interest needed to produce bounding box predictions. Modules are augmented with multiple anchor boxes by concatenating multiple tensors of this format in the depth direction.

Key Point Detection

CNNs can be used to detect semantic key points on an object of interest in an image. The structure of these CNNs is similar to those used for segmentation. CNNs are structured for key point localization with heat map outputs. That is, given an $[N, M, 3]$ RGB input image, the CNN will output an $[N, M, D]$ feature map where D is the number of key points that the model is trained to predict. Key points are obtained by taking the maximum activation over all $N \times M$ cells for each of D key points. The loss function for key point localization is considerably more manageable than the one used for bounding boxes:

$$L_{kp}(\theta) = \sum_{i=0}^{N\times M} \sum_{j=0}^{D} \lambda_{in-box} \mathbb{I}_{i,j}^{in-box} [(l_{i,j} - l'_{i,j})^2] + \lambda_{out-box} \mathbb{I}_{i,j}^{out-box} [(l_{i,j} - l'_{i,j})^2]$$

(5.7)

Where $l_{i,j}$ is the heat map value at the i^{th} pixel in channel j. This loss function is essentially an element-wise squared error over all outputs. Similar to the bounding box loss function, it is preferred to weight the loss higher for the sparser "positive" examples. In the case of key point predictions, this means weighting the loss higher for CNN outputs that are spatially close to the object of interest. The bounding box annotations from the training set are used for this. The element-wise error is weighted higher for elements that are within a bounding box, and lower for elements that are not. In practice this concept manifests in the form the of a "loss mask" that is applied to the squared error at each element.

6-DoF Pose Estimation from RGB Images

Estimating the 6-DoF pose of an object from a single RGB image is a well-studied topic in computer vision (Zhou et al., 2015). The approach presented in this section is based on Pavlakos et al. (2017). This problem can be solved using optimization, where the objective function is based on the full-perspective camera model. The intuition behind the full-perspective model is that each pixel in an image corresponds to a vector extending out into 3D space. The direction of this vector is determined by a series of parameters called the camera intrinsic parameters. However, in the case of RGB images, it is unknown how far this vector should extend into space for a given key point.

The optimization step attempts to place a prior known geometry into 3D space by finding the correct orientation of the geometry as well as the correct vector for each predicted key point (Figure 5.2).

Let $K_x \in \mathbb{R}^{1\times p}$ and $K_y \in \mathbb{R}^{1\times p}$ represent the vectors of x- and y-coordinates in pixel dimensions of the predicted key points for an object. These coordinates are converted to homogenous coordinates $W \in \mathbb{R}^{3\times p}$ using the camera intrinsic parameters, where $w_i \in \mathbb{R}^3$ is the homogenous vector for the i^{th} key point. The homogenous coordinates can be determined based on:

$$W = [\ ((K_x^T - p_x)/f_x)\ \big|\ ((K_y^T - p_y)/f_y)\ \big|\ 1\ ]^T \tag{5.8}$$

f_x and f_y are the camera's focal lengths, and p_x and p_y represent the principal point of the image plane. The homogenous coordinates are used to define an objective function that relates the pose of a known object geometry in 3D space to corresponding homogenous coordinates of the key points:

$$L(\theta) = \frac{1}{2} \times \left\|(WZ - RB - T1^T)D^{1/2}\right\|_F^2 \tag{5.9}$$

Where $R \in \mathbb{R}^{3\times 3}$ is a rotation matrix, $T \in \mathbb{R}^3$ is a translation vector, $Z \in \mathbb{R}^{p\times p}$ is a diagonal matrix, $B \in \mathbb{R}^{3\times p}$ represents the 3D Cartesian coordinates of the semantic key points on the object geometry in 3D space (STL nodes), and $D \in \mathbb{R}^{p\times p}$ is a diagonal matrix where $d_{i,i}$ is the confidence of the prediction for key point i. Leveraging the prediction confidence using the matrix D allows the objective function to penalize confident key points more harshly. In order to estimate the object pose in 3D space, the objective function must be minimized with respect to $\theta = \{Z, R, T\}$.

The optimization problem is solved using block coordinate descent. That is, the objective function is iteratively minimized while optimizing for one of the 3 parameters while holding the other two parameters constant. The updates for each of these individual parameters can be found with closed-form solutions. By relating the squared Frobenius norm of a term to the trace of a matrix, the updates for each term can be found using the classic least-squared approach.

$$\|A\|_F^2 = tr(A^T A) = tr(AA^T) \tag{5.10}$$

The least-squares approach is performed by equating the gradient of the function ($\frac{\partial L(\theta)}{\partial \theta}$) to 0 and isolating the parameter of interest. The updates for T are directly performed in this way:

$$T = (\frac{2}{t_3 + t_4}(t_1 - t_2))^T \tag{5.11}$$

$$t_1 = 1^T D^{\frac{1}{2}}(WZD^{\frac{1}{2}})^T \tag{5.12}$$

$$t_2 = 1^T D^{\frac{1}{2}}(RBD^{\frac{1}{2}})^T \tag{5.13}$$

$$t_3 = 1^T D 1 \tag{5.14}$$

$$t_4 = (1^T D 1)^T \tag{5.15}$$

Updates for the other two parameters necessitate additional considerations. R is a rotation matrix and thus must satisfy the constraint that $R^T R = 1$ and $\det(R) = 1$. In order to update R, the objective function is rearranged to match the form needed to perform orthogonal Procrustes analysis (Schönemann, 1966) and then solved accordingly:

$$X = (BD^{\frac{1}{2}})^T \tag{5.16}$$

$$Y = (WZD^{1/2} - T1^T D^{1/2})^T \tag{5.17}$$

$$U, S, V^T = SVD(X^T Y) \tag{5.18}$$

$$R = UV^T \tag{5.19}$$

Z must satisfy the constraint that it is a diagonal matrix so off-diagonal elements must be zero. The update of Z cannot be performed all at once. Instead, each diagonal element of Z is treated as its own least-squares problem and the updates are determined accordingly.

$$z_{i,i} = (X^T X)^{-1} X^T Y \tag{5.20}$$

$$X = d_{i,i}^{1/2} \times w_i \tag{5.21}$$

$$Y = d_{i,i}^{1/2} \times (Rb_i + T) \tag{5.22}$$

Given a series of predicted key points in the image plane as well as the corresponding object geometry, the optimization step is performed by iteratively updating R, T, and Z until some convergence criteria is met (Figure 5.2). The resulting values for R and T determine where the object lies in the local camera coordinate system. The diagonal elements of Z describe the distance of the key points on the object from the image plane. All nodes of the object geometry can be re-projected back onto the image plane to qualitatively verify the result.

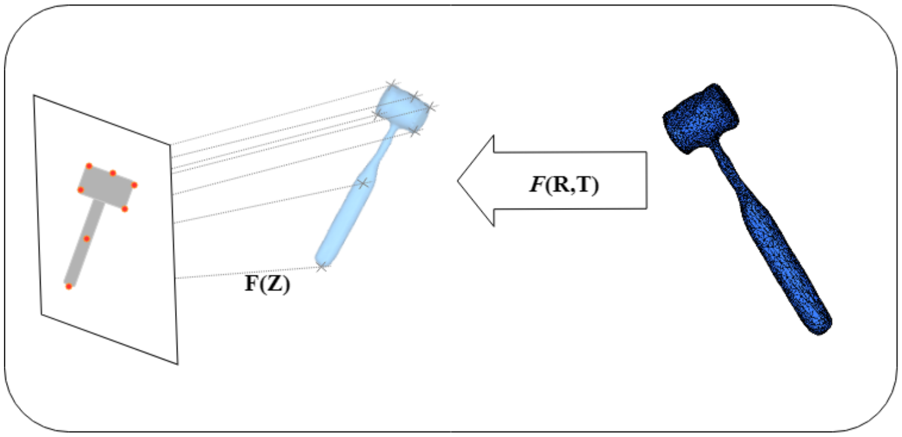


Figure 5.2. An illustration of the optimization problem for determining 6-DoF pose from predicted key points on the image plane. The optimization step minimizes a residuals function based on R, T, and Z

5.3 A Deep Learning-Based Computer Vision System for Estimating Surgical Tool Pose from RGB Images

Overview

A computer vision system was developed for the detection and pose estimation of 2 different surgical tools used in TKR from RGB video frames. The proposed system is illustrated in Figure 5.3 and is largely based on Pavlakos et al. (2017). The system initially detects object instances using the described object detection formulation of CNNs. Key point predictions are extracted from within the spatial bounds of each bounding box using the same CNN architecture, after applying NMS. The CNN is trained to predict key points for all object classes. Therefore, the class of the predicted bounding box dictates which key point predictions are extracted from the CNN and passed to the optimization step. The optimization step takes as input the predicted key points (converted to homogenous coordinates) as well as the prior known geometry of the predicted object class in the form of a point cloud. Finally, the block coordinate descent scheme is executed to estimate the object's position in the camera's coordinate system. The tools supported by the model are presented in Figures 5.4 and 5.5, with corresponding semantic key points.

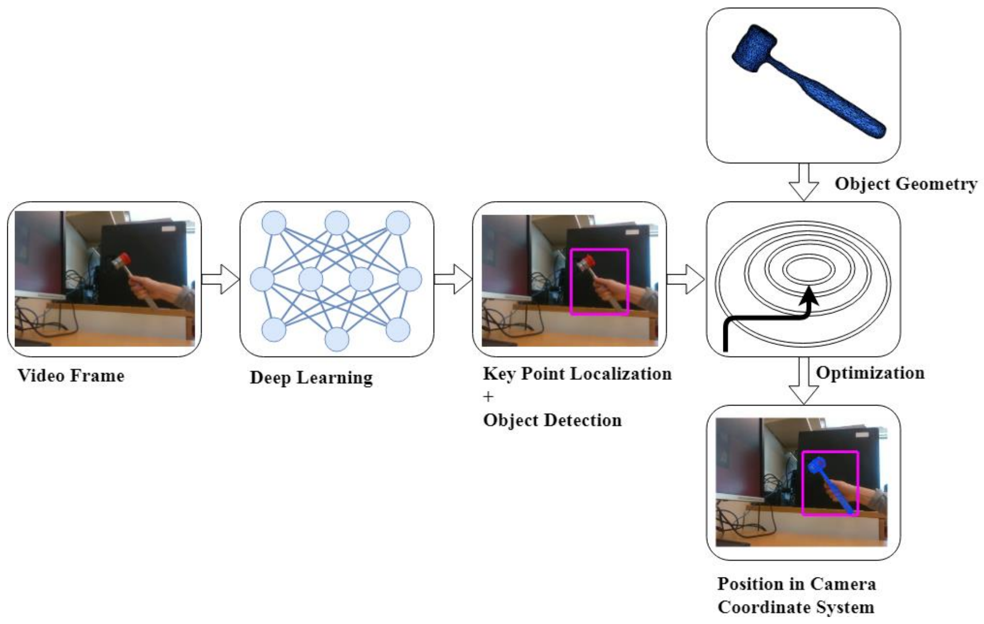


Figure 5.3. The proposed computer vision system. A CNN extracts bounding boxes and key point predictions from a given frame and the resulting key points are fed to an optimization step to yield 6-DoF pose.

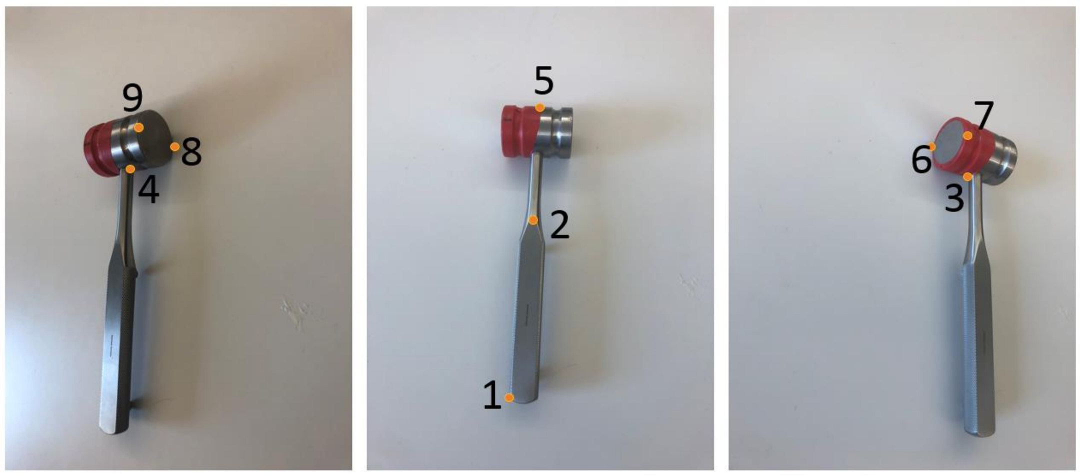


Figure 5.4. The first TKR tool that the proposed system was developed for was a hammer.

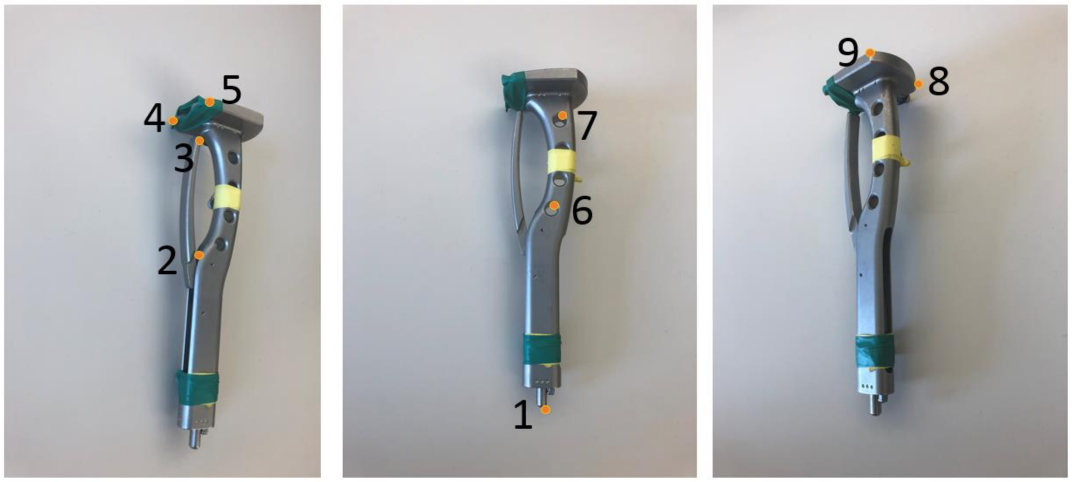


Figure 5.5. The second TKR tool that the proposed system was developed for was a broach handle.

Data

This section describes the approaches used to obtain the data necessary for training the system. CNNs have largely dominated the object detection space in recent years, with most benchmark leaderboards consisting primarily of deep learning-based approaches (Everingham et al., 2010; Lin et al., 2014). These benchmarks provide thousands of training instances on which researchers develop their algorithms. The resulting models serve as robust object detectors with the ability to detect a variety of generic classes. But meeting the data requirement for training CNNs on a niche task presents a challenge. For the proposed computer vision system, training data was initially aggregated by manually annotating 5,000 images with bounding boxes and key points. This resulted in insufficient performance, even after significant hyperparameter tuning. Therefore, a pipeline was developed for generating synthetic data with ground truth annotations.

A series of video frames of each object class was captured (about 75 frames per object class). The MATLAB (Mathworks, Exeter, UK) segmenter toolbox implementation of graph cut (Boykov et al., 2001) was used to generate masked images of just the tool in each video frame (Figure 5.6). Then, a proprietary GUI was developed to annotate the bounding boxes and key points on each of these frames (Figure 5.7). As a result, each original captured frame was associated with a segmented image of a tool, a bounding box, and key points.

Large-scale image datasets were downloaded and combined with the segmented images in the data synthetization pipeline. Specifically, the pipeline randomly sampled a background image and a segmented tool mask with corresponding annotations. Random perturbations were applied in the form of scaling, rotation, translation, and brightness before placing the segmented tool mask image on the sampled background. The pipeline sampled 1-3 tools per image.

This pipeline allowed for the generation of thousands of instances for the training of CNNs. Additionally, an improved version of the pipeline simulated tool movement over multiple frames by randomly placing a tool, and then sampling small movements and rotations of each tool in subsequent frames (Figure 5.8). This approach was used to train a specialized CNN that leveraged temporal context.

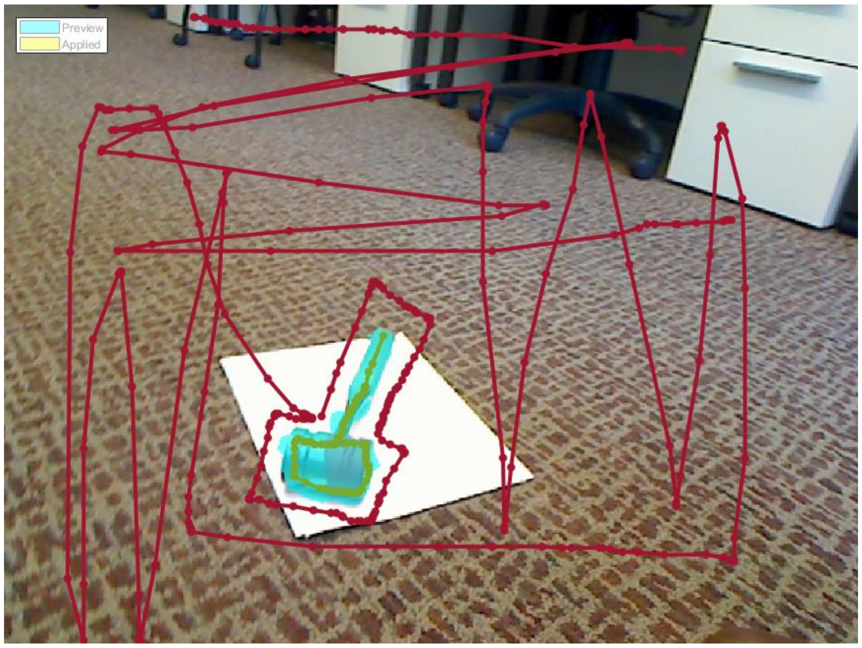
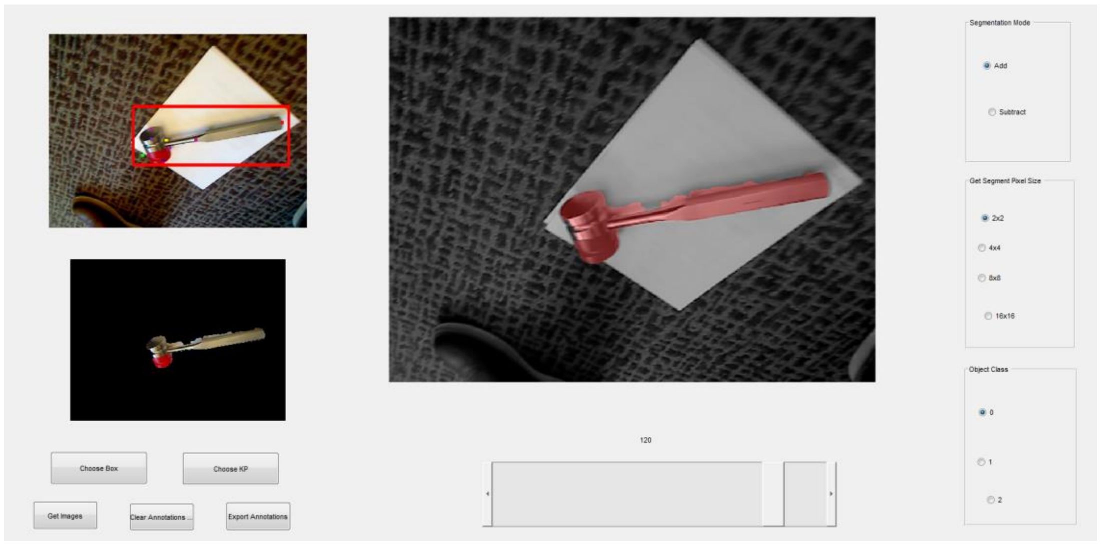


Figure 5.6. Graph cut results for quick segmentation of surgical tools.



Segmentation Mode
[✓ Add]
Subtract

Get Segment Pixel Size
[✓ 2x2]
4x4
8x8
16x16

Object Class
[✓ 0]
1
2

Figure 5.7. GUI for data annotation. The GUI imports an image, corresponding segmentation mask, and allows for annotation of boxes and key points for multiple object types.

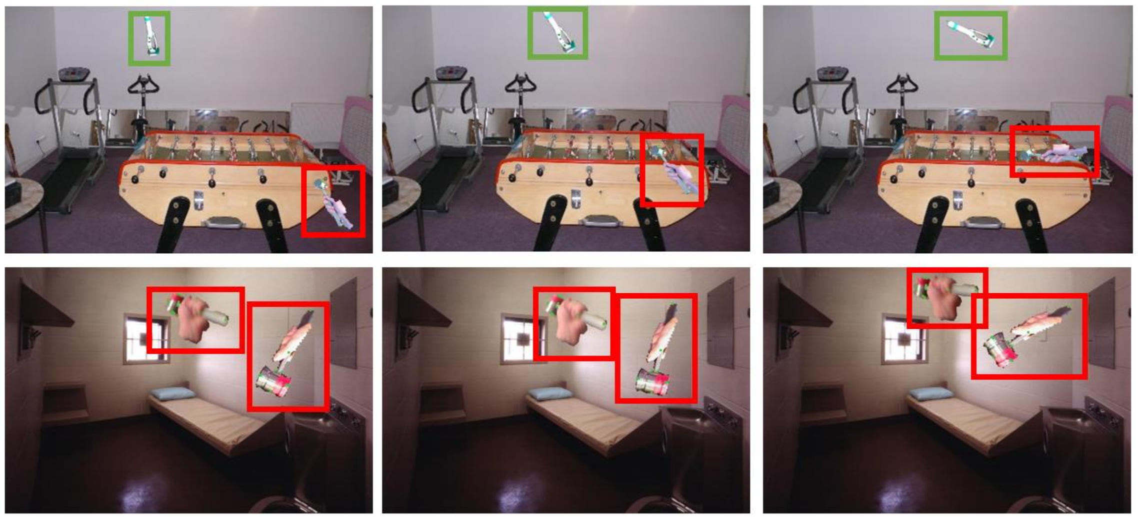


Figure 5.8. Two sequences of generated training data with overlaid ground truth that simulate tool movement.

Multitask CNNs for Object Detection and Key Point Prediction

The proposed computer vision system necessitated object detection and key point prediction. Both of these tasks were accomplished using multitask CNNs. Three different CNN architectures were explored.

The first architecture was characterized by an encoder-decoder structure composed of residual units, residual skip connections between corresponding scales of the encoder and decoder, and atrous spatial pyramid pooling modules (Chen et al., 2016) (Figure 5.9). The CNN was trained to predict bounding boxes at two different scales and key points at the final layer. Each bounding box prediction module was associated with 36 anchor boxes that resulted from all combinations of the set $\{0.15, 0.3, 0.45, 0.6, 0.75, 0.9\}$ (Where the units are in percentage of image dimension). Deep supervision was used at scales that weren’t responsible for box or key point predictions. Accordingly, the total loss function of the CNN was defined by:

$$L(\theta) = L_{box}(\theta) + L_{kp}(\theta) + L_{deep-sup}(\theta) + \lambda\|\theta\|_2^2 \quad (5.23)$$

The CNN was trained using images of resolution 256 by 352 and gradient clipping. This CNN was named Multitool.

One fundamental issue of Multitool was that it failed to leverage temporal context. That is, given a single frame in a video sequence, the CNN had no way of aggregating information from previous frames to inform predictions in the current frame. A second CNN architecture, called Multitool2 (Figure 5.10), was developed in an attempt to address this issue. The CNN architecture was built by augmenting the original Multitool with convolutional long short-term memory (CLSTM) units. CLSTMs are a reformulation of the original LSTM that replace the underlying matrix transformations with convolutional operators. This allows a CNN to leverage temporal context without being subjected to the computational expenses associated with the fully-connected layers of the original LSTM formulation. Given the current time step's tensors X_t, H_t, and C_t of the input, hidden state, and cell state, the CLSTM units were implemented similar to (Shi et al., 2015):

$$i_t = \sigma(W_1 * X_t + W_2 * H_{t-1} + b_1) \quad (5.24)$$

$$f_t = \sigma(W_3 * X_t + W_4 * H_{t-1} + b_2) \quad (5.25)$$

$$C_t = f_t \circ C_{t-1} + i_t \circ \tanh(W_5 * X_t + W_6 * H_{t-1} + b_3) \quad (5.26)$$

$$o_t = \sigma(W_7 * X_t + W_8 * H_{t-1} + b_2 + b_4) \tag{5.27}$$

$$H_t = o_t \circ \tanh(C_t) \tag{5.28}$$

Where W_i's are learned convolutional filters, b_i's are learned channel-wise biases, and $*$ and $\circ$ denote convolution and Hadamard product, respectively.

The CLSTMs were placed at two different scales of the network, directly before each bounding box prediction module. The inputs to the CLSTM units were concatenated with the outputs:

$$O_t = [H_t \; X_t] \tag{5.29}$$

Multitool2 was trained in the same way as the first CNN except that the training instances used were the frame sequences that simulated tool movement, as described in the previous section. Details for both CNNs are presented in Tables 5.1 and 5.2.

Both CNNs were initially trained with a batch size of 1 to allow for quick training iterations. Using a batch size of 1 resulted in poor convergence so a batch size of 10 was subsequently used. However, the preprocessing required for setting up batches with size of 10 was computationally expensive. Naively importing an image with annotations and then encoding these annotations into the format required by the CNN resulted in training steps that took 3 seconds each. Therefore, a parallel processing pipeline was developed to allow for faster training iterations. First, a list of 10 batches (with each batch comprising

10 training instances) with information properly encoded into the format required by the CNN was generated before beginning training. Training iterations were performed by randomly sampling from the list of preprocessed batches. The list of batches was updated in parallel by asynchronously generating a training instance and assigning the result to the instance queue upon completion (Figure 5.11). As a result, computational time for a single training iteration was reduced to about 0.9 seconds, which effectively sped up training for a given number of training instances by 200%.

A third CNN architecture, called SegBox, reformulated the object detection problem as a segmentation problem. The bounding box modules towards the throat of the network were replaced with segmentation map outputs at the end of the network (Figure 5.12, Table 5.3). A single segmentation map was allocated for each class of the model and the model was formulated such that the pixel-wise classifications at each location were not mutually exclusive. This is intuitive because, given an image, a pixel may correspond to multiple objects at different depths.

Thus, this third architecture terminated with a single tensor that was responsible for both segmentation as well as key point predictions (Figure 5.13). The segmentation feature maps were trained using dice loss and the key point predictions were trained as usual. During inference, object predictions were taken by thresholding segmentation maps at a confidence level of 0.5 to get binarized maps. Dilation was applied with the purpose of joining potentially disconnected components. A connected components algorithm was applied to find the largest component in each segmentation map (Each class). The bounds of the resulting component were taken as the bounding box prediction.

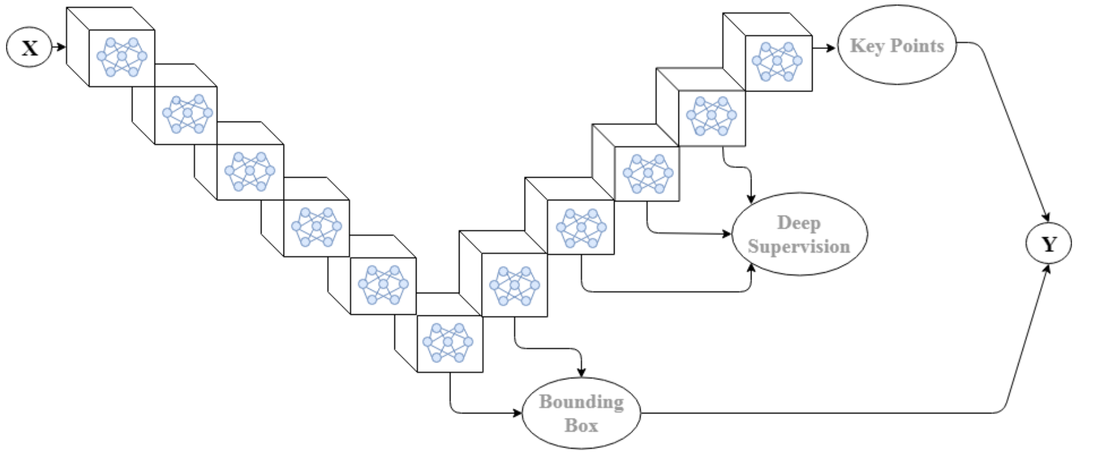


Figure 5.9. A representation of the Multitool architecture. Blue cubes represent residual modules. The scale of feature maps are down sampled and up sampled 5 times using max pooling and transpose convolutional layers. 2 scales of the CNN are augmented with bounding box prediction modules. The final scale is responsible for key point predictions. The remaining scales of the decoder are augmented with deep supervision modules for accelerated training.

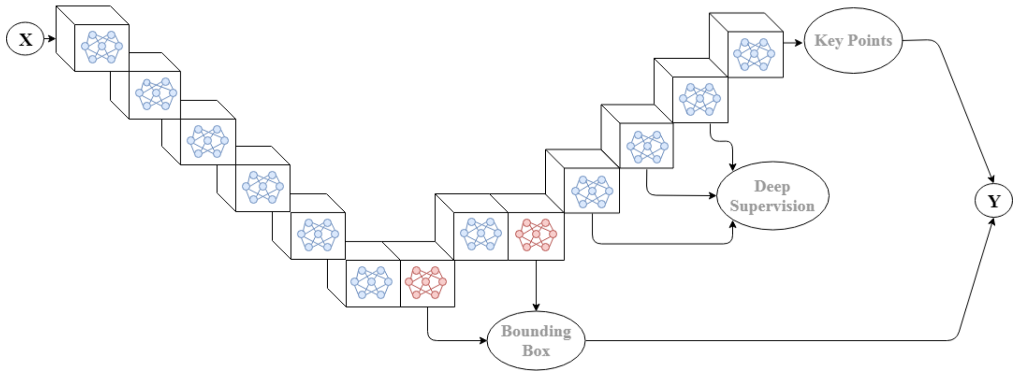


Figure 5.10. A representation of the Multitool2 architecture. Multitool2 is similar to the first architecture except that CLSTM modules are added to the architecture prior to each bounding box prediction, as shown by the red cubes.

Table 5.1. Details for Multitool.

Layer	Number of Filters	Features	Prediction Module
Conv 1	64	Residual module; followed by max pool	
Conv 2	128	Residual module; followed by max pool	
Conv 3	256	Residual module; followed by max pool	
Conv 4	512	Residual module; followed by max pool; followed by dropout layer	
Conv 5	1024	Residual module; followed by max pool; followed by dropout layer	
Conv 6	1024	Atrous spatial pyramid pooling module	
Conv 7	1024	Atrous spatial pyramid pooling module; followed by dropout layer	Object detection module
Upconv 1	1024	Followed by residual connection with Conv 5	
Conv 8	1024	Residual module; followed by dropout layer	Object detection module
Upconv 2	512	Followed by residual connection with Conv 4	
Conv 9	512	Residual module	Deep supervision module
Upconv 3	256	Followed by residual connection with Conv 3	
Conv 10	256	Residual module	Deep supervision module
Upconv 4	128	Followed by residual connection with Conv 2	
Conv 11	128	Residual module	Deep supervision module
Upconv 5	64	Followed by residual connection with Conv 1	
Conv 12	64	Residual module	Key point prediction module

Table 5.2. Details for Multitool2

Layer	Number of Filters	Features	Prediction Module
Conv 1	64	Residual module; followed by max pool	
Conv 2	128	Residual module; followed by max pool	
Conv 3	256	Residual module; followed by max pool	
Conv 4	512	Residual module; followed by max pool; followed by dropout layer	
Conv 5	1024	Residual module; followed by max pool; followed by dropout layer	
Conv 6	1024	Atrous spatial pyramid pooling module	
Conv 7	1024	Atrous spatial pyramid pooling module; followed by dropout layer; augmented with CLSTM	Object detection module
Upconv 1	1024	Followed by residual connection with Conv 5	
Conv 8	1024	Residual module; followed by dropout layer; augmented with CLSTM	Object detection module
Upconv 2	512	Followed by residual connection with Conv 4	
Conv 9	512	Residual module	Deep supervision module
Upconv 3	256	Followed by residual connection with Conv 3	
Conv 10	256	Residual module	Deep supervision module
Upconv 4	128	Followed by residual connection with Conv 2	
Conv 11	128	Residual module	Deep supervision module
Upconv 5	64	Followed by residual connection with Conv 1	
Conv 12	64	Residual module	Key point prediction module

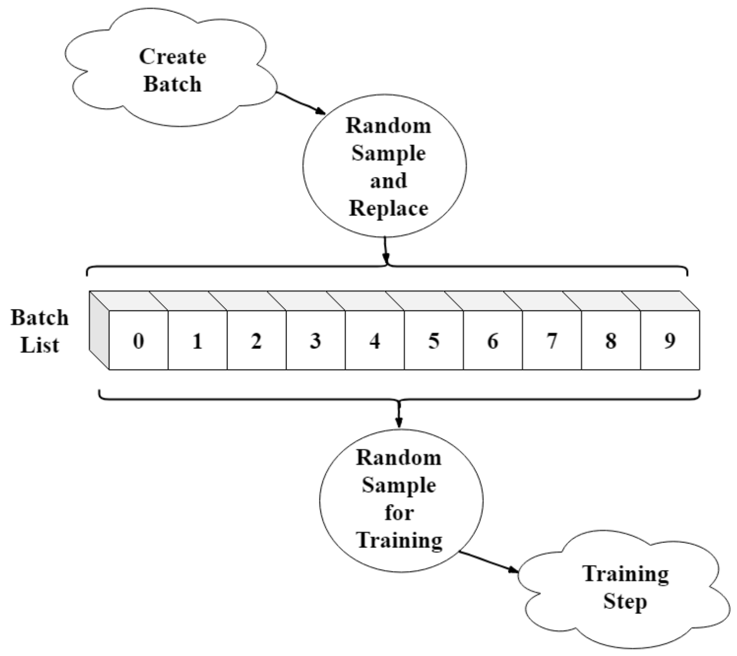


Figure 5.11. A pipeline for asynchronous execution of batch preprocessing and training steps.

Table 5.3. Details for SegBox.

Layer	**Number of Filters**	**Features**	**Prediction Module**
Conv 1	64	Residual module; followed by max pool	
Conv 2	128	Residual module; followed by max pool	
Conv 3	256	Residual module; followed by max pool	
Conv 4	512	Residual module; followed by max pool; followed by dropout layer	
Conv 5	1024	Residual module; followed by max pool; followed by dropout layer	
Conv 6	1024	Atrous spatial pyramid pooling module	
Conv 7	1024	Atrous spatial pyramid pooling module; followed by dropout layer	
Upconv 1	1024	Followed by residual connection with Conv 5	
Conv 8	1024	Residual module; followed by dropout layer	Deep supervision module
Upconv 2	512	Followed by residual connection with Conv 4	
Conv 9	512	Residual module	Deep supervision module
Upconv 3	256	Followed by residual connection with Conv 3	
Conv 10	256	Residual module	Deep supervision module
Upconv 4	128	Followed by residual connection with Conv 2	
Conv 11	128	Residual module	Deep supervision module
Upconv 5	64	Followed by residual connection with Conv 1	
Conv 12	64	Residual module	Segmentation + key point prediction module

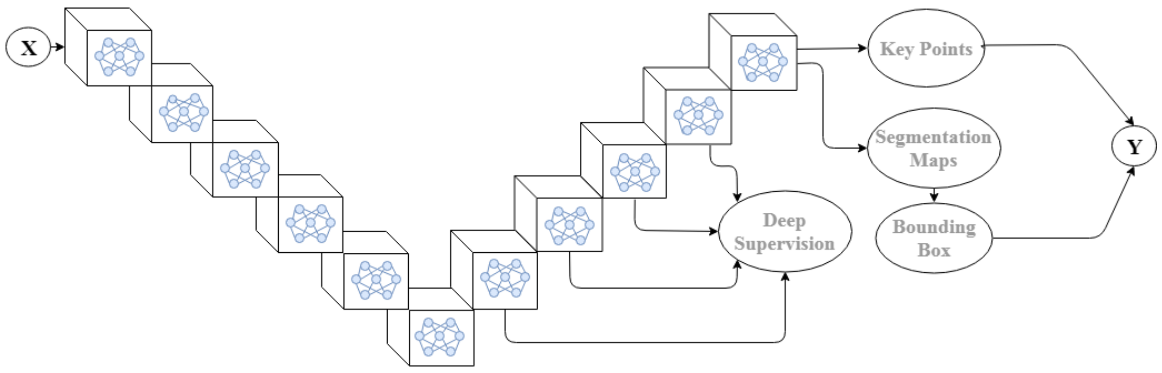


Figure 5.12. A representation of the SegBox architecture. SegBox reformulates traditional deep learning-based object detection as a segmentation problem.

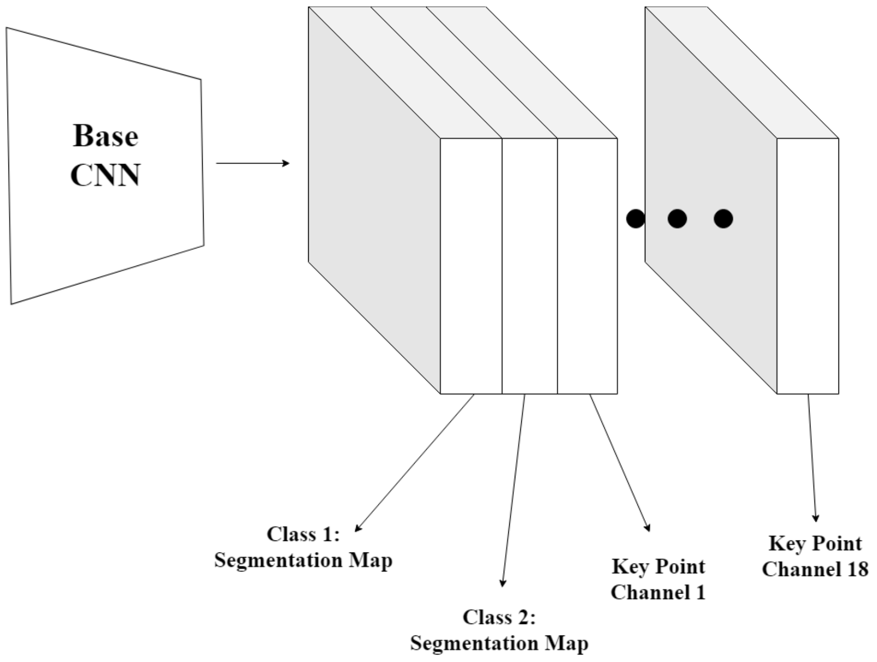


Figure 5.13. A representation of the SegBox architecture prediction module. A single output tensor is responsible for segmentation as well as key point predictions.

Optimization

The optimization step previously described was implemented on a GPU using PyTorch (Paszke et al., 2017). STLs models (Figure 5.14, 5.15) were generated using Skanect software and the nodes on the geometries that most closely corresponded to the semantic key points on each tool were manually estimated (Figures 5.4, 5.5). One weakness of the original approach was the brittleness of the solution in the presence of false positive key point predictions (Figure 5.16). It was observed that overconfident predictions of key points that were far from the true key point location would lead to poor solutions. After all, key point predictions are chosen by only taking the maximum heat map activation within a predicted bounding box, which is not a robust method.

An additional step was added to the optimization workflow to address this problem. Instead of naively taking only the maximum activation within a bounding box from each key point channel, all activations that exceeded some threshold were considered for the optimization problem. For each heat map of interest, all activations that exceeded 0.1 were filtered using NMS and then stored. This lead to a combination of potential solutions. To choose ideal key points, it was found that performing an exhaustive search over the solutions of all key point combinations did not significantly affect computation time if a loose convergence criteria was used. A modified version of the objective function (Equation 5.24) was solved using all combinations of key point predictions taken from the heat map. Solutions were stored once the optimization procedure failed to change by at least 1% over a single cycle of block coordinate descent. The confidence of the key point predictions was removed from this modified objective function:

$$L(\theta) = \frac{1}{2} \times \|(W \times Z - R \times B - T \times 1^T)\|_F^2 \tag{5.30}$$

After searching over all combinations of key points, the combination with the best solution was taken to be the true combination of key points and then fed to the full optimization step. A convergence criteria of $\Delta = 0.1\%$ was used for the full optimization step.

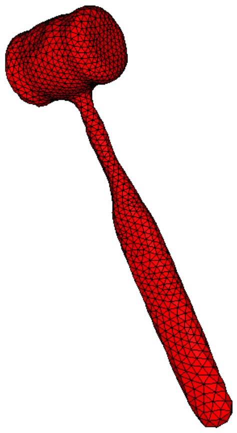


Figure 5.14. STL of the hammer.

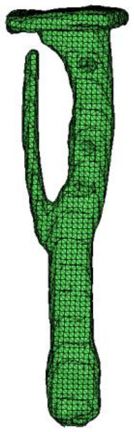


Figure 5.15. STL of the broach handle.

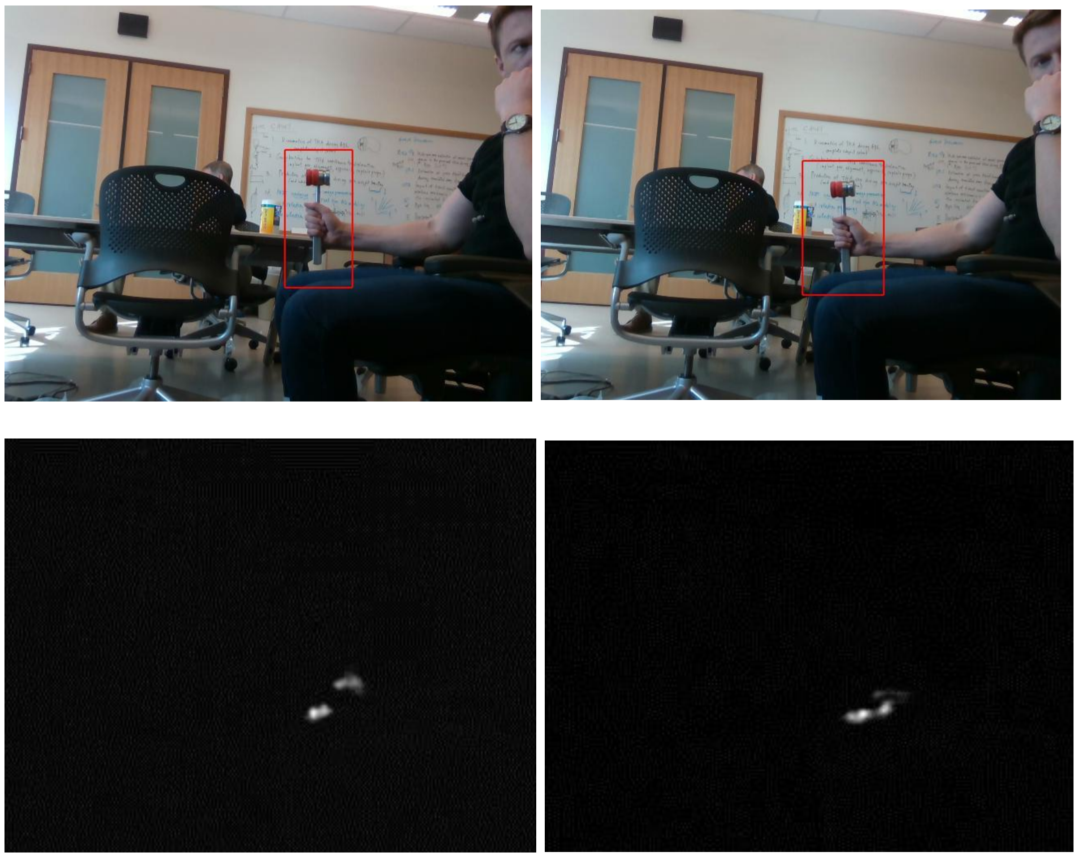


Figure 5.16. Two examples of heat map predictions with multiple proposals for a specific key point. An exhaustive search scheme is proposed to explore each key point proposal as a potential solution to the optimization problem.

Training Methods

The proposed computer vision system was developed to support the detection and pose estimation of 2 surgical tools commonly used in TKR surgeries: A hammer and a broach handle. One side of the hammer head was wrapped in red tape to supplement its appearance with more discriminative features, as well as to counteract the symmetry of the object about the handle. The broach handle was wrapped in green tape. STLs and point clouds were created for each of the tools using Skanect.

Nine key points were chosen to represent each of the tools for the key point localization step (Figures 5.3, 5.4). Synthetic data with ground truth bounding boxes and key point annotations was generated using the described pipeline. 350,000 instances were generated to train Multitool and SegBox, which was trained for 3 epochs. Multitool2 was pre-trained on these images for one epoch before being trained on the synthesized sequences of frames. 50,000 sequences of 10 frames each were generated to further train Multitool2. Training was performed by randomly sampling 5 subsequent frames from one of the sequences and then performing a training iteration. Because the frames were generated to simulate movement of the tools in RGB images, Multitool2 was trained using backpropagation through time to accumulate gradients over different time steps through the CLSTMs. All CNNs were also trained with gradient clipping by constraining gradients to the range [-1, 1].

Evaluation Methods

The computer vision system was evaluated by assessing the quality of the object detection component, as well as the accuracy of pose predictions. An Intel RealSense D-435 camera (Figure 5.17) was used to capture a video stream of 200 frames with each of the tools present at 5 frames per second (FPS) in the Experimental Biomechanics Lab (EBL) at the University of Denver. The video stream was manually annotated with bounding boxes and key point annotations for evaluating the object detection performance of the CNNs. The video stream was fed to each of the CNNs and bounding box predictions were obtained. Bounding box predictions were compared to ground truth using an Average Precision (AP) with IoU at 0.5 (Lin et al., 2014). AP was not recorded for the SegBox predictions because this metric necessitates some form of bounding box prediction confidence, and it was not obvious how to apply this to bounding boxes obtained via segmentation maps. Confusion matrices were also obtained for each model and class. For frames in which there was both a ground truth box and a predicted box, a true positive was recorded if the predicted box yielded a IoU of at least 0.5 with the ground truth. Otherwise, a false positive was recorded because the bounding box was too inaccurate. An “accuracy” metric was determined from the confusion matrices to allow for the comparison between SegBox and the Multitool CNNs.

A method for validating the accuracy of the pose predictions was developed. Each tool was fitted with three motion capture markers and placed on the ground in the Human Dynamics Lab (HDL) at the University of Denver. STLs were generated for each tool with three markers attached. A calibration wand with five markers was placed adjacent to each tool and the Intel RealSense was placed so that the calibration wand and tools were in the camera’s field of view (Figure 5.18). The Intel RealSense captured a color frame while a VICON system was used to locate the markers on the calibration wand and on the tools in the scene. Six different scenes were captured.

The 5 markers on the calibration wand were manually annotated on the image plane and the optimization problem from Equation 5.30 was solved to determine a transformation between the wand markers in the Vicon coordinate system and in the local camera coordinate system. The transformation was qualitatively verified by transforming the wand and tool markers to the local camera coordinate system and reprojecting the result onto the image plane (Figure 5.19). The transformation was applied to the tool markers to retrieve the wand markers in the camera coordinate system. The use of depth data from the Intel RealSense camera was originally explored as an alternative to using the optimization step. However, the depth data was noisy and resulted in marker coordinates that did not maintain the geometric constraints of the wand.

Each scene was processed by the computer vision system using each CNN separately to predict the pose of the tool in the scene in the local camera coordinate system. The resulting values of R and T from Equation 5.9 were applied to STL nodes that represented the marker positions on the STL to get the predicted marker positions in the camera coordinate system. The predicted position of the tool markers was compared to the markers as located by the Vicon system and transformed into the camera coordinate system. The translation error for each scene was determined based on L2 norm between predicted and ground truth marker positions. Additionally, pose estimation error was assessed from manually annotating key points in an image frame. Errors were calculated for all 6 scenes. The resulting computer vision system was thus evaluated by assessing the object detection performance of the CNNs as well as the final pose estimation produced by the full system.

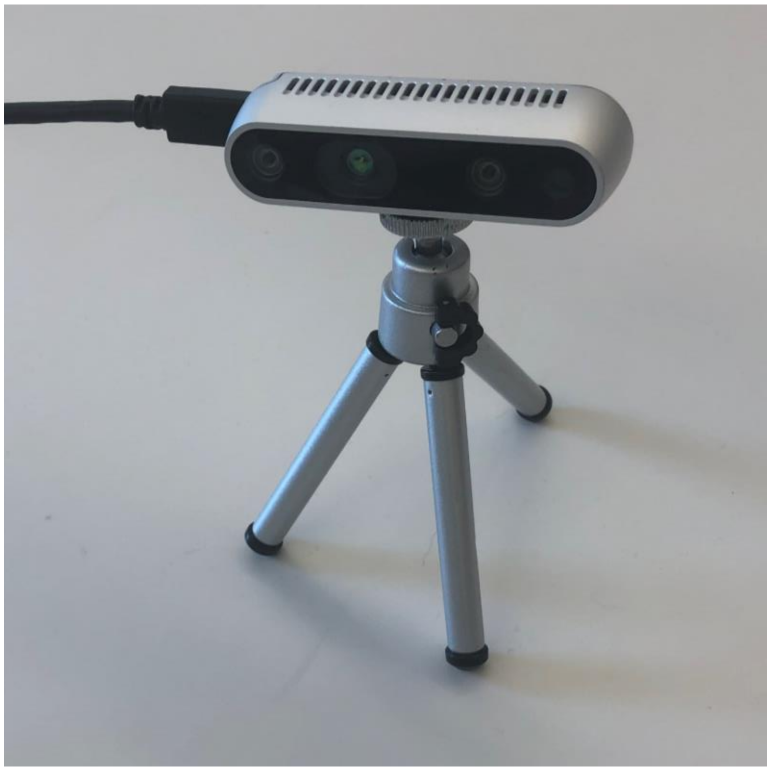


Figure 5.17. The Intel RealSense D-435 camera.

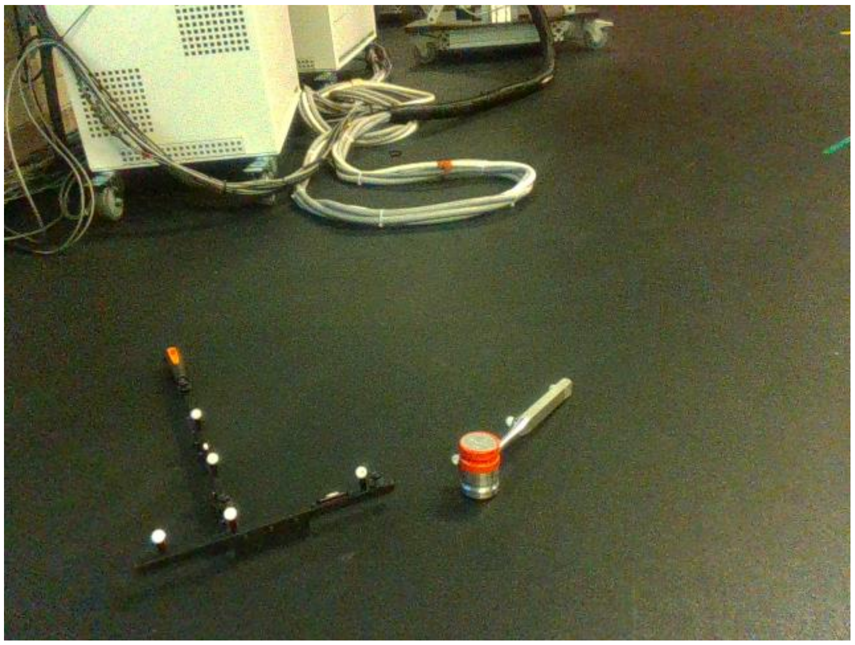


Figure 5.18. A Vicon calibration wand and the hammer with attached Vicon markers were placed in the camera's field of view. The markers on the calibration wand were located in both the local coordinate system and Vicon coordinate system. A transformation was determined in order to compare the ground truth tool marker locations to the system's prediction.

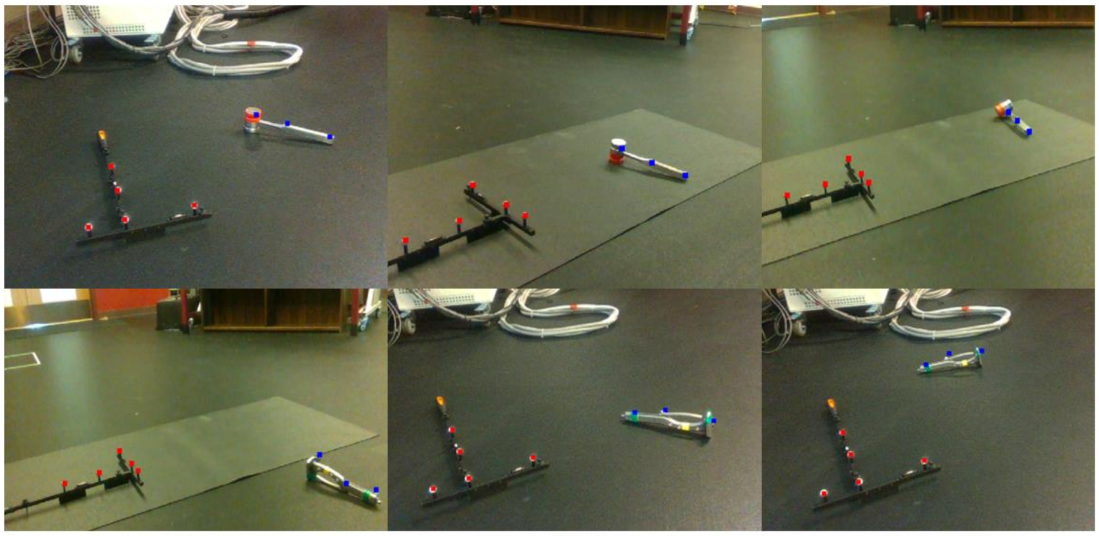


Figure 5.19. The transformation between the Vicon and camera coordinate systems was verified by applying the transformation to the markers in the Vicon coordinate space, and then reprojecting the result onto the image plane.

Results

Qualitative results for the object detection and key point predictions are presented for Multitool and Multitool2 (Figure 5.20). Segmentation maps and resulting bounding box and key point predictions are presented for SegBox as well (Figure 5.21).

AP metrics for both CNNs and tools are presented in table 5.3 and compared to numbers from literature. Multitool2 performed slightly better on average than Multitool, with AP values of 87.2 and 78.6 for the hammer and broach handle, compared to 59.1 and 89.0 for Multitool. AP values for a variety of laparoscopic tools from a recent paper are presented for comparison, which range from 17.5 to 86.3. Performance from benchmark literature is also reported to lend intuition around object detection performance on less specialized tasks.

Class-wise confusion matrices and derived accuracy metrics are presented for each CNN in Tables 5.4-5.10. Multitool2 outperformed the other two architectures on these metrics as well.

The pose estimation predictions are presented for each validation frame by projecting the position of STL nodes back onto the image plane (Figure 5.22-5.27).

Median nodal translation errors from pose estimations are presented for each frame (Table 5.12). SegBox failed to recognize 3 of the 6 validation scenes. Multitool2 failed to recognize 1 of the scenes. Medial translation error ranged from 1.4-89.6 cm. Additionally, errors resulting from manually annotated key points are presented in Table 5.13. These errors ranged from 1.1-7.5 cm.

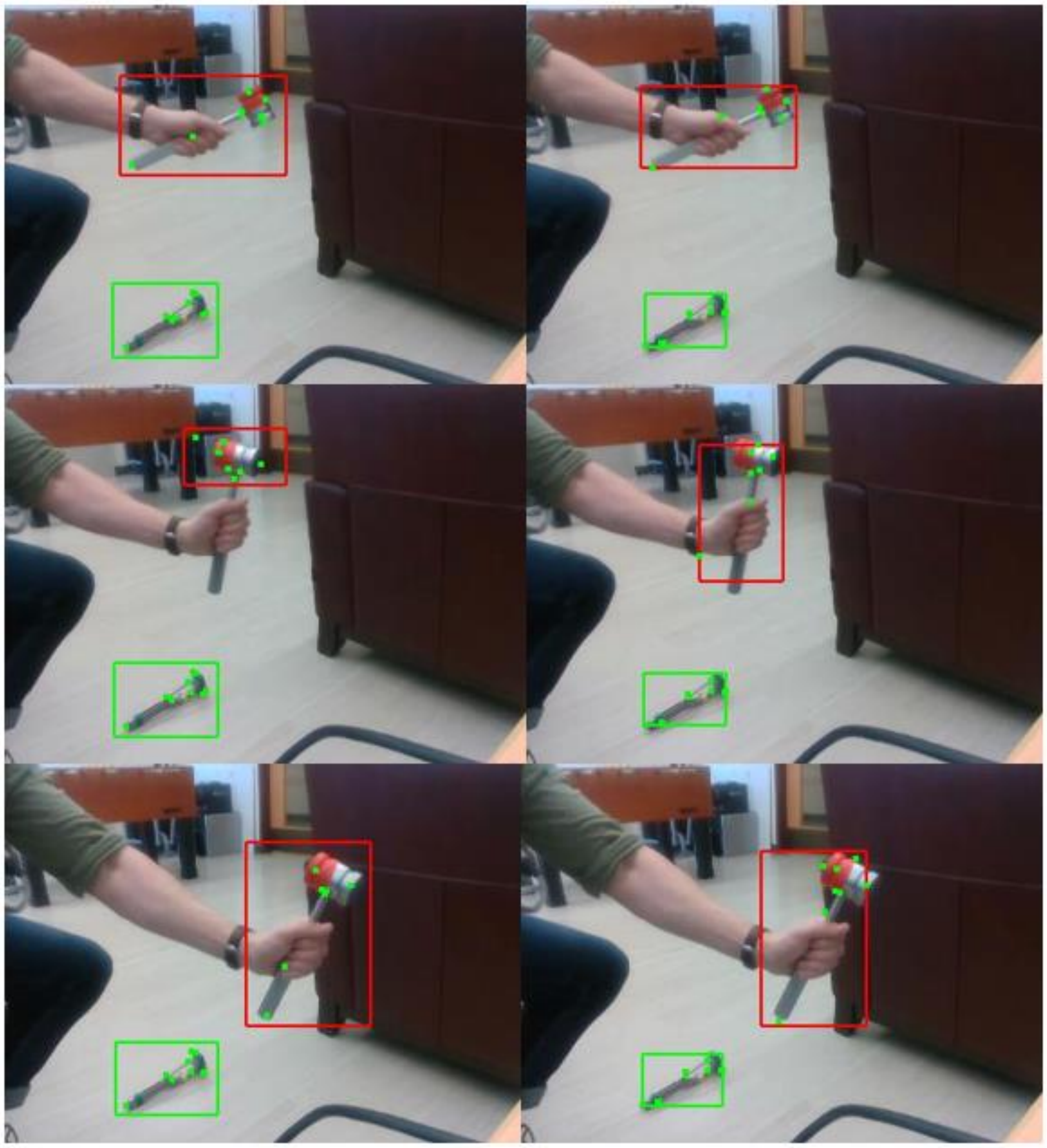


Figure 5.20. Test frame predictions from Multitool (left column) and Multitool2 (right).

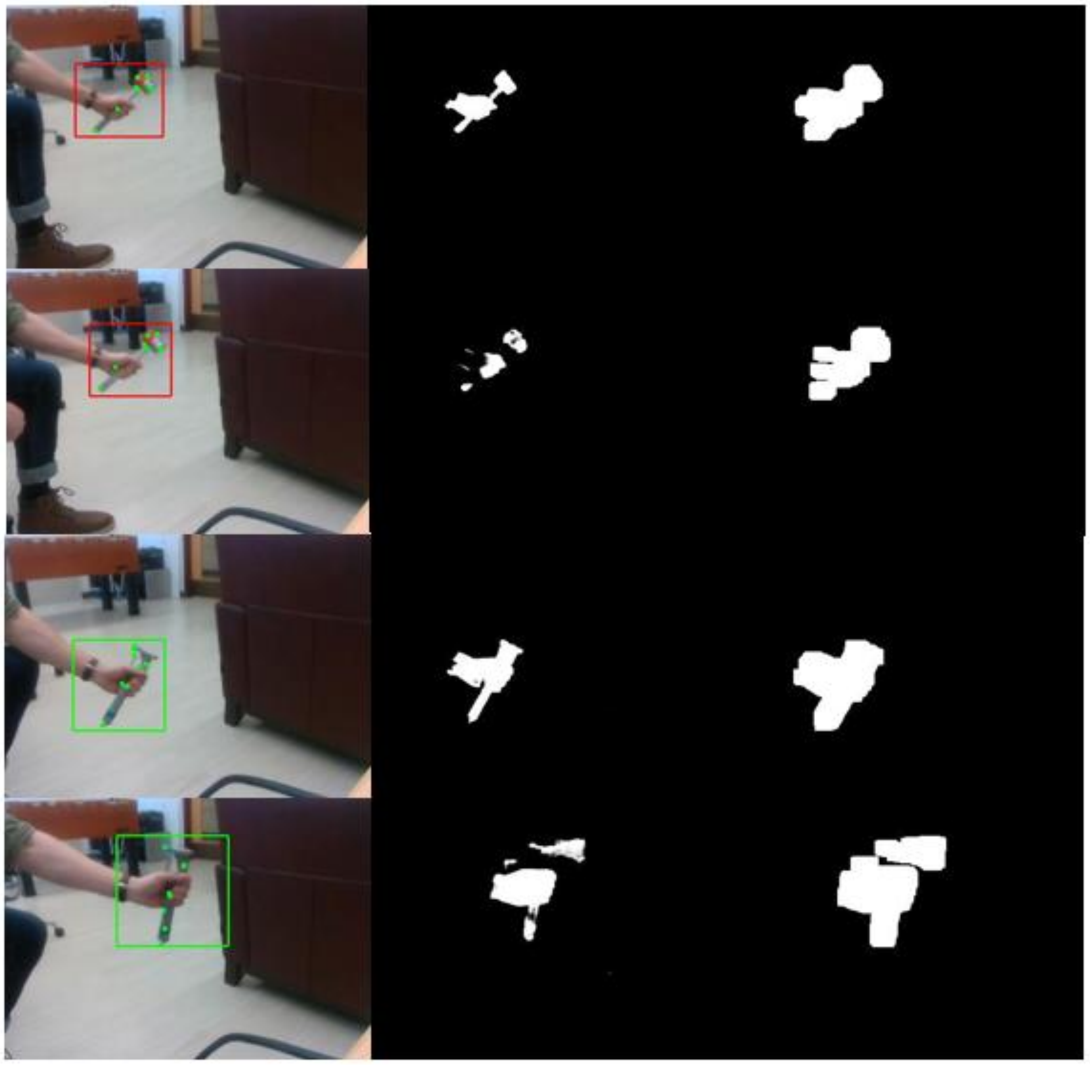


Figure 5.21. Test frame predictions (left column) from SegBox along with class-specific segmentation maps before (middle) and after (right) applying dilation.

Table 5.4. Comparison of results with object detection literature.

Method	Object	AP
(Jin et al., 2018)	Laparoscopic grasper	48.3
(Jin et al., 2018)	Laparoscopic bipolar	67.0
(Jin et al., 2018)	Laparoscopic hook	78.4
(Jin et al., 2018)	Laparoscopic scissors	67.7
(Jin et al., 2018)	Laparoscopic clippers	86.3
(Jin et al., 2018)	Laparoscopic irrigator	17.5
(Jin et al., 2018)	Laparoscopic specimen bag	76.3
SSD512 (Liu et al., 2016)	COCO test-dev2015 Mean over class (mAP)	46.5
YOLOv2 (Redmon and Farhadi, 2017)	COCO test-dev2015 Mean over class (mAP)	44.0
Faster RCNN (Ren et al., 2017)	PASCAL VOC 2012 Mean over class (mAP)	73.2
Multitool (Us)	Hammer	**59.1**
Multitool (Us)	Broach	**89.0**
Multitool2 (Us)	Hammer	**87.2**
Multitool2 (Us)	Broach	**78.6**

Table 5.5. Confusion matrix for the hammer as predicted by Multitool.

	Predicted box	No predicted box
Ground truth box	73	5
No ground truth box	30	90

Table 5.6. Confusion matrix for the broach handle as predicted by Multitool.

	Predicted box	No predicted box
Ground truth box	83	4
No ground truth box	5	106

Table 5.7. Confusion matrix for the hammer as predicted by Multitool2.

	Predicted box	No predicted box
Ground truth box	77	3
No ground truth box	7	111

Table 5.8. Confusion matrix for the broach handle as predicted by Multitool2.

	Predicted box	No predicted box
Ground truth box	75	6
No ground truth box	13	104

Table 5.9. Confusion matrix for the hammer as predicted by SegBox.

	Predicted box	No predicted box
Ground truth box	35	5
No ground truth box	48	110

Table 5.10. Confusion matrix for the broach handle as predicted by SegBox.

	Predicted box	No predicted box
Ground truth box	68	3
No ground truth box	24	103

Table 5.11. Accuracy derived from confusion matrices.

	Hammer	Broach Handle
Multitool	82%	95%
Multiool2	95%	90%
SegBox	73%	86%

Table 5.12. Median absolute nodal translation error for pose estimations (cm).

	Multitool			Multitool2			SegBox		
	Head Marker	Middle Marker	Bottom Marker	Head Marker	Middle Marker	Bottom Marker	Head Marker	Middle Marker	Bottom Marker
Scene 1	18.1	9.9	2.9	4.4	4.3	11.0	N/A	N/A	N/A
Scene 2	7.3	5.2	5.4	74.9	89.7	106.4	N/A	N/A	N/A
Scene 3	43.2	47.6	52.6	N/A	N/A	N/A	N/A	N/A	N/A
Scene 4	12.6	14.0	14.9	8.6	9.8	10.3	16.5	12.3	25.4
Scene 5	5.9	4.5	9.3	5.8	2.0	4.5	5.1	7.4	11.7
Scene 6	5.2	2.4	1.9	5.6	2.0	2.3	5.1	1.4	1.8

Table 5.13. Marker translation error from manual key point annotation (cm).

	Head Marker	Middle Marker	Bottom Marker
Scene 1	7.2	7.0	6.9
Scene 2	2.4	1.8	1.5
Scene 3	1.4	2.2	1.1
Scene 4	7.0	7.3	7.9
Scene 5	2.5	2.3	2.9
Scene 6	2.6	3.0	4.7

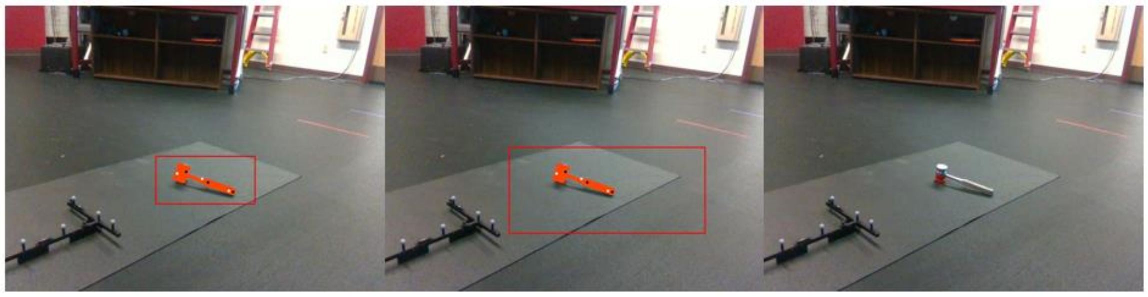


Figure 5.22. Pose estimation for scene 1 from Multitool, Multitool2, and SegBox.

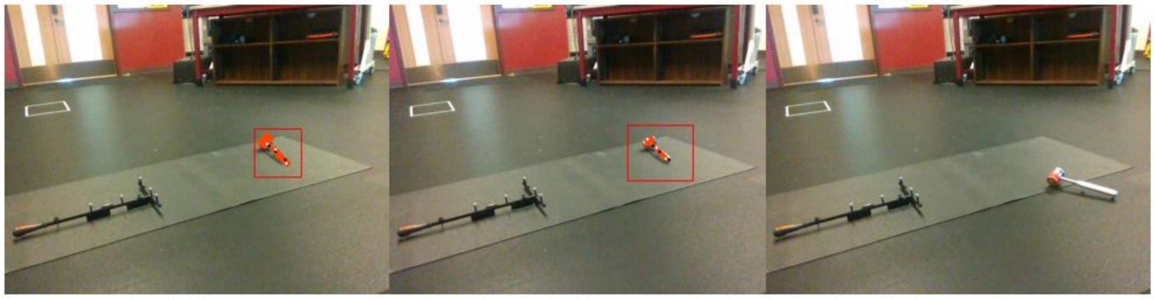


Figure 5.23. Pose estimation for scene 2 from Multitool, Multitool2, and SegBox.

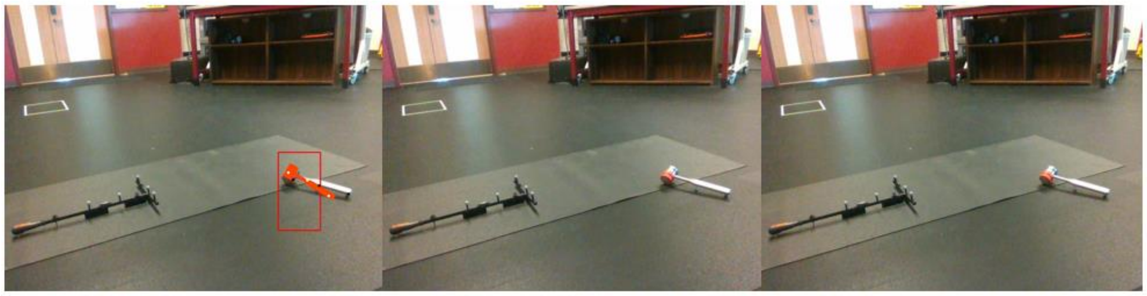


Figure 5.24. Pose estimation for scene 3 from Multitool, Multitool2, and SegBox.

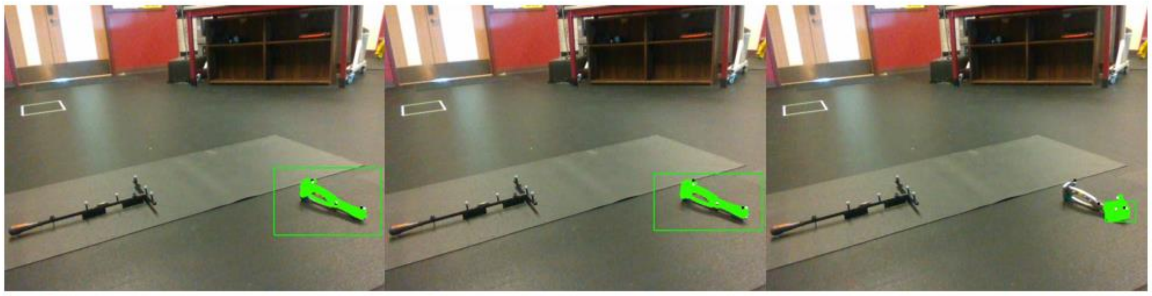


Figure 5.25. Pose estimation for frame 4 from Multitool, Multitool2, and SegBox.

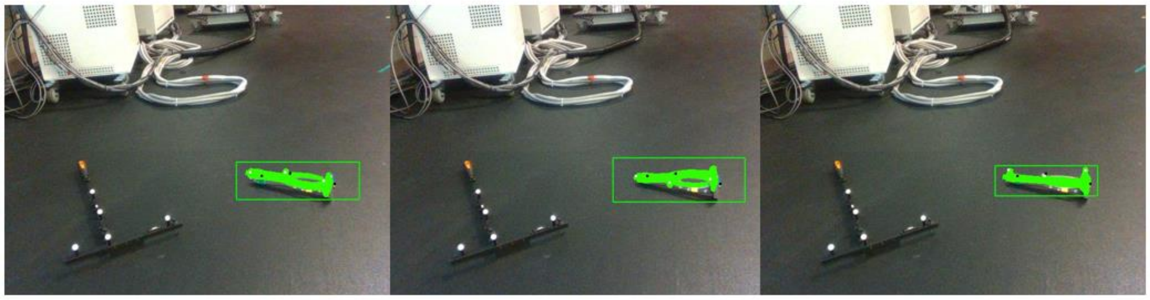


Figure 5.26. Pose estimation for frame 5 from Multitool, Multitool2, and SegBox.

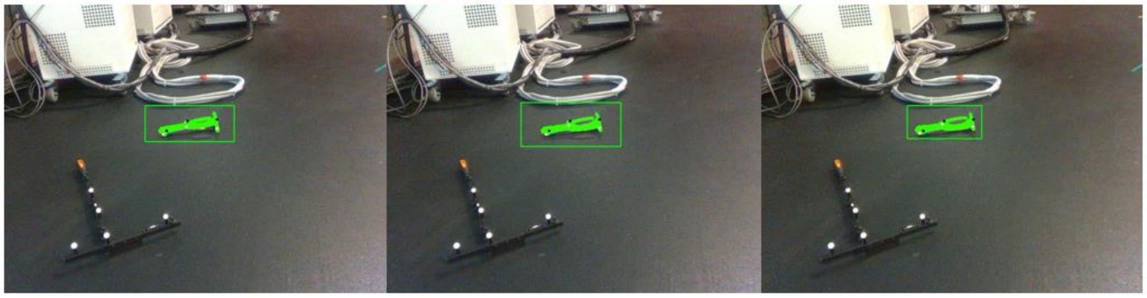


Figure 5.27. Pose estimation for frame 6 from Multitool, Multitool2, and SegBox.

Discussion

The proposed computer vision system represents one of many potential approaches for detecting and localizing surgical tools in the OR. The system requirements consist of an RGB camera with calibrated intrinsic parameters and a GPU. The use of a system that relies on only RGB images allows for efficient data synthetization for the development of the deep learning components of the system.

The AP metric was used to evaluate the object detection component of the deep learning system on a sequence of manually-annotated video frames. Because both the training and testing data sets were developed in-house, direct comparisons with literature are not conclusive. However, values from literature are presented to develop intuition about the quality of the object detection component of the system. A relevant paper by Jin et al. (2018) used CNNs for object detection of laparoscopic surgical tools in video frames captured during surgery. Average precision reported using IoU thresholds of 0.5 ranged from 17.5 to 86.3, with a mean over all classes (mAP) of 63.1. In contrast, Multitool and Multiool2 reported AP values ranging from 0.59 to 0.89. Although direct comparisons are not feasible between the two datasets, AP values reported for Multitool and Multitool2 demonstrate the effectiveness of the object detection component when compared to recent literature on surgical tool detection, with Multitool2 outperforming Multitool on average over the two classes.

The model by Jin et al. (2018) was trained on around 1,200 images, validated on 750, and tested on 500. The model was also pre-trained on ImageNet (Deng et al., 2009). The CNNs in this chapter were not pretrained. Finetuning of pretrained weights available through Keras (Chollet, 2015) was initially explored but the imported models were found to be slower than architectures built from scratch in TensorFlow. Pre-training was subsequently abandoned due to the importance of speed. Additionally, these models were trained on over 100,000 images each, not validated, and tested on 200 images. Validation data was previously used in the training of Multitool but was deemed to be unhelpful so the data was added to the training set. The noisy labels of the key point annotations on the validation data resulted in key point predictions that did not line up during training. As a result, the loss on the validation set seemed to diverge very early on during training, giving the illusion of overfitting to training data. However, predictions on the validation data continued to improve from a qualitative standpoint. This is important to keep in mind in cases of noisy data.

The mAP metric was also reported for state-of-the-art object detection literature on benchmark datasets. Everingham et al., (2010) present a dataset that contains 20 object classes, some of which are very difficult to detect. The dataset by Lin et al. (2014) contains 91 classes. In contrast, Multitool and Multitool2 were developed to support only 2 classes. This partially explains the gap in performance that is demonstrated in Table 5.3.

The data synthetization pipeline proved to be an effective tool for the training of CNNs. In order to develop a robust CNN, it is preferable to capture multiple views and lighting of the tools to be used. Additionally, it was found that if the model was not trained on images with multiple tools in a single frame, then the resulting model was not able to detect 2 tools in one frame. The training data should match the inference scenario as closely as possible. This tool would ideally be used in an OR but the training data was completely unrelated to OR scenes. A future research direction is to consider this issue either by choosing OR images as background for the data pipeline, or by capturing OR frames that contain the tools of interest. The captured data could then be leveraged in training with a semi-supervised framework to avoid manual annotation of the images. That is, a new CNN could be trained using the synthetic data but also by applying MT to the unsupervised OR footage. Bolstering the training data set to include real images is a future research direction.

The development of SegBox presented an experimental approach to the object detection problem. In addition to performing worse than the other two CNNs, another limitation of the SegBox formulation is its ad hoc inference scheme. The inference scheme assumes only a single instance per class is possible. This defeats the purpose of object detection, which ideally is agnostic to the possible number of instances in a frame. There is potential to extend the current formulation of SegBox to deal with multiple instances of a class. The use of associative embeddings is a potential solution (Newell et al., 2017).

The pose validation approach verified the efficacy of the computer vision system. Pose estimation errors stem from poor key point predictions by the CNN, disconnects between STL nodes and key point predictions, and uncertainties associated with the validation approach itself. Key points for each tool were manually annotated on the image plane by hand and the corresponding nodes on the tool geometry were also manually estimated. Therefore, there is an inherent disconnect between the key points predicted by the CNN and the actual location on the STL geometry. This source of error is accurately captured by the validation approach. An alternative method for verifying the relation between key point predictions and corresponding STL nodes may be instrumental in reducing pose estimation error. Another source of error stems from the STL geometry. The STL was generated using Skanect, which comes with its own source of error. In fact, the resulting STLs were of poor quality. This may result in differences between the STL geometry and the actual physical tool, leading to subpar optimization performances. Uncertainties in the validation approach stem from error in the transformation from the Vicon to camera coordinate system. Additional uncertainties stem from a disconnect between the Vicon markers on the tool and the STL nodes used to represent these markers. Pose estimation error was determined based on manually annotated key points in an attempt to yield intuition about the magnitude of these uncertainties

The proposed computer vision system demonstrates potential for tracking surgical tools in the OR. Based on the initial pose accuracy results, the system may be most beneficial for operations research applications, such as tracking the spatial flow of tools throughout an operation. However, by improving upon the detection and optimization components of the system, it is not inconceivable that this system could be used for applications that demand a higher precision. The current system is able to operate at around 4 FPS using a GPU, which approaches the speed required for operations research methods. Improvements in speed are a future goal.

Additional future research directions include further development of the deep learning components of the system and improvement of the validation approach to reduce uncertainties in pose estimation.

5.4 Conclusion

A computer vision system was presented for the detection and 6-DoF pose estimation of two surgical tools commonly used in TKR. The system utilized deep learning methodologies and an optimization step to place a prior known geometry in the local camera coordinate system. The object detection component of the system yielded performance on par with current literature and the pose estimation component was deemed to be accurate enough for operations research applications. However, ultimately, the object detection component still needs improvement because the success of the entire system depends on initial detection of the objects of interest. Future development of this system aims to improve the CNN components of the system and develop higher quality validation pipelines.

CHAPTER 6. CONCLUSIONS AND RECOMMENDATIONS

The work presented in this thesis explored how deep learning can be applied to the field of orthopaedics through the development of three distinct applications. In each of these applications, model performance was upper-bounded by the availability of data. Machine learning, by definition, requires data to be successful; and neural networks are particularly data-hungry. Methods for tackling the data bottleneck were presented for two of the three applications in this thesis. Semi-supervised learning algorithms were successfully applied to automatic segmentation of medical imaging to leverage unlabeled data in addition to the small quantities of labeled data. A pipeline for the generation of synthetic data was proposed for the training of CNNs used in the surgical tool tracking system. The employment of these methods led to successful models. However, for predictive modeling of patient mechanics, it was unclear how to efficiently generate synthetic labeled data; and even obtaining unlabeled data for use in semi-supervised learning frameworks would be expensive in this domain. The expense associated with obtaining labeled data for the training of these predictive models emphasizes the need for a data-driven approach for quick and accurate estimation of metrics of interest.

Thus, tackling a machine learning problem is never just about the algorithm itself. In fact, the algorithm may end up being the most trivial part of a machine learning algorithm. When facing a new machine learning problem, the first question should always focus on the nature of the data. No matter what form of algorithm is used, the performance will depend heavily on the availability of data.

Future development around automatic segmentation could focus on expanding to new anatomies and modalities and utilizing the resulting models in a practical workflow. Additionally, exploring how to get sufficient model performance from minimal labeled data would be useful.

The deep learning-based computer vision system performed decently well when trained on synthetic data but nevertheless it is hypothesized that the CNN predictions would benefit from real data as well. Real footage can easily be obtained for use in semi-supervised learning scenarios but there may be another way to cheaply obtain (real) labeled data as well. A proposed approach consists of capturing a video sequence that tracks an object of interest and annotating the first frame with ground truth bounding boxes and key points. Then, optical flow algorithms could be applied to estimate the optical flow at each subsequent frame. The original manual annotations could be iteratively updated at each frame to follow the estimated optical flow. It would be interesting to assess the quality of labeled data that is obtained in this manner.

It is currently unclear how to overcome the data bottleneck associated with the patient mechanics predictive modeling but it is hypothesized that the aggregation of significantly larger quantities of data could result in a production-ready system for the prediction of patient mechanics during ADLs.

BIBLIOGRAPHY

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., … Zheng, X. (2016). TensorFlow: A system for large-scale machine learning. *12th USENIX Conference on Operating Systems Design and Implementation*. 265-283.

Ali, A. A., Shalhoub, S. S., Cyr, A. J., Fitzpatrick, C. K., Maletsky, L. P., Rullkoetter, P. J., Shelburne, K. B. (2016). Validation of predicted patellofemoral mechanics in a finite element model of the healthy and cruciate-deficient knee. *Journal of Biomechanics*. 49(2):302-309.

Atkins, M. S., Mackiewich, B. T. (1998). Fully automatic segmentation of the brain in MRI. *IEEE Transactions on Medical Imaging*. 17(1):98-107.

Badrinarayanan, V., Kendall, A., Cipolla, R. (2017). SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 39(12):2481-2495.

Berchuck, M., Andriacchi, T. P., Bach, B. R., Reider, B. (1990). Gait adaptations by patients who have a deficient anterior cruciate ligament. *Journal of Bone and Joint Surgery*. 871-877.

Boykov, Y., Veksler, O., Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 23(11):1.

Burton, W. S., Sintini, I., Chavarria, J. M., Brownhill, J. R., Laz, P. J. (2019). Assessment of scapular morphology and bone quality with statistical models. *Computer Methods in Biomechanics and Biomedical Engineering*. 1-11.

Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A. L. (2017). DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 40(4):834-848.

Chollet, F. (2015). Keras: The Python deep learning library. *Keras.Io*.

Christ, P. F., Elshaer, M. E. A., Ettlinger, F., Tatavarty, S., Bickel, M., Bilic, P., … Menze, B. H. (2016). Automatic liver and lesion segmentation in CT using cascaded fully convolutional neural networks and 3D conditional random fields. *International Conference on Medical Image Computing and Computer-Assisted Intervention*. 415-423.

Çiçek, Ö., Abdulkadir, A., Lienkamp, S., Brox, T., Ronneberger, O. (2016). 3D U-net: Learning dense volumetric segmentation from sparse annotation. *International Conference on Medical Image Computing and Computer-Assisted Intervention.* 424-432.

Damsgaard, M., Rasmussen, J., Christensen, S. T., Surma, E., de Zee, M. (2006). Analysis of musculoskeletal systems in the AnyBody modeling system. *Simulation Modelling Practice and Theory*. 14(8):1100-1111.

Delp, S. L., Anderson, F. C., Arnold, A. S., Loan, P., Habib, A., John, C. T., Thelen, D. G. (2007). OpenSim: Open-source software to create and analyze dynamic simulations of movement. *IEEE Transactions on Biomedical Engineering*. 54(11):1940-1950.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. *IEEE Conference on Computer Vision and Pattern Recognition*. 248-255.

Du, X., Allan, M., Dore, A., Ourselin, S., Hawkes, D., Kelly, J. D., Stoyanov, D. (2016). Combined 2D and 3D tracking of surgical instruments for minimally invasive and robotic-assisted surgery. *International Journal of Computer Assisted Radiology and Surgery*. 11(6):1109-1119.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., Zisserman, A. (2010). The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*. 88(2):303-338.

Fan, Z., Chen, G., Wang, J., Liao, H. (2018). Spatial position measurement system for surgical navigation using 3-d image marker-based tracking tools with compact volume. *IEEE Transactions on Biomedical Engineering*. 65(2):378-389.

Fiorio, C., Gustedt, J. (1996). Two linear time union-find strategies for image processing. *Theoretical Computer Science*. 154(2):165-181.

Gal, Y., Ghahramani, Z. (2016). Bayesian convolutional neural networks with bernoulli approximate variational inference. *International Conference on Learning Representations.*

Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. *Proceedings of the 34th International Conference on Machine Learning.* 1243–1252.

Girshick, R. (2015). Fast R-CNN. *Proceedings of the IEEE International Conference on Computer Vision*. 1440-1448.

Goodfellow, I. J., Shlens, J., Szegedy, C. (2015). Explaining and harnessing adversarial examples. *International Conference on Learning Representations*.

Graves, A. (2012). *Supervised Sequence Labelling with Recurrent Neural Networks*.

He, K., Zhang, X., Ren, S., Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Proceedings of the IEEE International Conference on Computer Vision*. 1026-1034.

He, K., Zhang, X., Ren, S., Sun, J. (2016a). Deep residual learning for image recognition. *Conference on Computer Vision and Pattern Recognition*. 770-778.

He, K., Zhang, X., Ren, S., Sun, J. (2016b). Identity mappings in deep residual networks. *European Conference on Computer Vision*. 630-645.

He, K., Gkioxari, G., Dollar, P., Girshick, R. (2017). Mask R-CNN. *Proceedings of the IEEE International Conference on Computer Vision*. 2961-2969.

Heimann, T., Morrison, B., Styner, M., Niethammer, M., Warfield, S. K. (2010). Segmentation of knee images: A grand challenge. *MICCAI Workshop on Medical Image Analysis for the Clinic*. 207-214.

Hochreiter, S., Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*. 9(8):1735-1780.

Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K. Q. (2017). Densely connected convolutional networks. *Proceedings of the IEEE International Conference on Computer Vision*. 4700-4708.

Huang, X., Belongie, S. (2017). Arbitrary style transfer in real-time with adaptive instance normalization. *Proceedings of the IEEE International Conference on Computer Vision*. 1501-1510.

Hume, D. R., Navacchia, A., Rullkoetter, P. J., Shelburne, K. B. (2019). A lower extremity model for muscle-driven simulation of activity using explicit finite element modeling. *Journal of Biomechanics*. 84:153-160.

Ivester, J. C., Cyr, A. J., Harris, M. D., Kulis, M. J., Rullkoetter, P. J., Shelburne, K. B. (2015). A reconfigurable high-speed stereo-radiography system for sub-millimeter measurement of in vivo joint kinematics. *Journal of Medical Devices*. *9*(4):041009.

Jin, A., Yeung, S., Jopling, J., Krause, J., Azagury, D., Milstein, A., Li, F.-F. (2018). Tool detection and operative skill assessment in surgical videos using region-based convolutional neural networks. *IEEE Winter Conference on Applications of Computer Vision*. 691-699.

Kalchbrenner, N., Espeholt, L., Simonyan, K., van den Oord, A., Graves, A., Kavukcuoglu, K. (2016). Neural machine translation in linear time.

Kamnitsas, K., Ferrante, E., Parisot, S., Ledig, C., Nori, A. V., Criminisi, A., … Glocker, B. (2016). DeepMedic for brain tumor segmentation. *International Workshop on Brain Lesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*. 138-149.

Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*. 1097-1105.

Kucukelbir, A., Tran, D., Ranganath, R., Gelman, A., Blei, D. M. (2017). Automatic differentiation in machine learning: A survey. *Journal of Machine Learning*. 18:1-43.

Laina, I., Rieke, N., Rupprecht, C., Vizcaíno, J. P., Eslami, A., Tombari, F., Navab, N. (2017). Concurrent segmentation and localization for tracking of surgical instruments. *International Conference on Medical Image Computing and Computer-Assisted Intervention*. 664-672.

Laine, S., Aila, T. (2017). Temporal ensembling for semi-supervised learning. *International Conference on Learning Representations.*

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*. 86(11):2278-2324.

Lee, C.-Y., Xie, S., Gallagher, P., Zhang, Z., Tu, Z. (2015). Deeply-supervised nets. *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*. 38:562-570.

Lee, H., Hong, H., Kim, J. (2018). BCD-NET: A novel method for cartilage segmentation of knee MRI via deep segmentation networks with bone-cartilage-complex modeling. *IEEE 15th International Symposium on Biomedical Imaging*. 1538-1541.

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., … Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. *European Conference on Computer Vision*. 740-755.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., Berg, A. C. (2016). SSD: Single shot multibox detector. *European Conference on Computer Vision*. 21-37.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. *Proceedings of the Seventh IEEE International Conference on Computer Vision*. 99(2):1150-1157.

Milletari, F., Navab, N., Ahmadi, S. A. (2016). V-Net: Fully convolutional neural networks for volumetric medical image segmentation. *Proceedings - 2016 4th International Conference on 3D Vision*. 565-571.

Miyato, T., Madea, S., Koyama, M., Nakae, K., Ishii, S. (2016). Distributional smoothing by virtual adversarial examples. *International Conference on Learning Representations*.

Miyato, T., Maeda, S. I., Ishii, S., Koyama, M. (2018). Virtual adversarial training: A regularization method for supervised and semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Myers, C. A., Register, B. C., Lertwanich, P., Ejnisman, L., Pennington, W. W., Giphart, J. E., … Philippon, M. J. (2011a). Role of the acetabular labrum and the iliofemoral ligament in hip stability: An in vitro biplane fluoroscopy study. *American Journal of Sports Medicine*. 39(1):85-91.

Myers, C. A., Torry, M. R., Peterson, D. S., Shelburne, K. B., Giphart, J. E., Krong, J. P., … Steadman, J. R. (2011b). Measurements of tibiofemoral kinematics during soft and stiff drop landings using biplane fluoroscopy. *American Journal of Sports Medicine*. 39(8):1714-1723.

Myers, C. A., Torry, M. R., Shelburne, K. B., Giphart, J. E., Laprade, R. F., Woo, S. L. Y., Steadman, J. R. (2012). In vivo tibiofemoral kinematics during 4 functional tasks of increasing demand using biplane fluoroscopy. *American Journal of Sports Medicine*. 40(1):170-178.

Myers, C. A., Laz, P. J., Shelburne, K. B., Judd, D. L., Huff, D. N., Winters, J. D., … Rullkoetter, P. J. (2018). The impact of hip implant alignment on muscle and joint loading during dynamic activities. *Clinical Biomechanics*. 53:93-100.

Nevitt, M., Felson, D., Lester, G. (2006). The osteoarthritis initiative: protocol for the cohort study. *The Osteoarthritis Initiative*.

Newell, A., Huang, Z., Deng, J. (2017). Associative Embedding: End-to-End Learning for Joint Detection and Grouping. *Advances in Neural Information Processing Systems*. 2277-2287.

Ngai, V., Wimmer, M. A. (2009). Kinematic evaluation of cruciate-retaining total knee replacement patients during level walking: A comparison with the displacement-controlled ISO standard. *Journal of Biomechanics*. 42(14):2363-2368.

Noh, H., Hong, S., Han, B. (2015). Learning deconvolution network for semantic segmentation. *Proceedings of the IEEE International Conference on Computer Vision*. 1520-1528.

Paszke, A., Chanan, G., Lin, Z., Gross, S., Yang, E., Antiga, L., Devito, Z. (2017). Automatic differentiation in PyTorch. *31st Conference on Neural Information Processing Systems*.

Pavlakos, G., Zhou, X., Chan, A., Derpanis, K. G., Daniilidis, K. (2017). 6-DoF object pose from semantic keypoints. *IEEE International Conference on Robotics and Automation*. 2011-2018.

Perone, C. S., Cohen-Adad, J. (2018). Deep semi-supervised segmentation with weight-averaged consistency targets. *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*. 12-19.

Ping, W., Peng, K., Gibiansky, A., Arik, S. O., Kannan, A., Narang, S., Miller, J. (2018). Deep Voice 3: 2000-Speaker Neural Text-to-Speech. *International Conference on Learning Representations.*

Prasoon, A., Petersen, K., Igel, C., Lauze, F., Dam, E., Nielsen, M. (2013). Deep feature learning for knee cartilage segmentation using a triplanar convolutional neural network. *International Conference on Medical Image Computing and Computer-Assisted Intervention*. 246-253.

Raj, A., Vishwanathan, S., Ajani, B., Krishnan, K., Agarwal, H. (2018). Automatic knee cartilage segmentation using fully volumetric convolutional neural networks for evaluation of osteoarthritis. *IEEE 15th International Symposium on Biomedical Imaging*. 851-854.

Redmon, J., Farhadi, A. (2017). YOLO9000: Better, faster, stronger. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7263-7271.

Redmon, J., Divvala, S., Girshick, R., Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 779-788.

Ren, S., He, K., Girshick, R., Sun, J. (2017). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 91-99.

Rieke, N., Tombari, F., Navab, N. (2018). Computer vision and machine learning for surgical instrument tracking: Focus: random forest-based microsurgical tool tracking. *Computer Vision For Assistive Healthcare*. 105-126.

Rodriguez-Vila, B., Sánchez-González, P., Oropesa, I., Gomez, E. J., Pierce, D. M. (2017). Automated hexahedral meshing of knee cartilage structures–application to data from the osteoarthritis initiative. *Computer Methods in Biomechanics and Biomedical Engineering*. 20(14):1543-1553.

Ronneberger, O., Fischer, P., Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical Image Computing and Computer Assisted Intervention*. 234-241.

Rumelhart, D. E., Hinton, G. E., Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*. 323:533-536.

Sak, H., Senior, A. W., Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling. *15th Annual Conference of the International Speech Communication Association*. 338–342.

Schönemann, P. H. (1966). A generalized solution of the orthogonal procrustes problem. *Psychometrika*. 31(1):1-10.

Shelhamer, E., Long, J., Darrell, T. (2015). *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 3431-3440.

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W., WOO, W. (2015). Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. *Advances in Neural Information Processing Systems*. 802-810.

Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., … Blake, A. (2013). Real-time human pose recognition in parts from single depth images. *Communications of the Association for Computing Machinery*. 56(1):116-24.

Sintini, I., Burton, W. S., Sade, P., Chavarria, J. M., Laz, P. J. (2018). Investigating gender and ethnicity differences in proximal humeral morphology using a statistical shape model. *Journal of Orthopaedic Research*. 36(11):3043-52.

Smoger, L. M., Fitzpatrick, C. K., Clary, C. W., Cyr, A. J., Maletsky, L. P., Rullkoetter, P. J., Laz, P. J. (2015). Statistical Modeling to characterize relationships between knee anatomy and kinematics. *Journal of Orthopaedic Research*. 33(11):1620-1630.

Sudre, C. H., Li, W., Vercauteren, T., Ourselin, S., Jorge Cardoso, M. (2017). Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*. 240-248.

Sutskever, I., Hinton, G., Krizhevsky, A., Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*. 15(1):1929-1958.

Sutskever, I., Vinyals, O., Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems.* 3104–3112.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R. (2014). Intriguing properties of neural networks. *International Conference on Learning Representations*.

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., Liu, C. (2018). A survey on deep transfer learning. *International Conference on Artificial Neural Networks*. 270-279.

Tarvainen, A., Valpola, H. (2017). Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in Neural Information Processing Systems.* 1195–1204.

Vezhnevets, V., Konouchine, V. (2005). "GrowCut" - Interactive multi-label N-D image segmentation by cellular automata. *Proceedings of GraphiCon*. 1(4):150-156.

Wang, Y., Zhou, Y., Shen, W., Park, S., Fishman, E. K., Yuille, A. L. (2019). Abdominal multi-organ segmentation with organ-attention networks and statistical fusion. *Medical Image Analysis*. 55:88-102.

Wu, K., Otoo, E., Shoshani, A. (2005). Optimizing connected component labeling algorithms. *Medical Imaging 2005: Image Processing*. 1965-1977.

Deniz, C.M., Xiang, S., Hallyburton, R.S., Welbeck, A., Babb, J.S., Honig, S., Cho, K., Chang, G. (2018). Segmentation of the proximal femur from MR images using deep convolutional neural networks. *Scientific Reports*. 8(1):16485.

Yu, A., Carballido-Gamio, J., Wang, L., Lang, T. F., Su, Y., Wu, X., … Cheng, X. (2017). Spatial differences in the distribution of bone between femoral neck and trochanteric fractures. *Journal of Bone and Mineral Research*. 32(8):1672-1680.

Zhang, L., Ye, M., Chan, P. L., Yang, G. Z. (2017). Real-time surgical tool tracking and pose estimation using a hybrid cylindrical marker. *International Journal of Computer Assisted Radiology and Surgery*. 12(6):921-930.

Zhou, X., Leonardos, S., Hu, X., Daniilidis, K. (2015). 3D shape estimation from 2D landmarks: A convex relaxation approach. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4447-4455.

Zhu, L., Kolesov, I., Gao, Y., Kikinis, R., Tannenbaum, A. (2014). An effective interactive medical image segmentation method using Fast GrowCut. *International Conference on Medical Image Computing and Computer Assisted Intervention*.