Knee Bone Segmentation from MRI Images Using a Deep Learning Model

by
Ephraim Olubunmi Adeola

Submitted in partial fulfillment
of the requirements for the degree of
Doctor of Professional Studies
in Computing

at

School of Computer Science and Information Systems

Pace University

November 2020

We hereby certify that this dissertation, submitted by Ephraim O. Adeola, satisfies the dissertation requirements for the degree of *Doctor of Professional Studies in Computing* and has been approved.

Juan Shan

Dr. Juan Shan
Chairperson of Dissertation Committee

Date: December 11, 2020

Charles Tappert
Charles Tappert (Dec 16, 2020 22:21 EST)

Dr. Charles C. Tappert
Dissertation Committee Member

Date: December 11, 2020

Lixin Tao

Dr. Lixin Tao
Dissertation Committee Member

Date: December 11, 2020

Seidenberg School of Computer Science and Information Systems
Pace University

Abstract

Knee Bone Segmentation from MRI Images Using a Deep Learning Model

by
Ephraim Olubunmi Adeola

Submitted in partial fulfillment
of the requirements for the degree of
Doctor of Professional Studies
in Computing

November 2020

Abstract — This study proposed an automated segmentation method of knee bone from magnetic resonance imaging (MRI) slides, based on a Deep Learning Convolutional Neural Networks (CNN) architecture. The knee osteoarthritis (OA), oftentimes called "Degenerative Joint Disease" or "Wear and Tear" arthritis, is the most common form of over 100 types of arthritis. The effectiveness of MR images to safely provide a fast 3D visualization of the bones, cartilages, tendons and with good tissue contrast has resulted in its extensive use for the diagnosis and therapy of pathologies of the knee joint. Therefore, the need to accurately segment the knee cartilages and bones from the MR images, which is a challenging process, is paramount to the integrity of the diagnosis and monitoring of the progression of OA pathology.

This study proposed a novel method for the knee bone segmentation using MRI. The proposed method is called Simplified-UNet (S-UNet), which is an improved model based on the original U-Net model. The proposed method achieved Dice coefficient of 95.38%, which is 4.73% better than that of the state-of-the-art U-Net models. The knee MRI images were obtained from the public Osteoarthritis Initiative (OAI) dataset. The ground truth of the bone regions were manually delineated by trained personnel. The dataset contains 88 cases and was divided into training, testing, and validation sets with the ratio of 70%, 15%, and 15% respectively. The improved bone segmentation accuracy on the same dataset shows the superiority of the proposed model over the original U-Net model, indicating the potentials of future application in clinics to assist cartilage segmentation and knee OA diagnosis.

Index Terms —Knee Bone Segmentation, Magnetic Resonance Imaging (MRI), Deep Learning, U-Net.

Acknowledgements

I would like to acknowledge the roles of several individuals who were instrumental in the successful completion of my doctoral research work. Foremost, I want to express my deepest gratitude to my advisor and dissertation committee chairperson, Dr. Juan Shan, for her passion for research coupled with her qualitative critique that contributed as part of the driving force for me to give my best towards the successful completion of this dissertation work. She was magnanimously enough to give me a free remote access to her office Unix machine for the earlier part of the model training process, at a crucial time when my personal PC failed.

I would also like to express my sincere gratitude to my instructors and my dissertation committee members, Professor Charles C. Tappert and Professor Lixin Tao. Each day during this DPS studies has been a great opportunity for learning, thanks to synergetic effect from my cohorts, especially the DPS2018 and all my professors. Great appreciations go to the company where I worked for over fifteen years, Aeroflex Plainview PLC, which partly supported me financially.

To all my siblings, friends and families, including my late parents who never stop asking me the question “Bunmi, when do you think you will complete your doctorate work?”

To my lovely wife, Moronke and children, Ayobami and Temilolu who cannot wait until daddy finishes a work that has taken so much of his time from them.

Most importantly, to the Almighty God, the Giver of life, for sustenance and the grace to complete the work in His predetermined time.

Table of Contents

List of Tables

List of Figures

List of Abbreviations and Symbols

CDI	Cartilage Damage Index
CNN	Convolutional Neural Network
CV	Coefficient of Variation
DICE	Dice Similarity Coefficient
IoU	Intersection over Union
JSI	Jaccard Similarity Index
KL	Kellgreen and Lawrence Score
MRI	Magnetic Resonance Imaging
OA	Osteoarthritis
OAI	Osteoarthritis Initiative

PASCAL VOC12 - The PASCAL Visual Object Classes Challenge 2012 dataset

RoI	Region of Interest
RoIAlign	Region of Interest Alignment
RoIPool	Region of Interest Pooling
SD	Standard Deviation.
WORMS	Whole-ORgan MRI Scoring
mAP	Mean Average Precision

Chapter 1 Introduction

The largest anatomical joint in humans is the knee joint, which serves as a pivot point for the movement between the upper body at the thigh and the lower leg. Hence, the knee joint, like other biomechanical or load-bearing articular joints, is subject to degeneration due to overuse or congenital disorders, such as osteoarthritis, rheumatoid arthritis, etc.[1] Knee Osteoarthritis (OA), is the most common form of arthritis and the major cause of disability in older people [16]. CDC statistical estimates indicate that 54.4 million (or 25%) of adults in the United States of America are suffering from arthritis and that figure is projected to reach 78 million by the year 2040 [71]. Arthritis is the swelling and tenderness of one or more of the joints, such as the knee, hip, hands, feet, elbow and articular joints in general. The major symptoms of arthritis are joint pain and stiffness, which typically worsen with age. Osteoarthritis and rheumatoid arthritis the most common types of arthritis are. Bone wearing and cartilage thickness are closely related with OA development, which may be caused by both the degradation of the cartilage or due to congenital joint abnormalities. Clinical diagnosis often reveals a joint with OA showing evidences of the protective cartilage that cushions the ends of the bones wears down over time, and/or with increasing bone spurs, while a nice layer of cartilage still covers a healthy joint [19].

1.1 Significance of this Research

The magnetic resonance imaging (MRI) provides a noninvasive and fast three-dimensional visualization of the bones, cartilages, tendons for the assessment and the monitoring of the presence and progression of knee OA [19]. MRI-based estimation approaches directly visualize joint cartilages, and offer potential to more reliably define presence and severity

of osteoarthritis. The use of 3D MRI images and image processing methods allows the reconstructions of deep features, such as the Cartilage Damage Index (CDI), to find the correlation between cartilage changes and OA development. Du et al [16] used four machine learning methods (Artificial Neural Network (ANN), Support Vector Machine (SVM), Random Forest and Naïve Bayes) to predict the OA progression based on Kellgren and Lawrence (KL) grade, Joint Space Narrowing on Medial compartment (JSM) grade and Joint Space Narrowing on Lateral compartment (JSL) grade.

The prevalence and prognosis of osteoarthritis have been difficult to determine, with various clinical and radiological methods used to derive epidemiological estimates exhibiting significant heterogeneity.

Biomarkers, a shortened name for biological markers, is the use of biological measurements for the determination of signs of normal or abnormal process, or of a condition or disease. The decade-long reliance on radiographic biomarkers for the diagnosis of OA makes this research to be helpful in the analysis of knee joint biomarkers, such as the knee bone quality, strength and degradation. Subsequently, it is applicable in the assessment of the knee cartilage thickness, volume and degradation, which plays a vital role in computer-aided diagnosis (CAD) of the presences and progression of knee OA. It also find relevance in the determination of the knee joint space narrowing (JSN), the Kellgren & Lawrence (KL) grading scale [56], that are often used to define OA.

1.2 Contributions

Modern image analysis enables the accurate quantification of knee osteoarthritis (OA) bone using MRI. It has been clinically proven that three-dimensional changes in bone would be

characteristic of OA and provide a responsive measure of progression [2][5]. This work provides a deep learning automated approach for the instance segmentation of the knee femur bone, based on a Simplified-UNet model, and called S-UNet, with a model accuracy of 95.38%.

1.3 Organization of the Dissertation

Chapter 2 of this work is the literature review of the earlier traditional object detection and segmentation methods and an extensive outline of the modern image segmentation approaches. It also covers a brief of the progression of the deep learning for biomedical imaging methods and the varieties of the CNN, its extensions and the state of the art medical image segmentation methods.

Chapter 3 is the methodology of the proposed Simplified-UNet model for the segmentation of the knee femur bone. The detailed work of the state-of-the-art UNet segmentation architecture is used as the baseline. Also included is how we overcame the obstacle of low image quality, coupled with the processor capacity issue of the machine environment.

Chapter 4 is the presentation of the results and the analysis of the predicted bone images masks for the knee femur. Comparison of the results to that of the state-of-the-art method is presented in this chapter as well.

Chapter 5, which is the conclusion, presents the percentage improvement factor of the novel method over the state-of-the-art method and discusses suggestions for future work.

The appendices show a few snippets of both the standard U-Net and the proposed S-UNet codes.

Chapter 2 Background

2.1 Literature Review

Musculoskeletal diseases and articular disorders are the leading cause of global disability and affect especially the aging population [1][84]. The human knee joint is commonly affected by osteoarthritis (OA), a degenerative disease that is the primary cause of chronic disability in the United States.

The use of biomarkers of bone and cartilage quality may allow diagnosis in the early stages of OA and for the evaluation of the effectiveness of the intervention method. Several biomarkers from knee images, such as cartilage thinness, cartilage volume, bone joint space width, bone area measurement, and many more have been proven to provide new insights into osteoarthritis prediction and its progression [1] [5] [14]. Stammberger, T., et al. (1999) develop a computational approach for quantifying the three-dimensional thickness distribution of articular cartilage with MRI imaging, independent of the imaging plane its reproducibility. The reproducibility of the models shows the mean cartilage thickness, the maximal thickness and the thickness distribution in the patellar and tibia to be 1.5–3.4%, 2.1–7.9% and 2.3–6.1% respectively [100].

Marques, J. (2012) proposed the OA quantification using the analysis of the tibia trabecular bone structure of the knee MRI as biomarker for the prediction of tibia cartilage loss [74]. Du, Y., et al. (2017) exploration of the Cartilage Damage Index (CDI) biomarker on tibiofemoral compartment from OAI 3D MR imaging using four different machine learning methods, artificial neural network (ANN), support vector machine (SVM), random forest and naïve Bayes, revealed that ANN has the best performance [14] [16].

Semi-automatic segmentation proposals often require a large amount of users' input and interactions, and thus making such methods less ideal for real-time use in medical image segmentation systems. He, P. and J. Zheng (2001) approach performed radial basis function interpolation registration of a proximal femur atlas model to an hip performed radial basis function interpolation for semi-automatic piecewise registration of a proximal femur atlas model to hip MRI scan region of interest, and then used an active PDM for coarse-to-fine level segmentation [43]. Pang, J., et al. (2013) carried out the validity of a new semi-automated bone marrow lesions (BMLs) segmentation method with the hypothesize that new measure of BML size will increase with greater semi-quantitative scores and relate to articular cartilage loss on OAI datasets [83] [124].

More attentions were shifted towards automatic segmentation proposal in recent years because semiautomatic segmentation methods often use manual steps that may introduce measurement error or increase the burden on the assessor, such as manual tracing of the regions of interest or painstakingly marking areas of health bone [99]. Williams, T. G., et al. (2010) focused on the at-risked cartilage regions to improve the sensitivity of detection of cartilage loss due from 3D MR images achieved automatically using surface-based Active Appearance Models (AAMs) [113].

The state of the art for medical image segmentation, which includes the popular U-Net medical image segmentation by Ronneberger, O., et al. (2015) [93] has been greatly enhanced and used for a variety of medical segmentations, such as the knee bone and/or cartilage segmentations. Examples of the enhanced/hybrid models of UNet, such as the Seg-Net approach used by Do, N.-T., et al. (2019) for the knee bone tumor segmentation

[13]; Deep UNet, an end-to-end FCN pixel-level for sea-land segmentation that outperforms both the SegNet and UNet [61].

Yan, W., et al. (2019) proposed a hybrid of a well-trained UNet with Generated Adversarial Network (GAN) for the left ventricle (LV) segmentation task in MRI, using fast gradient sign method, termed UNet-GAN [119]. This is to overcome the domain shift challenges in medical image segmentation networks, such as CNN and UNet, by having GAN learns from UNet at the feature level that is segmentation-specific. Similarly, the UNet inspired Bounding-Box UNet (BB-UNet) for cardiac SegTHOR public dataset, incorporates priors within the skip connections to fine-tune the encoder layers and helps the neural training in order to guide the model on where to look for the organs. [20]. Ding, Y., et al. (2019 propose a framework called Stack Multi-Connection Simple Reducing Net that are stacked by our basic blocks called Simple Reducing Net for Brain Tumor Segmentation, by reducing the amount of parameters required by 80% in comparison with the original UNet[12].

Other segmentation proposal for organs include the RA-UNet, a 3D hybrid Residual Attention-aware UNet segmentation (RA-UNet) method, to precisely extract the liver volume of interests (VOI) and segment tumor lesions from the liver VOI, using the public MICCAI 2017 Liver Tumor Segmentation dataset and the 3DIRCADb dataset [51]. UNet++ for liver segmentation in abdominal CT scans, and polyp segmentation in colonoscopy videos and many more.

Classification and review of the knee bone segmentation from 1998 to 2013 by Aprovitola, A. and L. Gallo (2016) showed that several methods have been proposed in the literature

to segment the knee bones in MR images by adopting various levels of automation, from manual to fully automated methods [2]. Dalvi, R., et al. (2007), used optimized multi-contrast MR acquisition compared to conventional single-contrast approaches to achieve the sensitivity and specificity of 4.4% and 5.74% respectively for the femur bone segmentation [10].

Standard GANs normally employ the discriminator as a classifier while using the sigmoid cross entropy loss function. However, the drawback is that the loss function may lead to the vanishing gradients problem during the learning process. Mao, X., et al. (2017) proposed Least Squares Generative Adversarial Networks (LSGANs) which adopt the least squares loss function for the discriminator by to mitigate such problem, while at the same time that exhibits a more stable during the learning process [73]. Gaj et al (2020) proposed a UNet Conditional Generative Adversarial Networks (CGAN) based model that demonstrated a fully automated segmentation method with high accuracy for knee cartilage and meniscus, which trained and tested on 176, 3D double-echo steady state (DESS) knee images from the OAI [31]. Peake, E., et (2020 proposed the combination of predictions from two models, a U-Net for the segmentation of binary labels of cartilage vs background and a multi-label U-Net for specific cartilage compartments to achieve a dice of femoral 0.88, medial tibia 0.84, lateral tibia 0.88, patellar 0.85, medial meniscal 0.85 and lateral meniscal 0.90) [84]. .

Among the topmost performance in the literature are the hybrid models, such as the Knee bone and cartilage segmentation from MRI by Ambellan, F., et al. (2019) on three datasets; the OAI Imorphic dataset, OAI ZIB dataset and the SKI10 datasets, using the integration of 3D SSM-based anatomical knowledge in a CNN-based segmentation of knee MRI via a

voting scheme. A test on the OAI ZIB dataset shows a robust and accurate segmentation of even highly pathological knee structures with the DSC values of 98.6% for femur bone, 98.5% for tibia bone, 89.9% for femur cartilage, and 85.6% for tibia cartilage. An example of the multi-class ground-truth image mask for SKI10 is shown in Figure 1 [1].

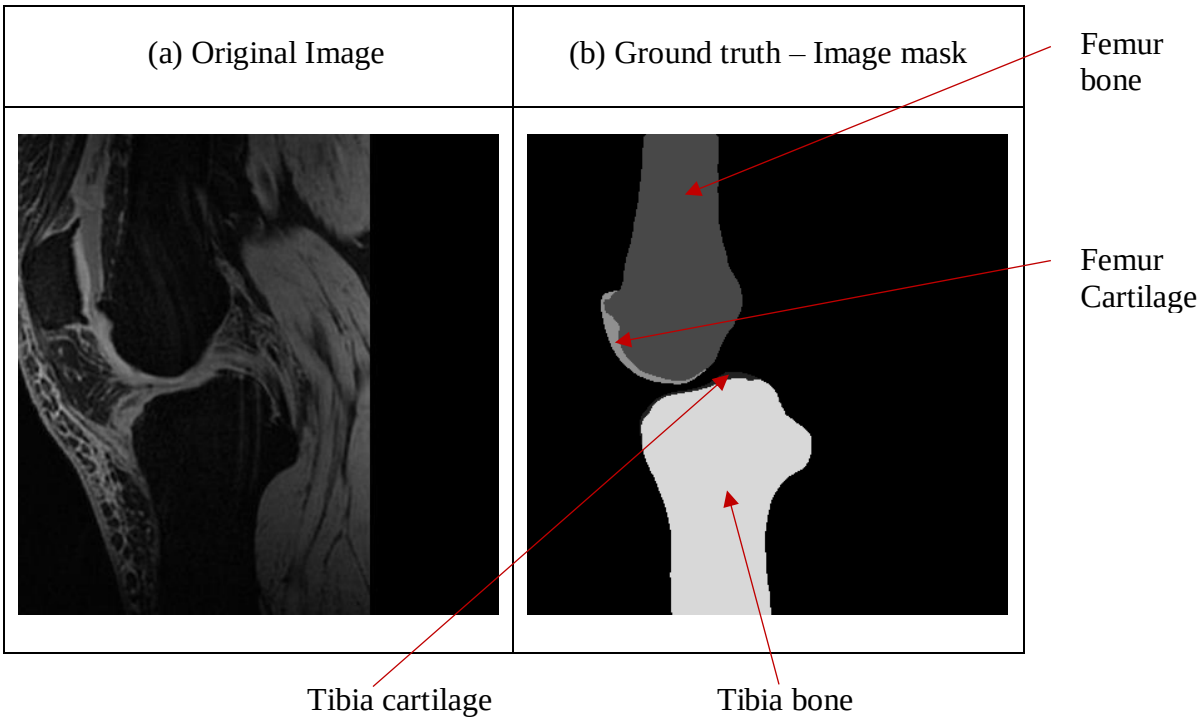


Figure 1: SKI10 Image and Image segmentation masks [1].

2.2 Osteoarthritis Detection and Scoring Criteria

The human knee is the largest joint in the body. Figure 2 shows the anatomical structure of the knee [39]. The bonehead (condyle) of these three are covered in articular cartilage, which is an extremely hard, smooth substance that function to decrease the friction forces.

The knee joint has two types of cartilage inside the joint; the articular cartilages and the menisci. The articular cartilage forms the smooth layer of the joint that covers the ends of

the femur (thighbone), tibia (shinbone) and patella (kneecap). The patella is the biggest sesamoid bone in the body. The menisci are different types of cartilage that forms a sort of shock absorber between the tibia and the femoral boneheads. Unlike the articular cartilages, menisci are not attached to any bone, but rather stands as cushion for the knee joints between the bone ends.

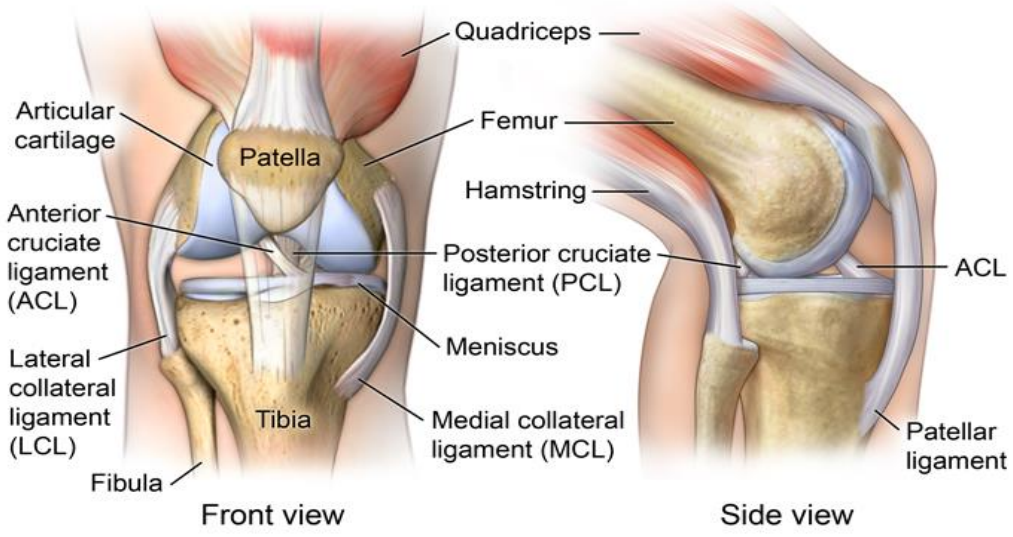


Figure 2: Anatomy Structure and Physiological Mechanism of the Knee [39]

Two main joints of the human knee are the femorotibial (between the femur and tibia) joint and the patellofemoral (between the patella and femur) joints. These allow the knee to move in three different planes, namely the sagittal, transverse, and frontal planes. Thereby offering six degrees of freedom range of motion, including flexion, extension (sagittal planes), internal, external rotation (transverse plane), varus, and valgus stress (frontal

plane). The femorotibial is the two longest lever arms of the body that function as the weight bearing, renders it susceptible to wearing and injuries.

The assessments of OA progressed from experts' manual labeling to semi-automatic and fully automatic methods. Semi-automated segmentation of the knee images are usually based on human intervention for the manual processing of training image slices, and then followed by automatic processing and visual checking of the remaining slices. Ho, N.-H., et al. (2019) used semi-supervised W-Network for Based Knee Bone Tumor [46].

Kellgren, J. and J. Lawrence (1957) radiological assessments of osteoarthritis with the use of X-rays images proved that there were both inter-observer and intra-observer differences in the interpretation of the OA results [56]. The wide disagreement with human observers and the need for effectiveness and standardization of assessment for this chronic and degenerative joint disease that now affects a significant fraction of the human population has led to numerous studies leading to proposals of semi-automatic to automatic method of assessment for OA. An example is the "Whole-Organ magnetic Resonance imaging Score (WORMS) method of knee assessment for osteoarthritis, that provides multi-feature, whole-organ assessment of the knee in OA using conventional MR images, and showed high inter-observer agreement among trained readers, that achieved intra-class correlation coefficients (ICC) values greater than 0.80 [85]. Recent studies in literature tend to use MRI for the observations of the changes in the radiological features of the knee joint for the early detection of OA, because MRIs are more versatile and are better for evaluating soft tissues, such as tendons, ligaments, and including cartilages. The studies also take into consideration the articular cartilages in addition to bone joint disorder.

While epidemiologic investigation could define OA pathologically, radiographically, or clinically, radiographic OA characterization has long been considered the reference standard. Radiographic measurements use biomarkers, such as joint space narrowing, joint line tenderness, presence of bone spurs (osteophyte) and many more. Nagaosa, Y., et al. (2002) investigated the size and direction of osteophyte in knee osteoarthritis (OA) and determined that there are associations between osteophyte size and other radiographic features [80].

Multiple radiographic measurements and scoring criteria have been devised to characterize the disease, such as the American College of Rheumatology Knee Arthroscopy Osteoarthritis Scale (ACR/KAOS) system [115], the Kellgren-Lawrence (K/L) scoring system [56]. However, the most common method is the K/L grading scheme and atlas, which has been in use for over six decades. Table 1 shows the K/L knee joint scoring criteria, based on five levels from 0 to 4, defining OA by the presence of definite/multiple osteophytes.

Table 1: Kellgreen and Lawrence five OA grading criteria for osteoarthritis.[56]

KL Score	OA Stage/Severity	Symptoms
KL 0	Healthy	Normal
KL 1	Early OA	Doubtful narrowing of joint space and possible osteophyte lipping.
KL 2	Minimal OA	Definite osteophyte and possible narrowing of joint
KL 3	Moderate OA	Multiple osteophytes, definite narrowing of joint space, some sclerosis and possible deformity of bone contour (condyle).
KL 4	Severe OA	Large osteophytes marked narrowing of joint space, severe sclerosis and definite deformity of bone contour (condyle).

There many factors in the OA scoring system as shown in Table 2. Among the characteristics is the formation, size and multiplicity of bone spurs (osteophyte). Also included is the joint space characteristics, such as the narrowing of joint cartilage associated with sclerosis of subchondral bone.

Du, Y., et al. (2018), used four machine learning methods, namely (artificial neural network (ANN), support vector machine (SVM), random forest and naïve Bayes), to predict the progression of OA by exploring the hidden biomedical information contained in the clinically used Cartilage Damage Index (CDI), to predict the change of KL, JSM, and JSL grades, respectively [16].

Table 2: Radiological features considered as evidence of osteoarthritis [56]

The formation of bone spurs (osteophytes) on the joint margins or, such as the knee joint, on the tibia spines.
Periarticular ossicles; found chiefly in relation to the distal and proximal interphalangeal joints.
Joint space characteristics.
Small pseudocystic areas with sclerotic walls situated usually in the subchondral bone.
Altered shape of the bone ends, such as the femoral medial and lateral condyle.

2.3 Traditional Methods of Object Detection

Image classification is the most fundamental unit of an object detection system in a computer vision system, in which a discrete class label is expected as an output, given an input image. Image classification assumed that a single class object is presented in the input

image. An image detection system, on the other hand, extends the classification label by localizing specific class image among multiple images, using the concept of a bounding box around the detected class image.

Earlier traditional object detection systems aim to detect images based on handcrafted features in combination with machine learning techniques, by using two-step approach. They use algorithms, which first compute a coarse region of interest (ROI) and then feed it into a detection refinement.[111]. A general examples include the use of hard and soft quantized dense image descriptors such as Scale-invariant Feature Transform (SIFT), HOG, pyramid HOG, and SSIM (self-similarity) and spatially aggregating them into histograms for super pixels

2.3.1 Scale-invariant Feature Transform (SIFT) and its Variants

Earlier traditional methods for object detection utilize feature descriptors. The Scale-invariant Feature Transform (SIFT), an image descriptor for image-based matching and recognition developed by David Lowe (1999, 2004) is among the popular ones [68]. Other SIFT improved variants, which are expected to be robust to invariant in the scaling, rotation, translation and partially to illumination changes or affine 3D projection [24][55][110]] The methodology transforms the image into a large collection of local feature vectors or descriptors, called SIFT keys. Other variants of SIFT attempt some form of enhancements through dimensionality reduction by eigenspace precomputation Principal Component Analysis (PCA) - SIFT [55], GSIFT, CSIFT, SURF, and ASIFT [117]. SIFT and its variants are implemented by searching for local peaks as an approximation of Laplacians (LoG) in a series of difference-of-Gaussian (DoG) images

after efficient construction of Gaussian pyramids and the convolution of the images with Gaussian kernel repeated.

William, F. T., & Roth, M. (1995) were able to can distinguish a small vocabulary of about 10 different hand gestures, based on a pattern recognition technique, by employing histograms of local orientation [112]. Recognitions based on depth Histograms of Oriented Gradients (HOG) descriptors have been used in bone segmentation, leukemia predictions. However, the HOG feature extraction methodology could not meet the required high accuracy in standard datasets such as PASCAL Visual Object Classes Challenge 2012 (VOC2012) [22][48]. Wang, Y.-Y., et al. (2013 used SIFT features based method for facial recognition, by adopting coarse to fine strategy with the combination of local features with a holistic classification method. This was achieved by choosing candidates based on the Euclidean distance between test samples and training sample, followed by choose a new candidates based on the SIFT features. The threshold used to count the number of well-matched pair of SIFT is the calculation of the average similarity between classes [110].

2.3.2 N-Dimensional Scale-invariant Feature Transform (N-SIFT)

SIFT key points are invariant to image rotation and scale and robust across a substantial range of affine distortion, distortion by noise, and variation in illumination. Cheung, W. and G. Hamarneh (2007), extended the stable features in two-dimensional images SIFT to n-dimensional images (N-SIFT), and evaluated the extensions in the context of medical images [7]. The N-SIFT features was applied to 3D human MRI brain scans and 4D canine computed tomography cardiac scans, to show the stability under rotation and scaling transformations.

2.4 Object Detection Evolution and Convolutional Neural Networks

2.4.1 Convolutional Neural Networks (CNN)

A class of neural networks called deep convolutional neural networks (CNNs or ConvNets for short) that accept images as inputs and are ideally suited for automatic learning, such as image and video recognition, image classification, medical image analysis, natural language processing and many more. Figure 3 [37]shows all the basic elements of a CNN, an example of which is the LeNet.

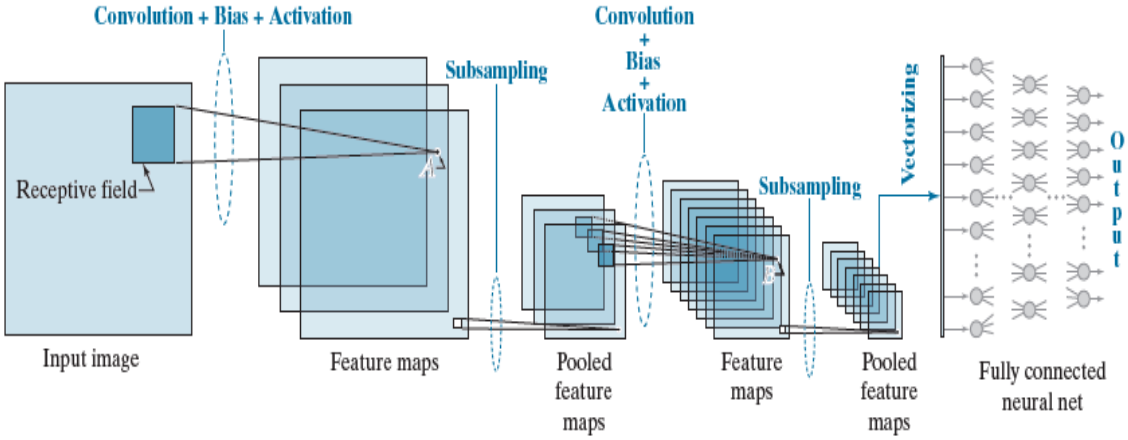


Figure 3: Basic Elements of a CNN Architecture [37]

A bias is added to the output and pass the result through an activation function to extract the feature maps. A subsample of the feature maps is polled and the process is repeated base on the depth of the CNN. The CNN final output is generated by vectorising the last pooled feature maps.

McCulloch Warren, together with Walter Pitts, created in 1943 created the first computational neural model of an artificial neuron by showing that artificial neurons based networks could, in principle, compute any arithmetic or logical function [105]. This led to

Rosenblatt's constructed of the first practical application network based on his Perceptions and the theory of brain mechanisms work [94][95]. Rosenblatt first simulated the multi-layer perceptron (MLP) on an IBM 704 computer before the practical demonstration on an electronic machine, called The Mark I Perceptron [105]. Unfortunately, Minsky, M. and S. Papert (1969), authored a Computational Geometry book on Perceptron in 1969 with widely publicized claims that to highlighting two major drawbacks of the perceptron [95]. First, that the single-layer neural networks with was the failure in solving the exclusive-or (XOR) problem. Second, the book also claimed that computers lack the sophistication to effectively handle the long runtime required by large neural networks. This caused many researchers to leave the field and nearly killed neural net research for over a decade. It turns out that Minsky and Papert's claims were trivial and later found out to be incorrect.

Fukushima's "neocognitron" [28]], a biologically-inspired hierarchical and shift-invariant model for pattern recognition could be considered an extension of the three-layered perceptron model, with which a multilayered neural network is effectively organized can be deduced. The basic idea of Convolutional Neural Networks was later introduced by Fukushima, K. (1981) in the 1981, whereby he proposed the use of improved neocognitron, a hierarchical, multilayered artificial neural network for line-detecting and bend-detecting cells to train networks for the recognition of hand-written numerals, that is unaffected by shift in position [28][29][30]. The Neocognitron was inspired by the notions of the visual cortex "simple" and "complex" cells work, proposed by Hubel, D. H. and T. N. Wiesel (1962) [49]. The capability of the improved neocognitron for pattern recognition was so high, with performance that superseded all the earlier proposal for pattern recognition, such as the Rosenblatt's Perceptions architecture and the Giebel's homogenous architecture in

1971[33] and the Fukushima's Cognitron architecture in 1975 [28]. However, the drawback of the neocognitron proposal is the lacked of a supervised training algorithm. Modern object detection system follows the principle of visual process in the human brain by using machine learning computer models to extract local features of the object and then attempt to match them with features on unknown objects. This was basic idea introduced by Fukushima in the 1980s [29] and improved by Yann LeCunn [59].

Yann LeCunn improved upon neocognitron concept in 1995, with his proposal of the convolutional neural networks deep learning architecture, that later become the popularly known LeNet, with superior performance on the full set of 60,000 modified NIST (MNIST) handwritten numeric digit recognition [33][58]. LeCun's learning algorithms for classification provided the missing algorithm for cognitron architecture, by incorporating the use stochastic gradient descent (SGD) via backpropagation to train convolutional neural networks (CNNs). Since then, CNNs have been used successfully for many recognition and segmentation tasks [58]. Many more advanced deep learning networks have evolved after the LeNet, such as the Krizhevsky AlexNet 2012, ZFNet 2013, GoogleNet 2014, ReS-UNet, DenseNet, and many more [33].

2.4.2 Region-based Convolutional Neural Networks (R-CNN)

The Region-based Convolutional Neural Networks (R-CNN), proposed by Girshick, R., et al. (2014), employs a high-capacity convolutional neural networks (CNNs) to bottom-up region proposals for the localization and segmentation of objects [35]. The combines the CNN methods with prior knowledge to propose the region of interest for the segmented image. R-CNN combines two major ideas of applying high-capacity CNN to bottom-up region proposals in order to localize and segment objects and followed by domain-specific

fine-tuning to boosts performance significantly. It was a simple and scalable detection algorithm that achieved 35 percent relative to the previous best result on VOC 2012. The more fine-tuned version of R-CNN released in 2015 improves mean average precision by more than 50 percent relative to the previous best result on VOC 2012 [36]. The overview of the object detection systems of improved R-CNN is shown in Figure 4.

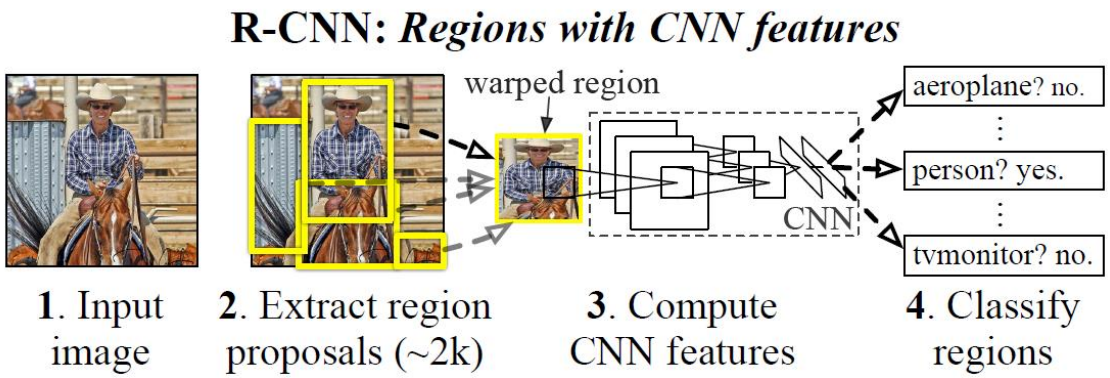


Figure 4: R-CNN Objection Detection System Overview [36]

The R-CNN takes an input image (2D) and extracts about 2000 bottom-up region proposals. It then computes the features for each proposal using a large CNN, such that each region is then classifies using class-specific linear support vector machine (SVMs).

2.4.3 Fast R-CNN

The Fast Region-based Convolutional (Fast R-CNN) for object detection, an extension of the R-CNN model by Girshick, R. (2015) employs several innovations to improve training and testing speed while also increasing detection accuracy. Fast R-CNN trains the very deep VGG16 network 9× faster than R-CNN and 213 times faster at test-time [34].

2.4.4 Faster R-CNN and its Variants

Ren, S., et al. (2017) [92] an improved and faster version of the Fast-R-CNN two years later, called Faster R-CNN, which still employs the VGG16 networks is its backbone with a CNN feature extractor to extract image features. Region proposal steps are the computational bottleneck in the Fast R-CNN, and may consumes as much running time as the detection network. Therefore, Faster R-CNN implements the proposals with a deep net to achieve an elegant and effective solution. It takes the original input image of any size and shrinks it 16x times at the fifth convolutional layer, and then applies 1x1 convolution to that feature map two times. CNN Region Proposal Network (RPN) is used to create regions of interest (RoIs), which are warp into fixed dimension, using RoI-pooling. Fully Connected Network (FCN) layers is then used to make the classifications and boundary box predictions. As one of the fastest high-accuracy detector, it operates at only 7 frames per second (FPS). Faster RCNN uses two IOU thresholds during training. Consequently, the region proposal step performed by the nearly cost-free RPNs in the Faster R-CNN improves the region proposal quality and thus the overall object detection accuracy.

Fan, X., et al. (2018) extended the Faster R-CNN architecture to proposed feature encoder can be used to capture both local ROI appearance and global context at the same time using substantially fewer parameters [23]. A Single Shot Detector (SSD) model, which discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location later, outperformed the Faster R-CNN model in the same year. SSD achieved 76.9% mAP when tested on the PASCAL VOC, COCO, and ILSVRC datasets [66].

Redmon, J. and A. Farhadi (2016) proposed a relatively fast unified Real-Time Object Detection model, called You Only Look Once (YOLO), which shares some similarities with Faster RCNN [89]. They both uses an anchor box based network structure and both uses bounding both regression. YOLO differs from Faster RCNN, in that it makes classification and bounding box regression at the same time. YOLO's major drawback is in its difficulty in detecting objects that are small and close to each other, due to only two anchor boxes in a grid predicting only one class of object. Faster RCNN on the other hand, is able to detect small objects efficiently because it has nine anchors per grid. However, Faster RCNN fails to perform real-time detection with its two-step architecture.

YOLO version 2, called YOLO9000, was introduced in 2017 and it runs at 67 FPS, achieving 76.8 mAP on VOC 2007. YOLO9000 40 FPS achieves 78.6 mAP, outperforming Faster RCNN with ResNet and SSD, while still running significantly faster [90]. Redmon and Farhadi released the YOLO version 3 in 2018, which is a little bigger than the version 2, but more accurate achieving in performance while at the same time 3.8 faster [91].

2.4.5 Mask R-CNN

He, K., et al. (2017), part of the Microsoft Research group that proposed the Faster R-CNN, extends the Faster R-CNN work at Facebook AI Research (FAIR) to propose Mask R-CNN [42]. Mask R-CNN surpasses the winner of the 2016 COCO key- point competition, and at the same time runs at 5 fps. Mask R-CNN, differs from Faster R-CNN with its incorporation of an additional branch for predicting an object mask in parallel with the existing branch for bounding box recognition. Another significant difference is that Mask R-CNN employs a more accurate feature extraction layer, called RoIAlign, to replace the

quantization-error proned RoIPool layer in Faster R-CNN, as show in Figure 5. It is important to note that Faster R-CNN's designed is limited to classification and not for segmentation purposes. Hence, it is not efficient for pixel-to-pixel alignment between network inputs and outputs, because it uses RoIPool to perform coarse spatial quantization for feature extraction. Therefore, Mask R-CNN replaces the RoIPool with a simple, quantization-free layer, called RoIAlign, which faithfully preserves exact spatial locations to overcome any misalignments.

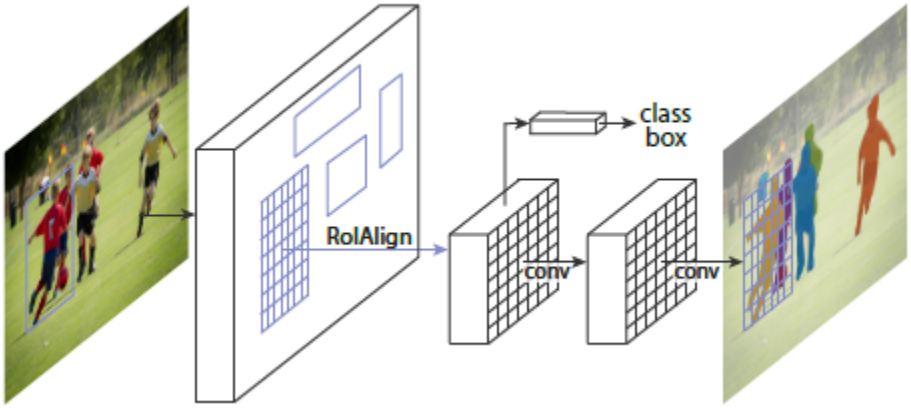


Figure 5: Mask R-CNN Objection Detection System Architecture [42]

The Mask R-CNN model is a general framework for object instance-level recognition that incorporates an object mask prediction module in parallel to bounding box segmentation module for the image class. Refined bounding box makes use of the prior knowledge to refine the output-bounding mask. The segmented output is the prediction of the mask in addition to the bounding box. The mask target is the intersection between a region of interest (RoI) and its associated ground-truth mask. A RoI is considered positive if it has intersection overt union (IoU) with a ground-truth box of at least 0.5, otherwise it considered negative. The “mask loss” is defined only on positive RoIs.

2.4.6 Other Object Detection and Segmentation with Machine Learning

Object detection task encompasses the classification and localization, usually through the concept of a bounding box, of all the objects in a given image and assigning class labels. Figure 6 shows the detection of four classes in an image input, person, car, helmet, motorcycle and background.

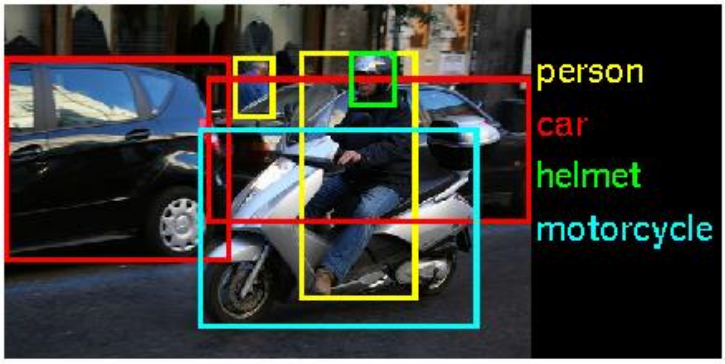


Figure 6: Object detection; Object classification with localization [48]

A semantic image segmentation or dense prediction, is a pixel level image classification, whereby the task is label each pixel of an image with a corresponding class of what is being represented. It can be seen that the semantic segmentation of figure 6(a) shown in Figure 6(b) indicates that that not all the three cars are distinguishable.

Instance segmentation is an advanced semantic segmentation whereby along with pixel level classification, each instance of a class is separately classified in the segmented output image.

Figure 7(c) is the instance segmentation of figure 7(a), showing different colors for the pixels belonging to the man, the horse and each of the three cars. Thus, instance segmentation detect instances, give categories and assigns labels that are class-aware and instance-aware.

a. Training image

b. Class segmentation showing background, car, horse and person labels.

c. Instance segmentation

Figure 7: Sample segmentation ground truth in PASCAL VOC12 dataset [22].

Recent research work has shown that deep neural networks outperform the traditional algorithms, with minimal tradeoffs, such as training time or test time.

2.5 Medical Image Segmentation Algorithms

2.5.1 General Segmentation Algorithms for Medical Images

Medical image segmentation algorithms may be classified into two general groups: pixel-based and geometry-based methods. A few examples of pixel-based segmentation methods

include edging, thresholding and region growing etc. For instance, thresholding may be viewed as a statistical-decision theory problem whose objective is to minimize the average error incurred in assigning pixels to two or more groups or classes [37]. The primary issue with most pixel-based segmentation algorithms is that they are computationally inefficient, since the algorithm normally process overlapping parts of images multiple times. However, recent segmentation algorithm takes advantages of the advancements in deep learning to addresses the shortcomings of the pixel-based segmentation with the use fully convolutional neural network (FCNN)or using encoder-decoder architectures, such as U-Net may process the entire image at the same time and output a 2-dimensional map of labels [75].

The geometry-based segmentations consist of deformable models such as active contours, snakes (parametric active contour) and active appearance models and many others [54]. Schmid, J. and N. Magnenat-Thalmann (2008) was able to carry out MRI bone segmentation using combination of physically based deformable models with shape priors. The Principal Component Analysis (PCA) of global shape variations and a Markov Random Field (MRF) of local deformations information are used to impose spatial restrictions in shapes evolution. The results gives a better efficiency with reduction in errors from 1.94 ± 1.7 to 1.62 ± 1.5 in the hipbones segmentation in comparison to the segmentation without the use of the priors [97].

Some common radiographic modalities used for medical image segmentation include the Magnetic Resonance Imaging (MRI), Computed Tomography (CT), Electron Microscopy (EM), Positron Emission Tomographic (PET), and sonographic imaging. A significant number of the recent medical image segmentation methods uses MRI datasets, especially

the publicly available datasets, such as the OAI datasets, the SKI10 dataset, the Brain Tumor Segmentation 2017 (BraTS2017) Challenge and BraTS2018 datasets [50][51]. The BraTS2018 dataset contains 285 training data with 210 high-grade glioma (HGG) patients and 75 low-grade glioma (LGG) patients, and validation data with 66 patients. For each patient, the BraTS2018 training dataset provides four MRI 3D scans (T1, T1Gd, T2, and FLAIR) with a 155×240×240 resolution and the corresponding ground truth, and the validation data, but without ground truth [50][67][77].

Zhengrong L., et al. (1994) used a statistical method for tissue classification and segmentation from multispectral brain MRI [122]. The class parameters associated with the tissue types used are the weighted intensity means, variances, correlation coefficients of the multivariate function and the number of voxels within regions of the tissue types were estimated by maximum likelihood. Recent studies show that there is an increase in the use CT-based clinical assessment that relies on quantitative measures and comprehensive analysis of multiple organs in order to identify disorders. Tong, T., et al. (2015) proposed a method that was evaluated on a database of 150 abdominal CT images and achieves a promising segmentation performance with Dice overlap values of 94.9%, 93.6%, 71.1%, and 92.5% for liver, kidneys, pancreas, and spleen, respectively [107].

Liu, F., et al. (2018) determine the feasibility of using a deep learning approach to detect cartilage lesions and degeneration within the knee joint on MR images, and achieved Dice coefficients of 95% for the femur cartilage and 81% for the tibia cartilage [63][64].

Fully Convolutional Networks (FCN) are a rich class of network models that extends the original CNN structure to enable intensive prediction, without the fully connected dense

layers. Zeng, G. and G. Zheng (2018)., extended FCN with context-guided, multi-stream for the MICCAI 2017 Grand Challenge on 6-month infant brain MR images segmentation with an outstanding average Dice Coefficient of 95.4%, 91.6% and 89.6% for cerebrospinal fluid (CSF), white matter (WM) and gray matter (GM) respectively [45], [123].

Pancreas-GAN is triple-module segmentation architecture, with a dilated convolutions auto-encoder module (DCAE) for receptive fields' enlargement, a local long short-term memory module (Local-LSTM) for contextual segmentation correlation capturing for enhancing across the boundary segmentation, and an adversarial module for constraining the spatial smoothness consistency among successive image slices [81].

Enokiya, Y., et al. (2018) came up with the hybrid combination of UNet with Wasserstein GAN for liver segmentation, was proposed, especially when training with a small data set, and the dice value was improved by about 1% to 4% to about 93% [21][119]. Segmented Adversarial Network (SegAN) proposes an end-to-end adversarial neural network for BRATS tumor image segmentation and performed more than the state-of-the-art in both dice score and precision [118]. Boundary Equilibrium Generative Adversarial Networks (BEGAN) proposed a method for controlling the trade-off between image diversity and visual quality while focusing on the image generation task [3].

Pancreas-GAN achieved a Dice similarity coefficient (DSC) of 88.72%±3.23 and the pixel wise accuracy of 95.34%±3.05. This outperformed DSC of 81.27 ±6.27% achieved through spatial aggregation of holistically nested convolutional networks (HCNs) on the three orthogonal pancreas CT volumes [96].

Numerous multi-organ image detection segmentation methods have been used to segment the liver, spleen, pancreas, and kidneys. Chu, C., et al. (2013) to segment the liver, spleen, pancreas, and kidneys with Dice similarity indices of 95.1%, 91.4%, 69.1%, and 90.1%, respectively used spatially divided probabilistic atlas [8]. Other multi-organ proposed methods include, statistical shape models (SSM) or probabilistic atlases (PA) or hybrid of two more models. [103] A hybrid of dictionary learning with sparse coding techniques and probabilistic atlases evaluated on a database of 150 abdominal CT images and achieves a promising segmentation performance with Jaccard index of 90.4%, 87.9%, 69.8%, and 91.9% for liver, kidneys, pancreas, and spleen, respectively [107].

Wang, G., et al. (2018) propose a deep learning-based framework for interactive 2D/3D multi-organ medical image segmentation. It is a unified framework for both unsupervised and supervised refinements of the initial segmentation, which uses a bounding box-based CNN for binary segmentation with additional capability of segmenting previously unseen organs. It was demonstrated with 2D images, P-Net model that trained with placenta and fetal brain images only, but it interestingly performed well on previously unseen fetal lungs and maternal kidneys. P-Net outperformed U-Net, FCN and Grabcut deep learning models with dice values of 84.57% ± 8.37, 89.44% ± 6.45, 83.59% ± 6.42 and 85.29%± 5.08 for the placenta, fetal brain images, fetal lungs and maternal kidneys respectively [108].

The leading medical image segmentation model in recent articles is the well-known U-Net-Model. In spite of the success of the U-Net model for medical image segmentation, it shows some deficiencies in the segmentation of images with some peculiarities, such as low contrast, corruption with noise etc. RA-UNet precisely extracts the liver volume of interests (VOI) and segment tumors from the liver Computed Tomography VOI [51].

Heidarkhan-Tehrani, A., et al. (2012) used Hough-Radon Transform analysis, an innovative image processing algorithm and high resolution microscopy associated with mechanical analysis, to establish a correlation between the gradient organization of cartiligous extracellular matrix (ECM) and its anisotropic biomechanical response [44].

2.5.2 Classical Knee Image Segmentation Algorithms

Earlier knee bone and/cartilage segmentation based on deformable models include, Active Contour models (ACM) like geodesic active contours (GAC) [41] [97], Statistical shape model (SSM), and active appearance model (AAM) and so on. Williams, T. G., et al. (2010) used surface-based Active Appearance Model (AAM) to construct a Volumetric Appearance Model in which the inter-image correspondence between the knee bones and the cartilages is refined to generate a more accurate segmentations. This offers the advantages of not requiring a training set of bone segmentations and enabling the identification of high stress regions, where loss of cartilage is most likely to occur. Morphological assessments of the articular cartilage in knee osteoarthritis (OA) by Eckstein, F., et al. (2006) and several other longitudinal studies have shown that the changes in cartilage volume are in the order of -4% to -6% per annum occur in most knee compartments in OA [18].

Grau, V., et al. (2004) proposed an improved Watershed Transformation algorithm with the introduction of functions based on prior information, so as to over the drawbacks of the Watershed model, whose block diagram is shown in Figure 8. The drawbacks include over segmentation, sensitivity to noise, and poor detection of thin or low signal to noise ratio structures. The results of the application of d to the segmentation of the knee cartilage and white matter/gray matter segmentation in the human brain. The algorithm validation was

carried out on 45 MRI volumes for the knee cartilage segmentation and compared with the repeated manual segmentations of seven cases to obtain a dice coefficient of 91.19%. It was also tested for the segmentation of the white matter and gray matter in the human brain resulting in a dice coefficient of 94.56% and 89.02% respectively.

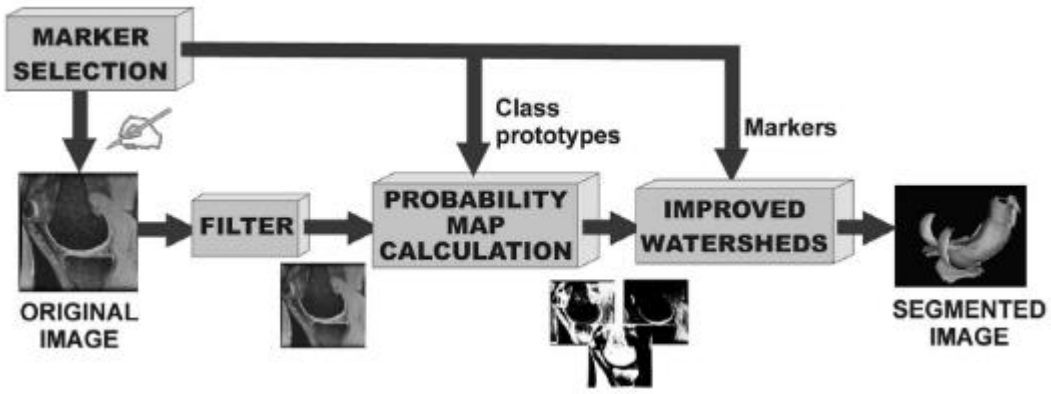


Figure 8: Knee Cartilage Segmentation with Improved Watershed Algorithm [40].

Kitney, R., et al. (1998) uses Medical Image Display and Analysis System (MIDAS), a UNIX workstation-based program under development at Imperial College, for the automated segmentation and visualization methods for MR images of the knee joint in arthritis [57]. The algorithm was used in locating the boundaries of the femur, tibia, patella, menisci and cruciate ligaments.

2.5.2.1 *Edge Detection and Thresholding Models*

The most popular features for guiding image segmentation are the edges, such as intensity edges, texture edges, parametric edges, and watersheds. Popular edge detectors, such as the Canny, Sobel, Prewitt, and Roberts operators, provide finite difference approximations of the gradient [37]. For instance, Canny edge detector aim to locate the blobs by searching for the local gradient maxima with the use the derivative of Gaussian (DOG) filter, while

Sobel operator uses median filter. However, neither Canny nor Sobel operators cannot remove salt and pepper noise very well. Image edge and thresholding approach uses simple and less computationally expensive intensity-based algorithms attempts to determine an intensity value, called the intensity threshold, which separates the desired classes. The output of the segmentation procedure is the results achieved by grouping all pixels with intensity greater than the threshold into one class, and all other pixels into another class. For instance, a single could identify knee bone image or multiple thresholds depending on its histogram pattern [101]. While it is adopted to obtain a fast detection of the knee bones, it drawback is that it cannot be used for quantitative purposes due to the non-uniform acquisition of an MR image [86].

2.5.2.2 Graph Based Models

Grade based models, such as Graph cuts formulates image segmentation into an optimization of energy cost function problem. The solution requires minimizing an energy function, which is known to be NP-hard [78]. A major drawback of classical graph cuts algorithm is that it does not support multiclass segmentation. Shim, H., et al. (2009) implemented semi-automatic method, using graph cut technique to segment femur, tibia and patella, to overcome the highly laborious manual segmentation of knee joint structures by boundary delineation that was subject to user-variation [99]. This is a two-step approach whereby a seed is place in any of sagittal, coronal, and axial planes followed by the computation of segmentation, based on a graph-cuts algorithm where the optimal segmentation is the one that minimizes a cost function. The model objective was to minimize the cost function, which incorporated the seeds placed by the user and the properties of both regions and boundaries of the segmented regions, using graph cuts

algorithm. The drawback is that this a semi-automated method requires significant human input, whicaccount for more than two-thirds (14min/21min) of the processing time, that was spent by the user for the placement of seeds.

Tianhu, L., & Sewchand, W. (1992) proposed an unsupervised stochastic model-based image segmentation technique for X-ray CT image that utilizes the finite normal mixture distribution and the underlying Gaussian random field (GRF) as the stochastic image model. The parameters of the stochastic model-based are estimated by expectation-maximization and classification-maximization algorithms, while the Image segmentation was performed by Bayesian classifier [106]. Wu, D., et al (2014) proposed a fully automated segmentation approach for the joint bones (femur, tibia, fibula, and patella), using a model-based marginal space learning framework with refinement using graph cut based on the shape priors derived from the initial segmentation [115] [116].

2.5.2.3 Deformable Models and Shape Priors

Deformable models have proven to achieve good segmentation results and are flexible in their approach for segmenting structures in medical images. The object model deforms iteratively until it matches the information present in the image to be segmented. However, the primary drawback of deformable models is that they are sensitive to initialization. Therefore, accurate and robust results often require initialization close to the true object in the image.

Guo, Y., et al. (2011) proposed a distribution-based active contour model, by measuring the Bhattacharyya distance between probability distributions of the object and background

along with the evolution of geodesic active contour (GAC) model. Guo's proposal is to overcome the flaws of boundary leaking and expensive evolving time in GAC models [41].

Statistical shape model (SSM) was used by (Fripp et al. 2007) [25] [26] [26] and the Shape model Fitting by Fripp, J., et al., (2009) [27] to achieve a Dice Similarity Coefficient of 0.75 and 0.77 for the medial and lateral meniscus respectively. The result was a significant improvement for a fully automatic segmentation then.

2.5.2.4 Classifier and Clustering Methods

Classifier methods employs the use pattern recognition techniques through a process of partitioning the image into feature space, using a feature like the image intensities, that are derived from the corresponding known labels of the image data [2]. Classifiers are also known as supervised methods because they require training data along with the corresponding manually segmented reference images and then used for automatically segmenting new data. A simple classifier is the nearest-neighbor classifier that classify each pixel or voxel in the same class as the training datum with the closest intensity. The k-nearest-neighbor (kNN) classifier is a generalization of this approach, where the pixel is classified according to the majority vote of the k closest training data.

A non-parametric classifier, among which is the kNN classifier, makes no underlying assumption about the statistical structure of the data. A parametric classifier on the other hand, such as the maximum likelihood (ML) or Bayes classifier, assumes that the pixel intensities are independent samples from a mixture of probability distributions, usually Gaussian. Long, J., et al. (2015) uses ILSVRC classifiers into fully convolutional networks (FCNs) to combine coarse, high layer information with fine, low layer information. This

was achieved by adding skips between layers to fuse coarse, semantic and local, appearance information [67].

Clustering, or unsupervised methods make use of algorithms that essentially perform the same function as classifiers, but without the use of training data. Therefore, clustering methods train themselves using the available data, by iterating between segmenting the image and characterizing the properties of the each class [9] [17]. Kashyap, S., et al. (2017) used graph based layered optimal graph image segmentation of multiple objects and surfaces (LOGISMOS) framework to prepare data for training hierarchical RF classifier and to assess the benefits over a longitudinal study period [53].

2.5.3 Recent Knee Image Segmentation Algorithms

Figure 9 [32] is the summary chart of review of recent knee joint segmentation proposals that often covers both the bone and cartilage segmentation.

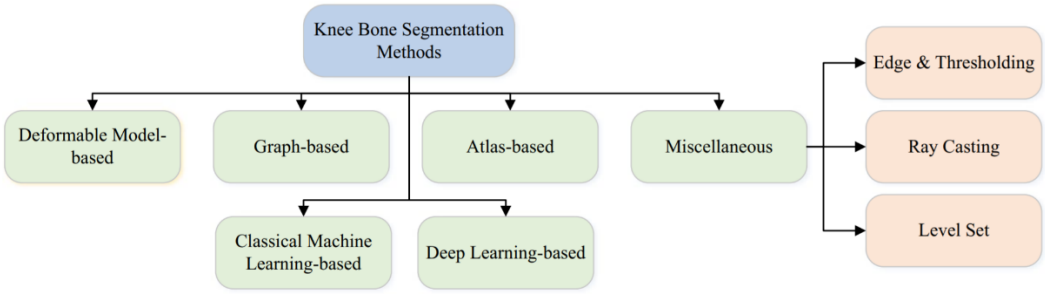


Figure 9: Recent Knee Image Segmentation algorithms [32]

Most recent proposals indicate the recognition of the knee osteoarthritis as a “whole joint” disease [70]. They also highlights on the diagnostic values of deep learning in emerging knee osteoarthritis research [32].

The block diagram in Figure 10(a) and Figure 10(b) compares the classical machine learning methods to the recent knee image segmentation algorithm, that most used machine-learning approaches. The deep learning architecture generally uses multiple hidden layers for the hierarchical feature extractions with output classification and segmentation layer. Most architecture of the deep learning are CNN [59]. The value of each node is estimated by parameterizing weights during model training, through convolutional filters while the objective function is optimized via backpropagation.

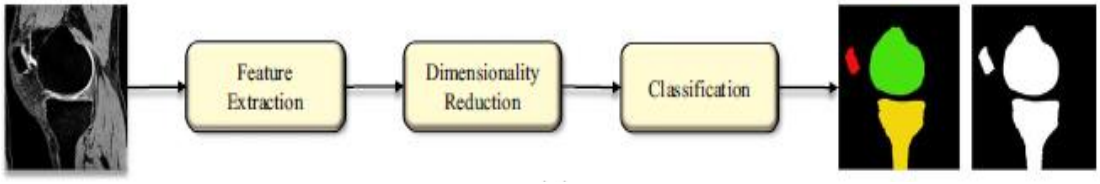


Figure 10(a) Classical Machine Learning [32]

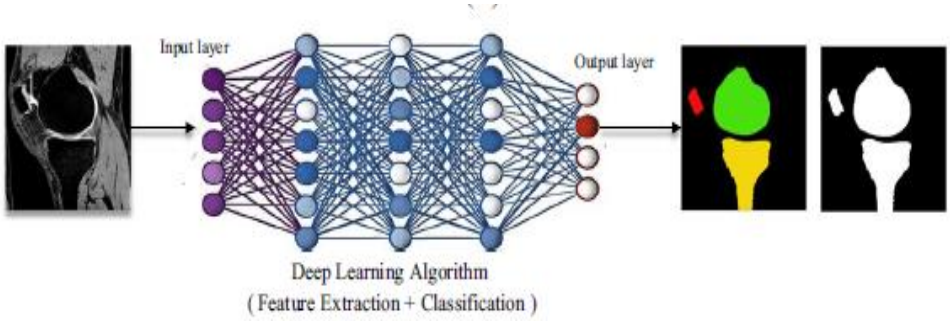


Figure 10(b) Deep Learning Architecture with Multiple Hidden Layers [32]

2.5.3.1 Active Contour Models with Convolutional Neural Networks (CNN)

Active contours model (ACM), considered the benchmark for deformable models, is defined by a collection of points along the curve, which is governed by either parametric or geometric information available in the image [54]. The active contour AC models could mostly be grouped into four groups: edge-based and region-based or a hybrid AC that exploits the combination of two or the entire three AC methods [41].

A few of Active Contour inspired medical image segmentation methods proposed in literature include the following: the Active Appearance Models (AAMs), [5][114]; the Active Shape Models" (ASMs) used for tibia segmentation from ultrasound CT images [43], [54]; the combination of the shape knowledge of Statistical Shape Models (SSM) with the strong classification capability of Convolutional Neural Networks (CNN) proposed by Ambellan, F., et al. (20190 [1]. Active shape models (ASM) use both the statistical analysis of the shape variation and gray level information as a priori knowledge to constrain the deformation of 3D objects [101].

Johnson, J. M. (2013) doctoral dissertation work combined the active contour knowledge with powerful feature learning of UNet to carried out the analysis, segmentation and prediction of Knee Cartilage using Statistical Shape Model (SSM) for the knee hard (bone) and soft (cartilage and ligaments) for healthy and K-L graded datasets [52]. He generated accurate maps of healthy and pathological cartilage maps to show the difference between genders and wear patterns and achieved mean DSC for all cases of 0.844 for the femur and 0.801 for the tibia cartilages.

2.5.3.2 Atlas or Multi-Atlas Based Segmentation

Atlas-based segmentation exploits the use of priori knowledge, such as shape, object class, priori probabilities and topological details of target object, from labeled training images for segmentation. Tamez-Pena, J. G., et al. (2012) used multi-atlas segmentation process for the fully automated quantitation of OA knee MRI biomarkers and achieved a dice similarity coefficient (DSC) of 0.88 for femur bones and 0.84 for the tibia bones for the OAI datasets [102]. Shan, L., et al. (2014) used a multi-atlas based bone segmentation to guide the registration of cartilage atlas, but still resulted in a major drawback of high computational cost, which is the typical disadvantage of all multi-atlas-based methods [98].

2.5.3.3 U-Net Model and Its Extensions

U-Net, a deep learning architecture built upon fully convolutional network, was the state of the art for biomedical image segmentation at the beginning of this research in 2015 was originally introduced by Ronneberger, O., et al. (2015) [93]. U-Net architecture was modified and extended to work with fewer training images to yield more precise segmentations, without compromising the accuracies.

Norman, B., et al. (2018) used 2D U-Net Convolutional Neural Networks for automated cartilage and meniscus segmentation of knee MRI to extract relaxation times and morphologic characterization and values that can be used in the monitoring and diagnosis of OA [82]. The model, which averaged 5 seconds to generate the automatic segmentations, produced strong Dice coefficients, ranging between 0.770 and 0.878 in the cartilage compartments to 0.809 and 0.753 for the lateral meniscus and medial meniscus,

respectively. Liu, F. (2019) came up with cycle-consistent generative adversarial network (CycleGAN) model termed SUSAN, an efficient CNN-based fully automated segmentation of unannotated knee bone and cartilage image method, with the added advantage of eliminating the need for large amounts of annotated training data [62].

Gaj et al (2020) proposal a hybrid U-Net mode that focused on overcoming the pixel-wise mapping based objective functions challenges to capture cartilage features during the training of the network, using a UNet Conditional Generative Adversarial Networks (CGAN) [31]. It is a fully automated segmentation method with high accuracy for knee cartilage and meniscus with MRI data from the OAI datasets.

Another extension of the U-Net was proposed by Raj, A., et al. (2018), named µ-Net model, for automatic knee cartilage segmentation using fully volumetric convolutional neural networks for evaluation of osteoarthritis. The 100 3D MR volumes from SKI10 datasets and 176 knee 3D MR volumes from 88 patients were used to validate the µ-Net model in the segmentation of the knee cartilage.

The dice loss is adapted to accommodate for multiple labels to generate a multi-class of 3 and 7 for SKI10 dataset and OAI dataset respectively. The block diagram of the µ-Net model in Figure 11, shows that shot-skip and long-skip are added to the architecture, in comparison to the standard U-Net.

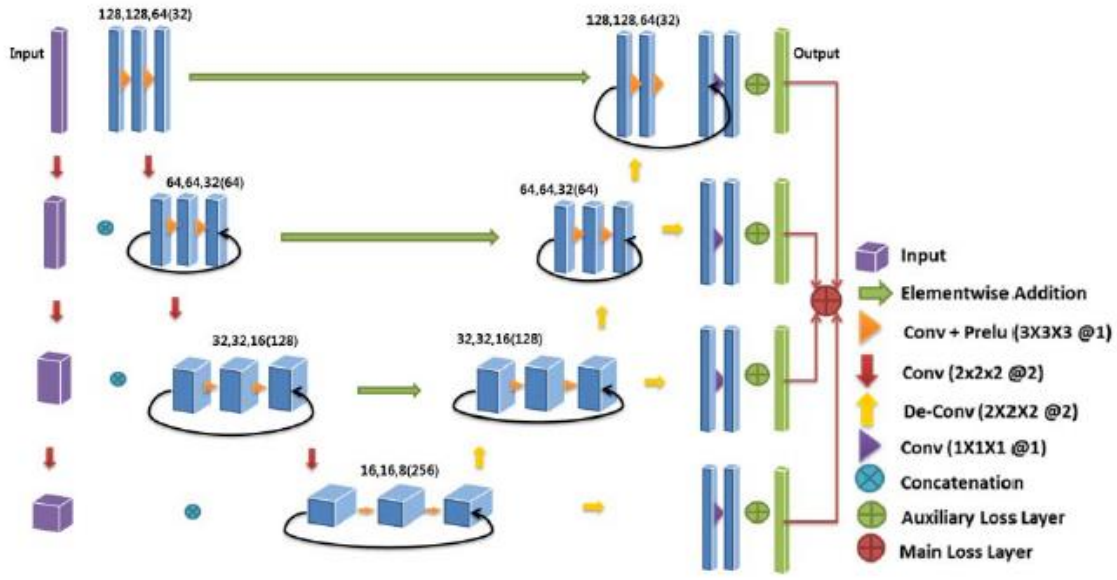


Figure 11: Architecture of μ-Net model used for the knee cartilage segmentation [88]

The short-skip connections are added within each resolution of the analysis and synthesis path for the mitigation of the problem of vanishing gradients, while the long-skip connections, added across the layers of equal resolution from the analysis to synthesis path enabling high-resolution feature flow in the network.

The UNet CGAN model achieved the average Dice coefficients of 0.84, 0.91, 0.89 and 0.87 for patellar cartilage, lateral tibia cartilage, lateral meniscus medial meniscus respectively. SUSAN bone segmentation accuracy of achieved a dice coefficient of 0.94 and 0.95 for the femur and tibia respectively [62], [63], [64]. Prasoon (2013a) proposed Triplanar CNN using voxel classification integrating three 2D convolutional neural

networks method for the segmentation of tibia cartilage in low field knee MRI scans and tested it on 114 unseen scans resulting into disc value of 82.49% [87].

Many of the UNet variants, such as UNet++ proposed by Zhou (2018), took advantage of re-designing the UNet skip pathways, purposely to aim at reducing the semantic gap between the feature maps of the encoder and decoder sub-networks [125][126].

2.5.3.4 Hybrid Segmentation Methods

Hybrid segmentation approach, whereby two or more algorithms are combined, such as the combination of a priori knowledge for region network to target the region of interest (ROI) with another deep learning models. A recent hybrid segmentation model by Ambellan (2019), combines a priori knowledge of anatomical shape with Convolutional Neural Networks (CNNs) by incorporating 3D Statistical Shape Models (SSMs) as well as 2D and 3D CNNs to achieve a robust and accurate segmentation of the knee cartilage and bones [52]. Ho, N.-H., et al. (2019), proposed the use of cascaded U-Nets into two regenerative semi-supervised UNet networks to form a bidirectional W-network (RSS-BW), for the tumor state prediction at the knee bone from X-ray images [46]. Memari, N. and M. Moghbel (2020) propose a hybrid segmentation model by combining random walker algorithm with integrated priors as the density estimation of the knee cartilage region, generated from the anatomical regions estimated by the deep learning networks [76].

Wang, Q., et al. (2013) effectively exploit the combinations of the semantic context information in the knee joint with distance features and the dense anatomical landmarks on the bone surface, the spatial constraints between cartilages and bones for improved the segmentation [109]. Figure 12 shows the Multi-organ FCN based Y-Net model, proposed

by Tan, C., et al. (2018) [104]. This is the Deep Multi-Task and Task-Specific Feature Learning Network for Robust Shape Preserved Organ Segmentation used for MR femur bone and CT kidney modalities.

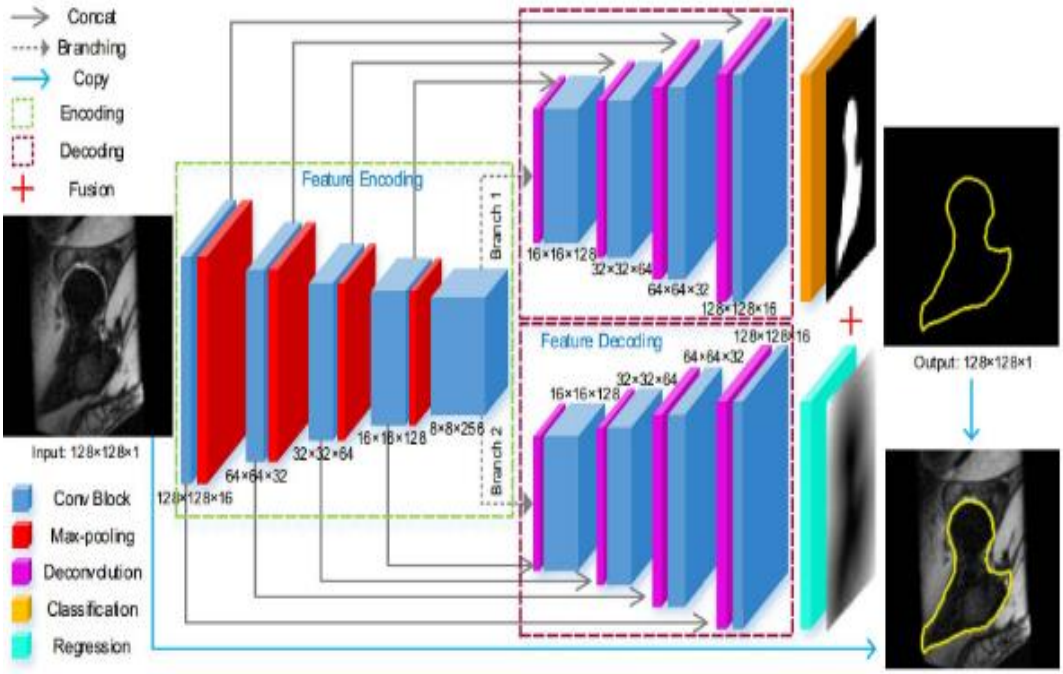


Figure 12: Multi-organ FCN based Y-Net model [104].

Figure 13 shows the Boundary Equilibrium GAN (BEGAN) model , which enforces the boundary equilibrium and paired with a loss derived from the Wasserstein distance for training auto-encoder based Generative Adversarial Networks, while simultaneously controlling the trade-off between image diversity and visual quality [3][60]. Generative Adversarial Networks (GANs) are generative models that use supervised learning to approximate an intractable cost function, such as the KL divergence for maximum likelihood estimation. This is analogous to the Boltzmann machines that use Markov chains to approximate their cost, while the variational autoencoders (VAEs) uses the variational lower bound to approximate their cost [38].

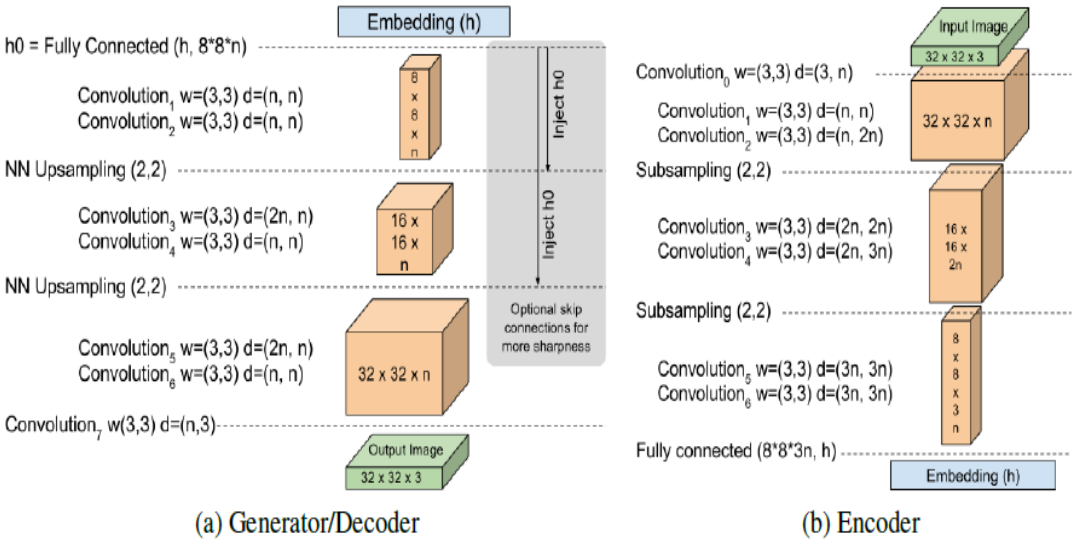


Figure 13: BEGAN Generator and Discriminator Network Architecture [3]

Enokiya, Y., et al. (2018) proposed as enhanced Deep Convolutional Generative Adversarial Network (DC-GAN), which is a combined U-Net with Wasserstein GANs for the automatic Segmentation of the liver. The dice value results for the hybrid model improved in comparison to the standard UNet from 88% to 92% and from 92% to 93% when performed on with 33 and 392 training data sets respectively [21]. Figures 14 and Figure 15 show the block diagram for the hybrid UNet with DC-GAN generator and its discriminator respectively. Other variants of the DC-GAN with residual neural network (Res-UNet) was proposed to for blood cell image classification frameworks [69]. In addition, in the reduction of the design time and optimization overhead, Hodge, J. A., et al. (2019) used a deep convolutional generative adversarial network (DC-GAN) approach to perform inverse design for radio frequency (RF) applications, such as communications and radar, that have strict radiation requirements [47].

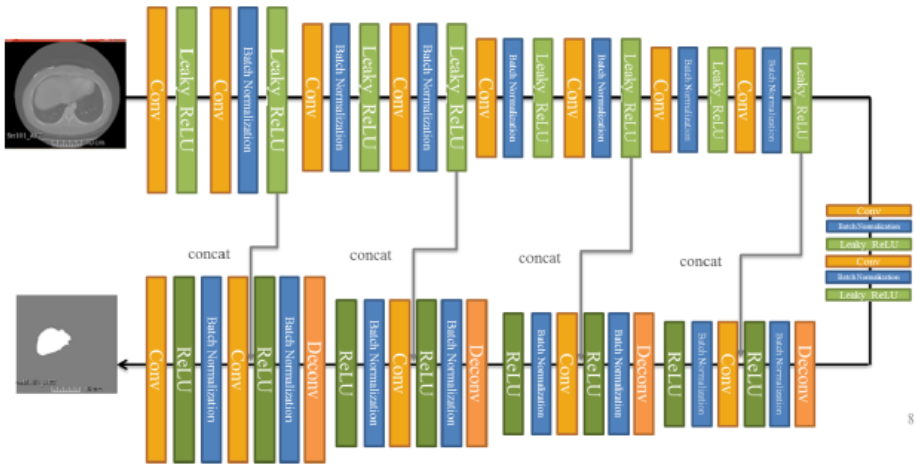


Figure 14: U-Net with Enhanced DC-GAN Generator [21].

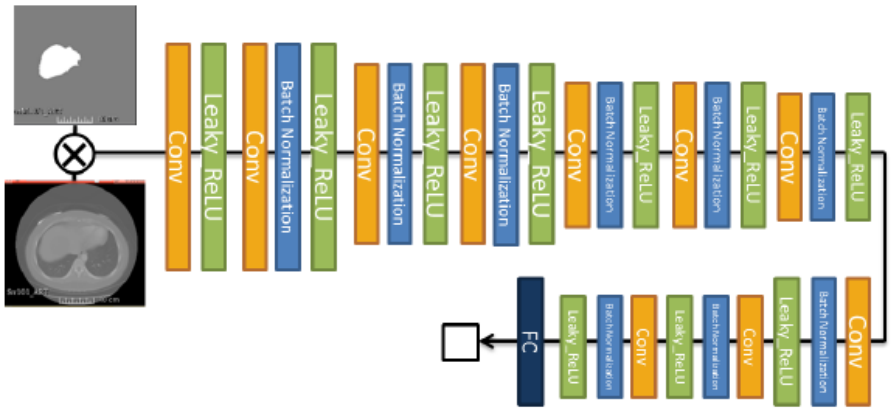


Figure 15: U-Net with Enhanced DC-GAN Discriminator Architecture [21].

Ma, L., et al. (2020) proposed a framework that combined deep convolutional generative adversarial network (DC-GAN) and a residual neural network (ReS-UNet) for the classification of white blood cell images. The DC-GAN-ReS-UNet classification accuracy outperformed that of CNN, RNN_CNN models with accuracies of 84.08%, 89.38% and 91.68% respectively [69]. Another hybrid U-Net architecture models proposed by El Jurdi, R., et al. (2020) [20], is called Bounding-Box U-Net (BB-UNet), a U-Net inspired deep

learning model that integrates both the location as well as shape prior at the level of skip connections onto model training. The BB-UNet architecture is given in figure 16.

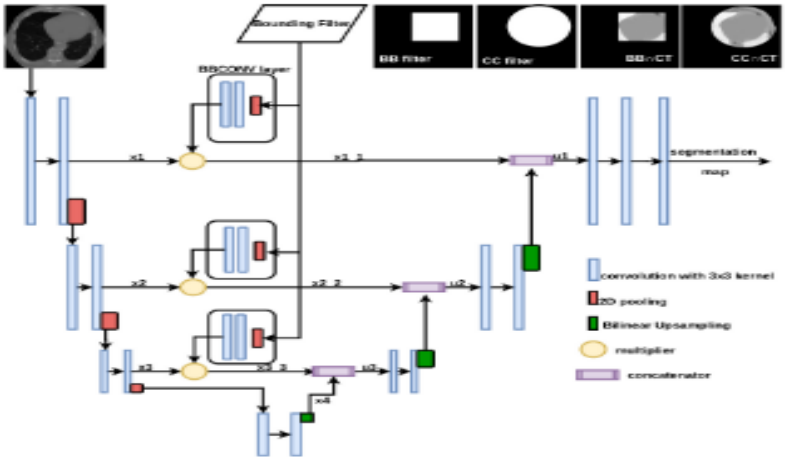


Figure 16: BB-UNet Architecture [20]

Figure 17 shows the UNet++ model proposed by Zhou, Z., et al. (2019), which is an evolution from the original U-Net [125]. The encoder shares the rich features learnt about the image with the decoder.

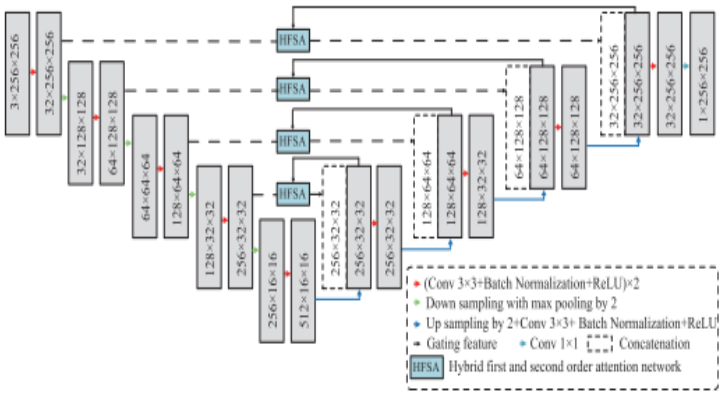


Figure 17: UNet++ Model Architecture [125]

2.6 Common Statistical Evaluation Metrics in Knee Segmentation Studies

The image segmentation Dice similarity coefficient (DSC), is the degree of agreement between the segmentation and groundtruth on pixel (2D image) or voxel (3D image) basis as shown in equation (1.0)

$$\text{Dice similarity coefficient} = \text{DSC} = \frac{2TP}{(TP+FP)+(FN+TP)} \quad \textbf{(1.0)}$$

Whereby: TP = True Positive, TN = True Negative, FP = False Positive and FN = False Negative.

The sensitivity (sens) in equation (2.0), is defined as the measure of correct classification of bone/cartilage areas on both segmentation and groundtruth. The specificity (spec), on the other hand, is defined as the measure of correct classification of non-bone/non-cartilage areas on both segmentation and groundtruth, as shown in equation (3.0)

Sensitivity $\text{Sens} = \frac{TP}{TP+FN}$ (2.0)

Specificity $\text{Spec} = \frac{TN}{TN+FP}$ (3.0)

The Coefficient of variation (CD) is the degree of dispersion around the mean value (in percentage), as shown in equation (4.0).

$$\text{Coefficient of variation} \quad \text{CV} = \frac{100\ (SD)}{mean} \quad \textbf{(4.0)}$$

Where SD is the standard deviation.

The Similarity coefficient, which is also called Jaccard Index (JI) or Jaccard Similarity Coefficient (JSI), is the intersection of both segmentation and groundtruth divided by the union of both segmentation and groundtruth. The Jaccard similarity coefficient of two sets; the segmented image set X and groundtruth set Y is expressed as shown in equations (5.0), (6.0) and (7.0).

$$\text{Similarity (Jaccard Similarity Index)} \qquad \frac{|X \cap Y|}{|X \cup Y|} \tag{5.0}$$

Where |X| represents the cardinal of set X. Thus, the Jaccard Similarity Index (JSI) or Jaccard Similarity Coefficient is also known as the intersection over union or (IoU).

The Jaccard index can also be expressed in terms of true positives (TP), false positives (FP) and false negatives (FN) as:

$$\text{Jaccard Index(X,Y)} = \frac{TP}{(TP+FP+FN)} \tag{6.0}$$

The Jaccard index is related to the Dice index according to:

$$\text{Jaccard Index (X,Y)} = \frac{dice\ (X,Y)}{(2-dice\ (X,Y))} \tag{7.0}$$

Chapter 3 Methodology

Utilizing the advantage of the enhanced skip connections in the UNet++, the purpose of this study is to propose an automatic segmentation method for the femur bones for subsequent work as quantitative MR imaging biomarkers of knee joint composition.

3.1 The Proposed Segmentation Method: Simplified-UNet

The proposed Simplified-UNet was inspired by the work from both Zhou, Z., et al. (2019) UNet++ [126] and the El Jurdi, R., et al. (2020) Bounding-Box UNet (BB-UNet) [20], which were discussed in the previous chapter. The flowchart of the proposed model is shown in Figure 18.

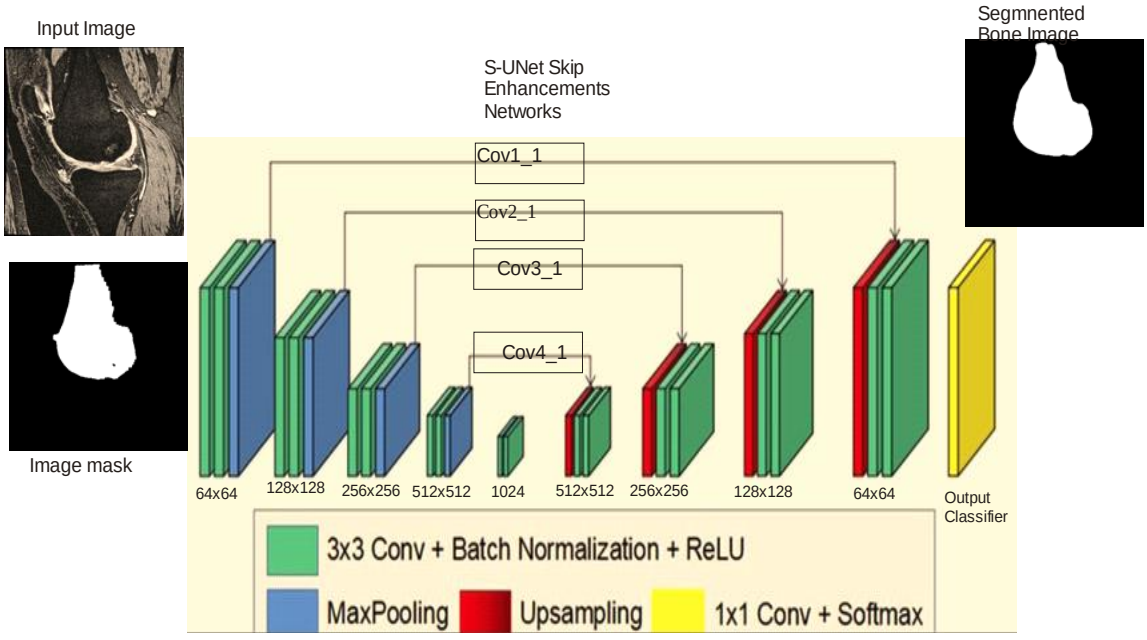


Figure 18: Proposed Simplified-UNet Model Architecture

The input image and the image mask (ground truth) of the femur bone are 384x384 pixels. They are fed into the contraction path (encoder) that starts with the first convolutional module (Cov1). There are four such convolutional modules, Cov1, Cov2, Cov3 and Cov4. Each of the four modules comprises of two consecutive convolutional layers followed by a max-pooling layer.

The contraction path functionality of the standard UNet architecture has a limitation that the network learns the "WHAT" information in the image, however, it loses the "WHERE" information. The proposed S-UNet incorporates enhancement skip-connection convolutional modules (Cov1, Cov2, Cov3 and Cov4) that share the rich features of the "WHAT" information already acquired by the encoder with the newly "WHERE" information that is extracted by the decoder. The S-UNet output uses a sigmoid classifier function, which is a special case of the Softmax classifier to transform the raw outputs into probabilities.

3.2 Dataset

The Osteoarthritis Initiatives (OAI) established and maintains a natural history of radiological (x-ray images and magnetic resonance images) database for clinical evaluation. The OAI MR images were taken with 3.0 Tesla MRI scanners from a group of 4,796 middle-aged men and women ages 45-79, including OA progressions and those at risk of OA for a period of 2 years. Osteoarthritis Initiative (OAI) dataset maintains data of 4,796 participants at baseline and follow-up at 12 months, 24 months, 36 months, and 48 months. Each follow-up includes a sagittal MRI for quantifying cartilage volume and thickness.

In this research, we used a subset of OAI with 88 knees cases. Each knee contains a whole sequence of 160 MR images. The size of each of the image is 384x384 pixels. Trained personnel have manually delineated the femur bone in each image. The manual delineation of femur bones serves as ground truth to train the machine learning models. Each original image in the dataset has a corresponding binary image mask. The dataset is split into three groups for the train, test, and validation with the ratio of 70%, 15%, and 15% respectively.

3.3 Environment Set Up

The S-UNet model algorithm was coded in python, with the following python plugin modules requirement to be installed in the environment: SimpleITK, photutils, tiff file and libtiff, and skimage. Initially, the training of the data was carried out on a local PC, Power Spec Windows 10 PC G223, with 6GB of GPU capability, until it ran out of memory. The training was carried out via Anaconda navigation virtual environment version 1.9.12 with NumPy 1.7, tensorflow-GPU and all its dependencies. The Conda dependencies include numpy, scipy, cython, h5py, Pillow, scikit-image, tensorflow-GPU version 1.5, keras and jupyter.

The data training was eventually migrated onto Google Colab for a faster result. However, the GPU dedicated and shared memory capacity may affect the processing performance, especially for MR images with very low contrast after preprocessing. Moving the model training to a faster processing power environment, such as Google CoLab is a great method to overcome such challenges.

Below are the steps to run the program on Google Colab, a Jupyter notebook environment that requires no setup to use and runs entirely in the cloud.

Copy the train, test and validation dataset onto a google drive. Change the Colab notebook settings hardware accelerator from None to GPU. Mount your google drive containing the dataset to "*Colab*" by using the following "*import*" command:

```
from google.colab import drive
drive.mount('/content/drive')
```

Authorization code will be sent via the Brower to be used for the google drive mounting.

Use the following command to run the data preparation python code in dataGC.py, which is located in the google Colab mounted drive, to prepare the NPY files from the input images.

```
#!python "/content/drive/My Drive/DPS2018AI/app/UnetCTRL/dataGC.py"
!python "/content/drive/My Drive/DPS2018AI/app/UnetCTRL/dataGC.py"
```

The dataGC.py is used for loading the train, test and validation data is shown in Appendix-C. Each iteration for the training model takes around 127 seconds/step, with the Google Colab GPU. Then Unet_1024GC.py python code snippets run directly in the browser notebook to generate the predicted results and the graph plot. This facilitates the results visualizations and makes the comparison easy, since multiple notebooks could be opened simultaneously for hyper-parameter tuning. The hypermeters tuning could be performed by changing the model variable in the Unet_1024 file, such as the number of epochs, the batch size and others.

3.4 Other Deep Learning Models Investigated

3.4.1 Hybrid Combination of Mask-RCNN RoIAlign with U-Net

The earlier part of the research work was the investigation into the proposal of a hybrid model that incorporates some layers of the Mask R-CNN architecture into the U-Net model, for the segmentation of the knee bones and cartilages, with OAI datasets. Mask R-CNN was described in more detail in section 2.4.5. A powerful tool in the Mask R-CNN [42] is its use of a more accurate feature extraction layer, called RoIAlign, in comparison to the quantization-error prone RoIPool layer used by Faster R-CNN [92].

An investigation of a feasible deep learning model that introduces a region of interest alignment (RoIAlign) layer into the UNet architecture, such that it will serve as a region proposal refinement network to target and narrow the region of interest (cartilages and bones) for the U-Net. While the enhanced UNet model trained well, it fails to converge for the test datasets. The enhanced UNet model performed fairly well, when tried on another medical image dataset, the breast lesion dataset, as shown in figures 19 (a-b) and figures 20(a-b).

Additionally, enhanced UNet model was reporting a problem of low input image contrast during the model training. Work is still ongoing to resolve the problem of this RoIAligned enhanced UNet model.

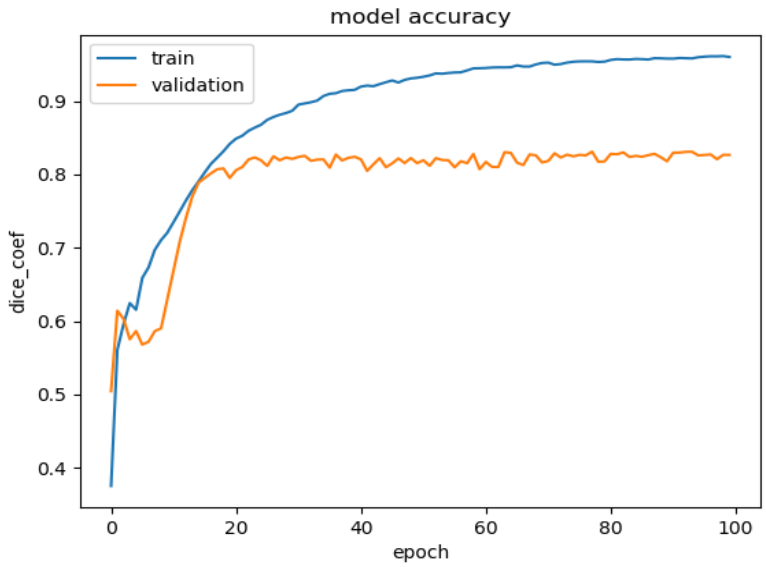


Figure 19 (a) ROI-UNet Model Accuracy (Breast CT scan)

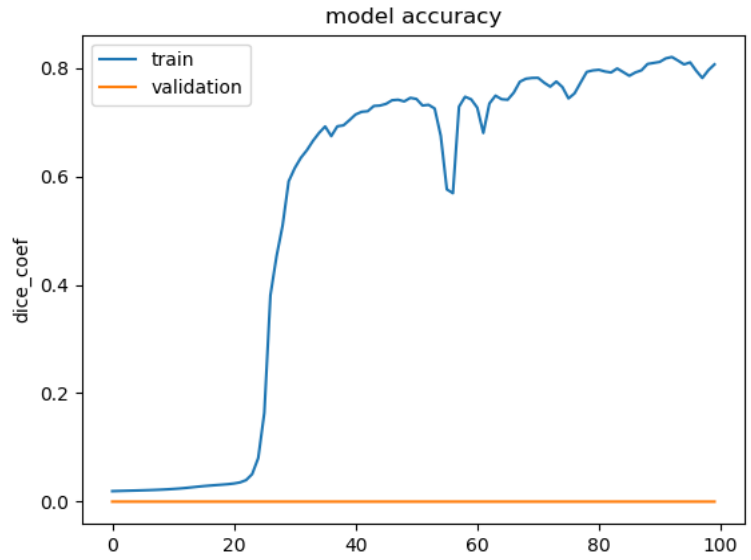


Figure 19(b): UNet Model Accuracy (Knee OAI datasets)

Additionally, it was reporting a problem of low input image contrast during the model training. Work is still ongoing to resolve the problem of this RoIAligned enhanced UNet model.

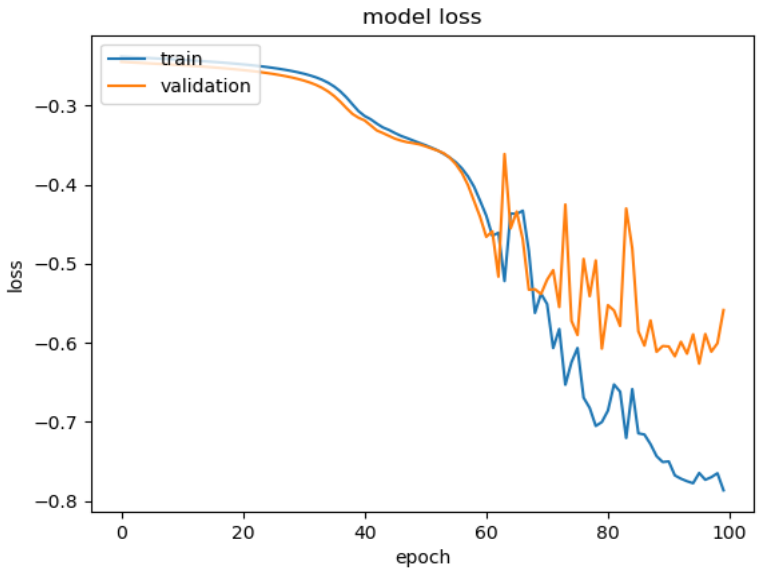


Figure 20 (a) ROI-UNet Model Loss failed validation (Breast CT scan dataset)

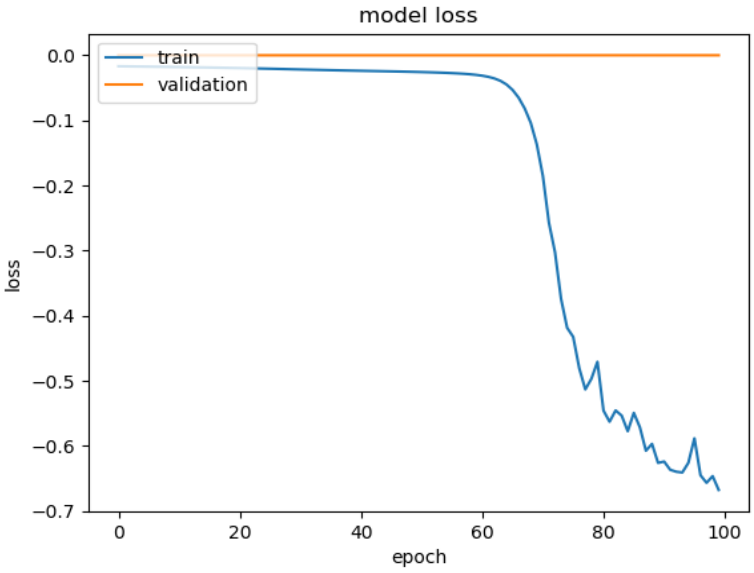


Figure 20(b): UNet Model Loss failed validation (Knee OAI datasets)

3.4.2 Hybrid Combination of BCD-Net with U-Net

Later, the search for the novel effective deep learning segmentation model shifted to an extension of Lee, H., et al. (2018) proposal of BCD-Net [60], using SKI10 datasets. The architecture of Lee's BCD-Net is shown in figure 21, in which deep segmentation networks (DSNs) are used for bone-cartilage-complex (BCC) modeling and the bone-BCC-difference (BCD) for the segmentation of the knee bones and the cartilages in the multi-class SKI10 MRI dataset. He used multi-view voting for the segmentation mask prediction.

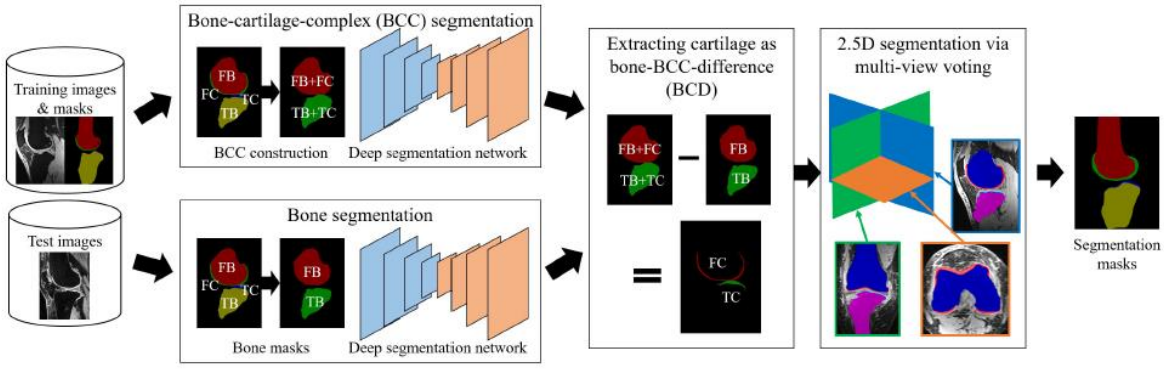


Figure 21: BCD-Net Architecture for SKI10 [60].

This work investigated the deep learning modelling of hybrid combination of BCD-Net with U-Net. Intensity thresholding was first used for the preprocessing and segmentation of the training of the multi-class image masks data, as shown in figure 21. However, the proposed algorithm failed to compile when tested on the SKI10 datasets. This is an ongoing work to address the error in the algorithm.

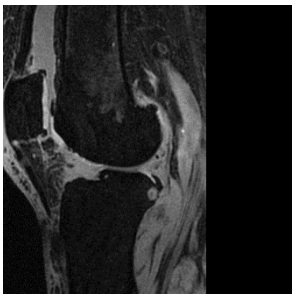
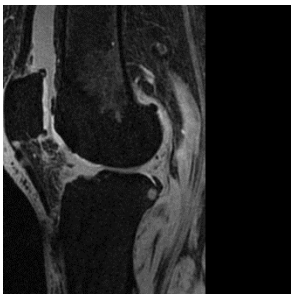
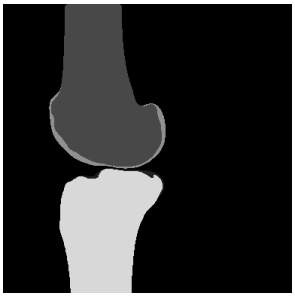
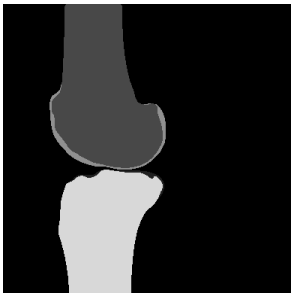
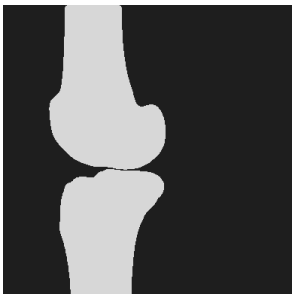
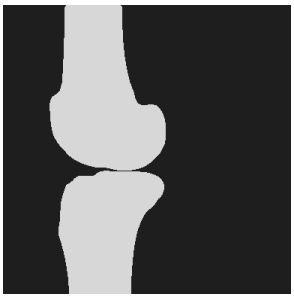
SKI10 (img_270.png)	SKI10 (img_271.png)
 (a) Original image	 (a) original image
 (b) Multi-class image mask	 b) Multi-class image mask
 (c) Segmented Femur and tibia bones	 (c) Segmented Femur and tibia bones
 (d) Segmented Femur only	 (d) Segmented Femur only

Figure 22: Sample SKI10 Preprocessed Knee MRI data.

Chapter 4 Experiment and Results

4.1 Parameter Setting

The contraction path (encoder) convolutional layers and the expansive path (decoder) transpose convolutional layers use ReLu activation with zero paddings. The max-pooling, which reduces the size of the feature map so that we can have fewer parameters in the network uses strides of 2. It means that the network selects the maximum pixel value and obtained a pooled feature map from every 2x2 block of the input feature. The final output uses sigmoid activation function. A smoothing rate value used for the S-UNet model is 0.00001.

The loss function and dice coefficient were tracked at each epoch in order to generate an output performance graph. The model is compiled with an Adam optimizer, using a binary cross-entropy loss function since there are only two classes in the experiment (bone and no bone). A batch size of 2 was used on the PC with 1080GTX NVidia GPU, while the Google Colab environment used a batch size of 8 during training. The model training was configured to stop early if the validation loss does not improve for 10 continuous epochs.

Each model was tried with 20 epochs, and then later increased to 50 epochs and 100 epochs. The training of the S-UNet model on the OAI datasets shows that the accuracy reaches exceeds 90% after the 10^{th} epochs.

4.2 Evaluation Metrics

The SUNet model is compiled with Adam optimizer that uses binary cross entropy loss function, since there are only two classes, “bone” or “no bone” (background). The Dice coefficient, otherwise known as Dice similarity index (DSC) in equation (8.0) , which is the same as the F1 score, is often used to quantify the performance of image segmentation methods. The S-UNet algorithm was validated by calculating the model Dice score, which is a measure of how similar the objects are to the ground-truth. In context, it is twice the size of the overlap of the segmented objects and the ground truth divided by the total size of the two objects.

$$\text{DSC} = \frac{2*|X \cap Y|}{|X|+|Y|} \quad \textbf{(8.0)}$$

Where |X| and |Y| are the number of elements in each set.

Besides Dice coefficient, we also employed Similarity coefficient, which is also called Jaccard Index in equation (9.0) . It is another commonly used evaluation metric for segmentation problems.

$$\text{Similarity (Jaccard Index)} = \frac{|X \cap Y|}{|X \cup Y|} \quad \textbf{(9.0)}$$

4.3 Results of the Original U-Net Model

Figure 23 (a-c) shows the original MR image of the knee, the ground truth and the UNet segmented femur bone respectively.

Knee bone segmentation with deep learning, using standard UNet model (image 970170754.png)

(a) Original Image

(b) Ground Truth

(c) UNet Predicted Image

Figure 23 [a-c]: Example data and segmentation results with standard UNet model

The average experimental results for the standard UNet model dice accuracy, model loss and similarity coefficients are shown in Table 3 and Figure 24.

Table 3: Result of model performance for original UNet

Metrics	Train data	Validation data	Test data
Dice coefficient	0.9744	0.9065	0.9076
Similarity value	0.9501	0.8324	0.8328
Loss value	-0.9746	-0.9066	-0.9089

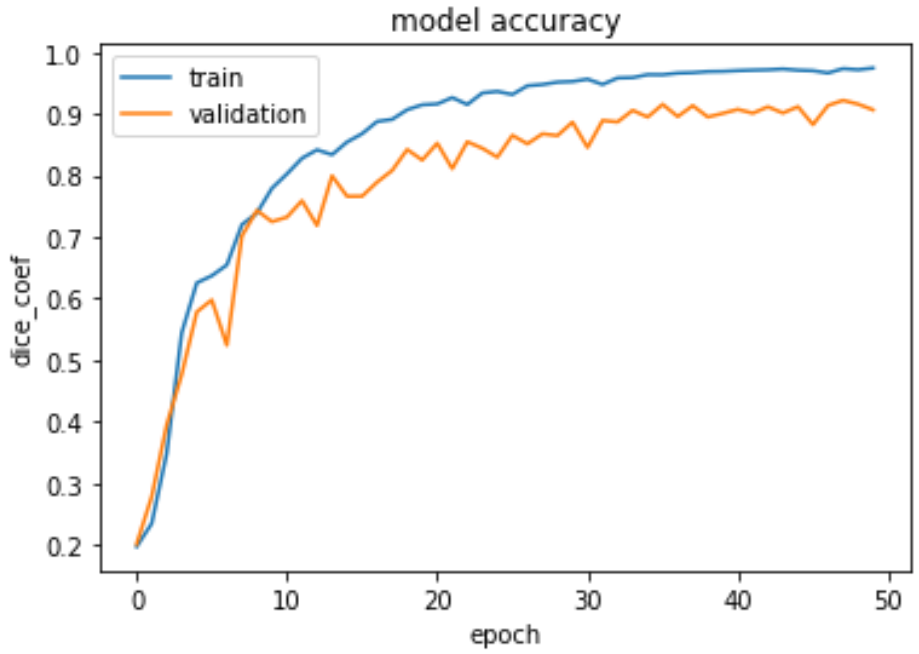


Figure 24: Original UNet plot of model accuracy dice-coefficient at each epoch

Figure 25 is the plot for the original UNet model loss at each of the 50, which converges to -0.9088.

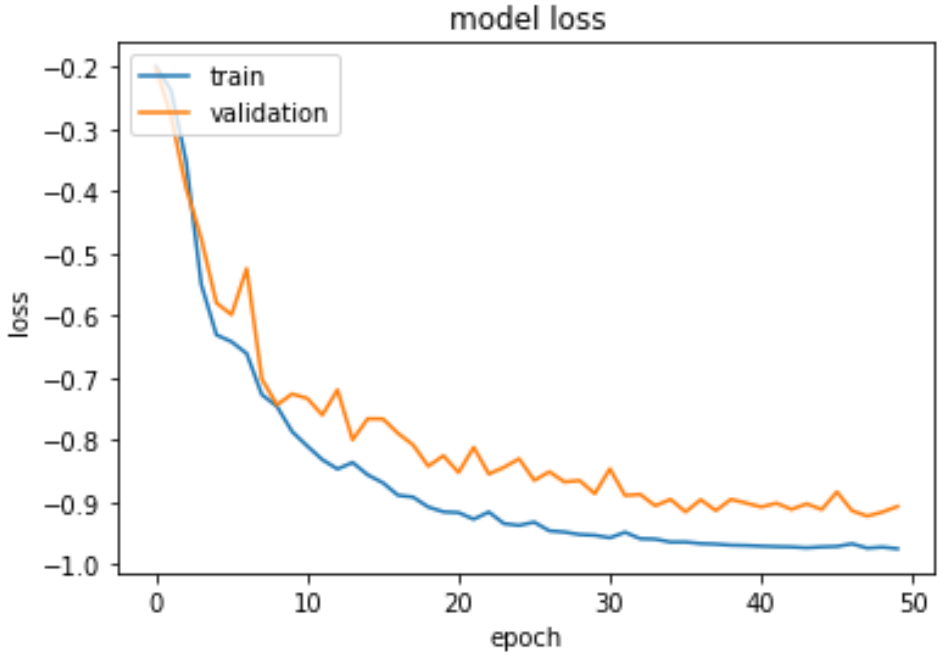


Figure 25: Original UNet plot of model loss at each epoch

The performance on the validate set is Dice = 90.65% and Similarity = 83.24%. The model is finally tested with the 15% test set. The performance on the test set is Dice = 90.76% and Similarity = 83.28%.

4.4 Results of Simplified-UNet (S-UNet) Model

The experimental results for the Simplified-UNet accuracy for the dice-coefficient and the model loss at each of the 50 epochs are shown in figure 26 and figure 27 respectively. The model dice-coefficient versus accuracy plot shows that both the train and validation are well above 90% in less than 12 epochs. The results indicates that the average dice-coefficient and the similarity coefficient are 95.52% and 91.42% respectively.

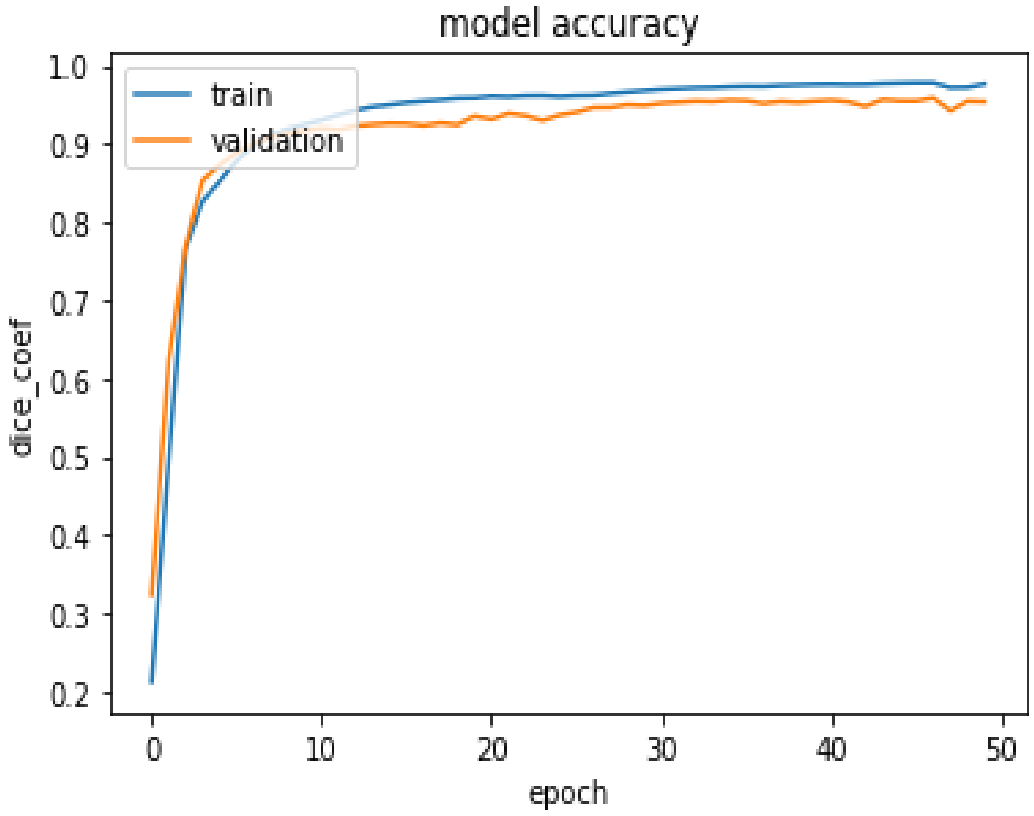


Figure 26: Simplified-UNet plot of model accuracy dice-coefficient at each epoch

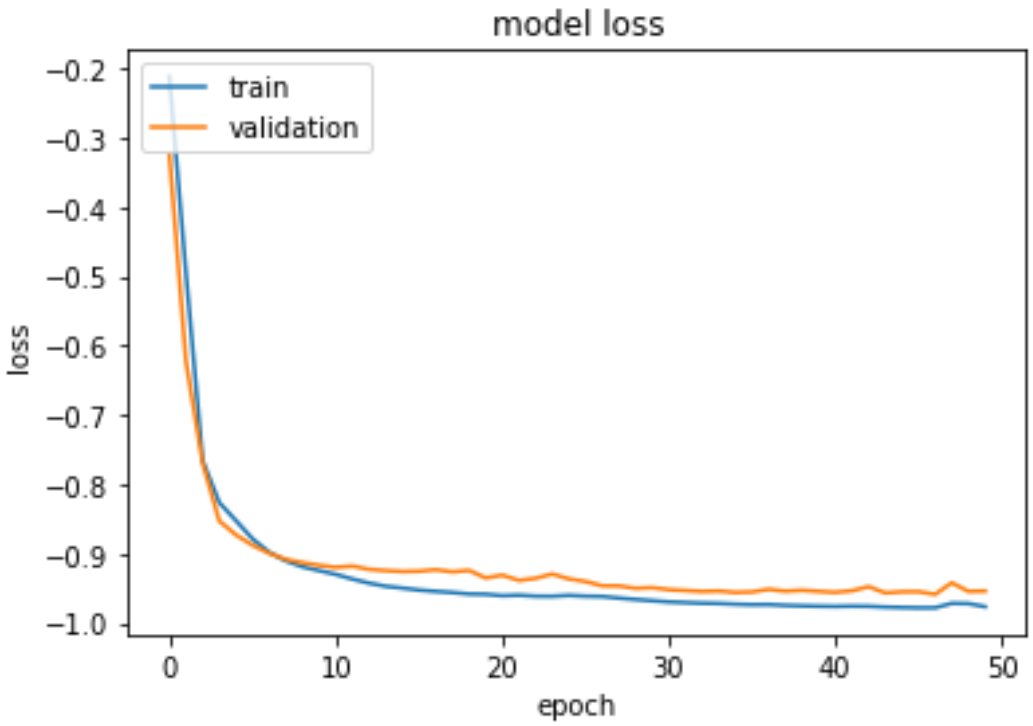


Figure 27: Simplified-UNet plot of model loss at each epoch

The performance on the validation set is Dice = 95.38% and Similarity 91.19%. The performance on the test set is Dice = 95.30% and Similarity = 91.02%. The results of the Simplified U-Net model using the OAI datasets, showing the dice similarity coefficient, the similarity value and loss value is presented in Table 4.

Table 4: Result of model performance for Simplified-UNet

Metrics	Train data	Validation data	Test data
Dice coefficient	0.9766	0.9538	0.9530
Similarity value	0.9544	0.9119	0.9102
Loss value	-0.9766	-0.9540	-0.9541

The segmentation results from S-UNet demonstrated good agreement with the overall contours of the ground truth as shown in the comparison between Figure 28 (b) and Figure 28 (c).

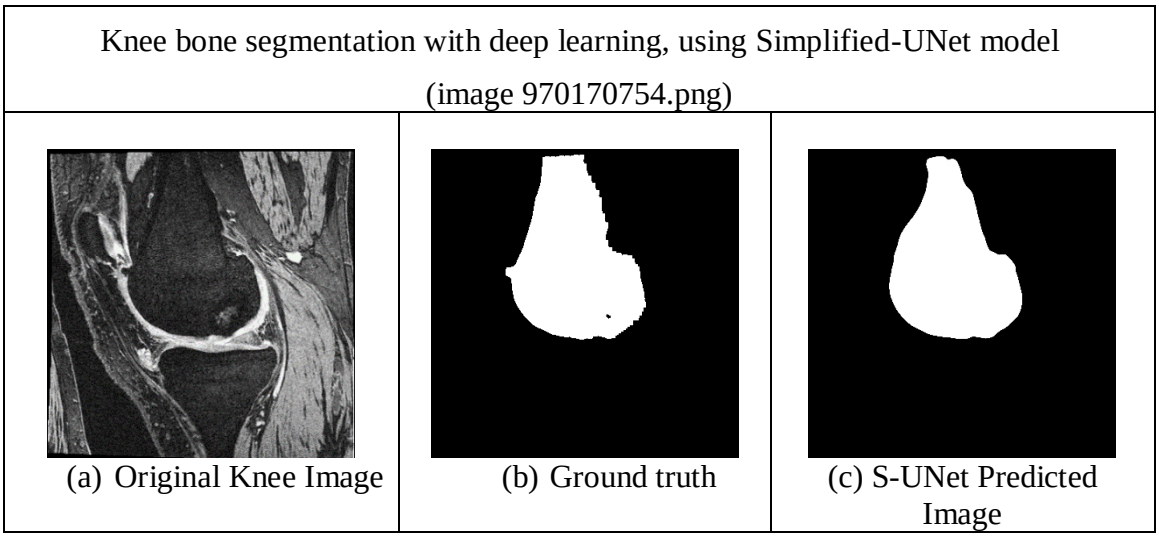


Figure 28 [a-c]: S-UNet Example data and segmentation results

4.5 Experiment Result Analysis

As the results shown above, the proposed Simplified-UNet model outperforms the original U-Net by more than 4%, with a resulting Dice accuracy coefficient of 95.38%. Table 5 shows the comparison of the segmentation results, based on the mean dice for the femur bone in the original UNet model and the proposed Simplified-UNet (S-UNet) model, on both validation and test sets.

Table 5: Comparison of original UNet and Proposed S-UNet Segmentation Results

Metrics	Original UNet	Proposed S-UNet	Improvement
Dice of validation set	90.65%	95.38%	4.73%
Similarity of validation set	83.24%	91.19%	7.95%
Dice of test set	90.76%	95.30%	4.54%
Similarity of test set	83.28%	91.02%	7.74%

Figure 29 shows a few quantitative results, that compares the standard U-Net model to the Simplified U-Net model. The visual comparison of the predicted images from the Simplified-UNet model are more similar to the ground truth the predicted images from the UNet model, as earlier indicated by the results in the Table 5.

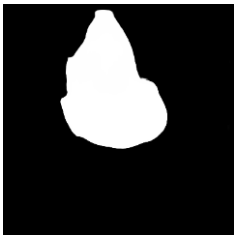
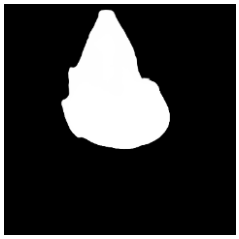
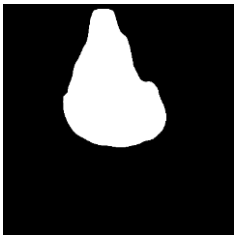
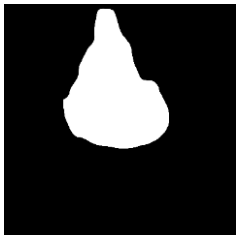
	Image 970170752	Image 970170751	Image 970170750
Original input Image			
Ground Truth			
Predicted image using original-UNet Model			
Predicted image using S-UNet Model			

Figure 29: Comparison of the predicted images of the S-UNet and UNet models.

Chapter 5 Conclusion

This research work proposed for an instance segmentation of the knee femur bone from Magnetic Resonance Imaging sequences, using data from OAI database, based on an extension of the U-Net model, call simplified-UNet (S-UNet). S-UNet is a new powerful and more accurate bone segmentation method, which re-designed skip network to reduce the semantic gap between the feature maps of the encoder and decoder subnetworks, thus, simplifying the optimization problem.

The datasets was divided into three data groups and placed into three separated folders; the train, and test and validation data, which were divided in the ratio 70%, 15% and 15% respectively. The model was first trained on the Windows 10 Core i5 PC; with 32GB RAM and NVIDIA GeForce GTX 1060, 6GB dedicated and 8GB shared GPU memory. The training was carried out via Anaconda navigation virtual environment version 1.9.12 with NumPy 1.7, tensorflow-GPU and all its dependencies. However, due to memory capacity issue, the software environment was eventually migrated onto the Google Colab cloud with GPU runtime option. The S-UNet achieved a better accuracy in the segmentation of femur bone images compared to the state-of-the-art model, with Dice of 95.38% by improving the results of U-Net by a factor of 4.73%.

Among the three major knee bones in a human knee joint (femur, tibia, and patella), this work only segmented the most important femur bone. However, the proposed Simplified-UNet (S-UNet) approach is extendible to the segmentation of tibia bone and patella bone. The S-UNet could also be extended for the segmentation of cartilage of the human knee joint and the results will be effective in the prediction and monitoring of the progression of

OA. The S-UNet output uses a sigmoid classifier function, which is a special case of "Softmax" classifier. A multi-label classifier, such as softmax is recommended, if the classes for segmentation are more than two, say the femur, tibia and the fibula bones are to be included in the segmentation process.

Appendix A

Standard U-Net Architecture Python Code

```
"""UnetStdUnet_Dice90,76.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/driv/11GdkB8sTf5PwGiCIyt-ZB41AOU-n4EJ6
"""

from google.colab import drive
drive.mount('/content/drive')

# HOME DRIVE; drive.mount('/content/drive/My Drive/DPS2018AI/app/UNetPP')
# root_path = '/content/drive/My Drive/DPS2018AI/app/UNetPP'
import sys
sys.path.append('/content/drive/My Drive/DPS2018AI/app/UnetCTRL')
# sys.path.append('/content/drive/My Drive/DPS2018AI/app/UNetPP')

#!python "/content/drive/My Drive/DPS2018AI/app/UnetCTRL/dataGC.py"
!python "/content/drive/My Drive/DPS2018AI/app/UnetCTRL/dataGC.py"

pip install git+git://github.com/fchollet/keras.git --upgrade --no-deps

!pip install SimpleITK
!pip install photutils
!pip install tifffile
!pip install libtiff

from __future__ import print_function

import os
#Apr17
import tensorflow as tf
from skimage.transform import resize
from skimage.io import imsave
import numpy as np
from keras.models import Model
# from keras.layers import Input, concatenate, Conv2D, MaxPooling2D,
Conv2DTranspose
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint
```

```
from keras import backend as K
from keras.models import load_model
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense

#EA 2020Apr17  from data import load_train_data, load_test_data, load_validate_data
# from ValTsTr_data import load_train_data, load_test_data, load_validate_data
# from dataGC import load_train_data, load_test_data, load_validate_data

from keras.layers import Input, merge, Conv2D, ZeroPadding2D, UpSampling2D, Dense,
concatenate, Conv2DTranspose
from keras.layers.pooling import MaxPooling2D, GlobalAveragePooling2D,
MaxPooling2D
from keras.layers.core import Dense, Dropout, Activation
from keras.layers import BatchNormalization, Dropout, Flatten, Lambda
from keras.layers.advanced_activations import ELU, LeakyReLU
from keras.optimizers import Adam, RMSprop, SGD
from keras.regularizers import l2
from keras.layers.noise import GaussianDropout

K.clear_session()
K.set_image_data_format('channels_last')  # TF dimension ordering in this code

def load_test_data():
    imgs_test = np.load('imgs_test.npy')
    imgs_mask_test = np.load('imgs_mask_test.npy')
    imgs_id = np.load('imgs_id_test.npy')
    return imgs_test, imgs_id, imgs_mask_test

def load_validate_data():
    imgs_validate = np.load('imgs_validate.npy')
    imgs_mask_validate = np.load('imgs_mask_validate.npy')
    return imgs_validate, imgs_mask_validate

def load_train_data():
    imgs_train = np.load('imgs_train.npy')
    imgs_mask_train = np.load('imgs_mask_train.npy')
    return imgs_train, imgs_mask_train

img_rows = 384
img_cols = 384

smooth = 1e-5

def dice_coef(y_true, y_pred):
```

```
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)
    intersection = K.sum(y_true_f * y_pred_f)
    return (2. * intersection + smooth) / (K.sum(y_true_f) + K.sum(y_pred_f) + smooth)

def similarity(y_true, y_pred):
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)
    intersection = K.sum(K.abs(y_true_f * y_pred_f))
    return ((intersection) / (K.sum(K.abs(y_true_f)+ K.abs(y_pred_f)) - intersection))

def dice_coef_loss(y_true, y_pred):
    return -dice_coef(y_true, y_pred)

def get_unet():
    inputs = Input((img_rows, img_cols, 1))
    nb_filter = [64,128,256,512,1024]
    inputs = Input((img_rows, img_cols, 1))
    conv1_1 = Conv2D(nb_filter[0], (3, 3), activation='relu', padding='same')(inputs)
    conv1_1 = Conv2D(nb_filter[0], (3, 3), activation='relu', padding='same')(conv1_1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1_1)

    conv2_1 = Conv2D(nb_filter[1], (3, 3), activation='relu', padding='same')(pool1)
    conv2_1 = Conv2D(nb_filter[1], (3, 3), activation='relu', padding='same')(conv2_1)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2_1)

    conv3_1 = Conv2D(nb_filter[2], (3, 3), activation='relu', padding='same')(pool2)
    conv3_1 = Conv2D(nb_filter[2], (3, 3), activation='relu', padding='same')(conv3_1)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3_1)

    conv4_1 = Conv2D(nb_filter[3], (3, 3), activation='relu', padding='same')(pool3)
    conv4_1 = Conv2D(nb_filter[3], (3, 3), activation='relu', padding='same')(conv4_1)
    pool4 = MaxPooling2D(pool_size=(2, 2))(conv4_1)

    conv5_1 = Conv2D(nb_filter[4], (3, 3), activation='relu', padding='same')(pool4)
    conv5_1 = Conv2D(nb_filter[4], (3, 3), activation='relu', padding='same')(conv5_1)

    up4_2 = concatenate([Conv2DTranspose(nb_filter[3], (2, 2), strides=(2, 2),
padding='same')(conv5_1), conv4_1])
    conv4_2 = Conv2D(nb_filter[3], (3, 3), activation='relu', padding='same')(up4_2)
    conv4_2 = Conv2D(nb_filter[3], (3, 3), activation='relu', padding='same')(conv4_2)

    up3_3 = concatenate([Conv2DTranspose(nb_filter[2], (2, 2), strides=(2, 2),
padding='same')(conv4_2), conv3_1])
    conv3_3 = Conv2D(nb_filter[2], (3, 3), activation='relu', padding='same')(up3_3)
    conv3_3 = Conv2D(nb_filter[2], (3, 3), activation='relu', padding='same')(conv3_3)
```

```
    up2_4 = concatenate([Conv2DTranspose(nb_filter[1], (2, 2), strides=(2, 2), padding='same')(conv3_3), conv2_1])
    conv2_4 = Conv2D(nb_filter[1], (3, 3), activation='relu', padding='same')(up2_4)
    conv2_4 = Conv2D(nb_filter[1], (3, 3), activation='relu', padding='same')(conv2_4)

    up1_5 = concatenate([Conv2DTranspose(nb_filter[0], (2, 2), strides=(2, 2), padding='same')(conv2_4), conv1])
    conv1_5 = Conv2D(nb_filter[0], (3, 3), activation='relu', padding='same')(up1_5)
    conv1_5 = Conv2D(nb_filter[0], (3, 3), activation='relu', padding='same')(conv1_5)

    unet_output = Conv2D(1, (1, 1), activation='sigmoid')(conv1_5)

#   model = Model(inputs=[inputs], outputs=[conv10])
    model = Model(inputs=[inputs], outputs=[unet_output])

    # model.load_weights('weights.h5')

    model.compile(optimizer=Adam(lr=1e-5), loss=dice_coef_loss, metrics=[similarity, dice_coef])

    return model

def preprocess(imgs):
    imgs_p = np.ndarray((imgs.shape[0], img_rows, img_cols), dtype=np.uint8)
    for i in range(imgs.shape[0]):
        imgs_p[i] = resize(imgs[i], (img_cols, img_rows), preserve_range=True)

    imgs_p = imgs_p[..., np.newaxis]
    return imgs_p


def train_and_predict():
    print('-'*30)
    print('Loading and preprocessing train data...')
    print('-'*30)
    imgs_train, imgs_mask_train = load_train_data()

    imgs_train = preprocess(imgs_train)
    imgs_mask_train = preprocess(imgs_mask_train)

    imgs_train = imgs_train.astype('float32')
    mean = np.mean(imgs_train)  # mean for data centering
    std = np.std(imgs_train)  # std for data normalization

    imgs_train -= mean
```

```
imgs_train /= std

imgs_mask_train = imgs_mask_train.astype('float32')
imgs_mask_train /= 255.  # scale masks to [0, 1]

print('-' * 30)
print('Loading and preprocessing validation data...')
print('-' * 30)
imgs_validate, imgs_mask_validate = load_validate_data()

imgs_validate = preprocess(imgs_validate)
imgs_mask_validate = preprocess(imgs_mask_validate)

imgs_validate = imgs_validate.astype('float32')
imgs_validate -= mean
imgs_validate /= std

imgs_mask_validate = imgs_mask_validate.astype('float32')
imgs_mask_validate /= 255.  # scale masks to [0, 1]

print('-'*30)
print('Loading and preprocessing test data...')
print('-'*30)
imgs_test, imgs_id_test, imgs_mask_test = load_test_data()

imgs_test = preprocess(imgs_test)
imgs_mask_test = preprocess(imgs_mask_test)

imgs_test = imgs_test.astype('float32')
imgs_test -= mean
imgs_test /= std

imgs_mask_test = imgs_mask_test.astype('float32')
imgs_mask_test /= 255.  # scale masks to [0, 1]


print('-'*30)
print('Creating and compiling model...')
print('-'*30)
model = get_unet()
# Different
# EA 2020Ar17
callbacks = [tf.keras.callbacks.ModelCheckpoint('weights.h5', monitor='val_loss',
save_best_only=True),
             tf.keras.callbacks.EarlyStopping(monitor='val_dice_coef', patience=5,
min_delta=0, mode='max'),
```

```
        tf.keras.callbacks.CSVLogger('model_1_logs.csv')]

    model_checkpoint = ModelCheckpoint('weights.h5', monitor='val_loss', save_best_only=True)
    #
    print('-'*30)
    print('Fitting model...')
    print('-'*30)

    # Fitting model; increment epochs appropriately (default epochs=300, default batch_size=8)
    # EA -2020Apr17 history = model.fit(imgs_train, imgs_mask_train, batch_size=10, epochs=80, verbose=1, shuffle=False,
    # history = model.fit(imgs_train, imgs_mask_train, batch_size=2, epochs=20, verbose=1, shuffle=False,
    # 05152020 history=model.fit(imgs_train, imgs_mask_train, batch_size=2, epochs=20, verbose=1, shuffle=False,

    history=model.fit(imgs_train, imgs_mask_train, batch_size=8, epochs=50, verbose=1, shuffle=False,
            validation_data=(imgs_test, imgs_mask_test),
            callbacks=[model_checkpoint])

    #EA-Apr17 Evaluate the model
    print('-' * 30)
    print('Evaluating the model on test data...')
    print('-' * 30)

    score = model.evaluate(imgs_test, imgs_mask_test)
    print('\nloss: ', score[0])
    print('dice_coef: ', score[2])
    print('similarity: ', score[1], '\n')


    # list all data in history
    print(history.history.keys())

    plt.plot(history.history['dice_coef'])
    plt.plot(history.history['val_dice_coef'])
    plt.title('model accuracy')
    plt.ylabel('dice_coef')
    plt.xlabel('epoch')
    plt.legend(['train', 'validation'], loc='upper left')
    plt.show()

    plt.plot(history.history['loss'])
```

```
    plt.plot(history.history['val_loss'])
    plt.title('model loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'validation'], loc='upper left')
    plt.show()

    print('-'*30)
    print('Loading saved weights...')
    print('-'*30)
    # model.load_weights('weights.h5')

    print('-'*30)
    print('Predicting masks on test data...')
    print('-'*30)
    imgs_mask_test_pred = model.predict(imgs_test, verbose=1)

    # np.save('imgs_mask_test.npy', imgs_mask_test_pred)

    print('-' * 30)
    print('Saving predicted masks to files...')
    print('-' * 30)
    # pred_dir = 'preds'
    # 2020May15 Google COlab =GC
    # Google COlab home directory '/content/drive/My Drive/DPS2018AI/app/unet3/'
    pred_dir = '/content/drive/My Drive/DPS2018AI/app/UnetCTRL/preds'
    # pred_dir = '/content/drive/My Drive/DPS2018AI/app/UNetPP/preds'
    #if not os.path.exists(pred_dir):
    #    os.mkdir(pred_dir)
    for image, image_id in zip(imgs_mask_test_pred, imgs_id_test):
        image = (image[:, :, 0] * 255.).astype(np.uint8)
        imsave(os.path.join(pred_dir, str(image_id) + '_pred.png'), image)


if __name__ == '__main__':
#   model = UNetPlusPlus(96,96,1)
    model = get_unet()
    train_and_predict()
```

Appendix B

Simplified-UNet Architecture Python Code

```
# -*- coding: utf-8 -*-
""" UnetPP_OAI_Dice95,30_Tr1600Ts300V300.ipynb

Automatically generated by Colaboratory.

Original file is located at
  https://colab.research.google.com/driv/1VtKDLN0-VlUqEfiDB3_2yLbqAYnc_7Sf#scrollTo=Akf8XkDH3rjH

from google.colab import drive
drive.mount('/content/drive')

# HOME DRIVE; drive.mount('/content/drive/My Drive/DPS2018AI/app/UNetPP')
# root_path = '/content/drive/My Drive/DPS2018AI/app/UNetPP'
import sys
sys.path.append('/content/drive/My Drive/DPS2018AI/app/UnetCTRL')
# sys.path.append('/content/drive/My Drive/DPS2018AI/app/UNetPP')

#!python "/content/drive/My Drive/DPS2018AI/app/UnetCTRL/dataGC.py"
!python "/content/drive/My Drive/DPS2018AI/app/UnetCTRL/dataGC.py"

pip install git+git://github.com/fchollet/keras.git --upgrade --no-deps

!pip install SimpleITK
!pip install photutils
!pip install tifffile
!pip install libtiff

from __future__ import print_function

import os
#Apr17
import tensorflow as tf
from skimage.transform import resize
from skimage.io import imsave
import numpy as np
from keras.models import Model
# from keras.layers import Input, concatenate, Conv2D, MaxPooling2D, Conv2DTranspose
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint
```

```
from keras import backend as K
from keras.models import load_model
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense

#EA 2020Apr17  from data import load_train_data, load_test_data, load_validate_data
# from ValTsTr_data import load_train_data, load_test_data, load_validate_data
# from dataGC import load_train_data, load_test_data, load_validate_data

from keras.layers import Input, merge, Conv2D, ZeroPadding2D, UpSampling2D, Dense,
concatenate, Conv2DTranspose
from keras.layers.pooling import MaxPooling2D, GlobalAveragePooling2D,
MaxPooling2D
from keras.layers.core import Dense, Dropout, Activation
from keras.layers import BatchNormalization, Dropout, Flatten, Lambda
from keras.layers.advanced_activations import ELU, LeakyReLU
from keras.optimizers import Adam, RMSprop, SGD
from keras.regularizers import l2
from keras.layers.noise import GaussianDropout

K.clear_session()
K.set_image_data_format('channels_last')  # TF dimension ordering in this code

def load_test_data():
    imgs_test = np.load('imgs_test.npy')
    imgs_mask_test = np.load('imgs_mask_test.npy')
    imgs_id = np.load('imgs_id_test.npy')
    return imgs_test, imgs_id, imgs_mask_test

def load_validate_data():
    imgs_validate = np.load('imgs_validate.npy')
    imgs_mask_validate = np.load('imgs_mask_validate.npy')
    return imgs_validate, imgs_mask_validate

def load_train_data():
    imgs_train = np.load('imgs_train.npy')
    imgs_mask_train = np.load('imgs_mask_train.npy')
    return imgs_train, imgs_mask_train

img_rows = 384
img_cols = 384

smooth = 1e-5

def dice_coef(y_true, y_pred):
```

```
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)
    intersection = K.sum(y_true_f * y_pred_f)
    return (2. * intersection + smooth) / (K.sum(y_true_f) + K.sum(y_pred_f) + smooth)

def similarity(y_true, y_pred):
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)
    intersection = K.sum(K.abs(y_true_f * y_pred_f))
    return ((intersection) / (K.sum(K.abs(y_true_f)+ K.abs(y_pred_f)) - intersection))

def dice_coef_loss(y_true, y_pred):
    return -dice_coef(y_true, y_pred)

def get_unet():
    inputs = Input((img_rows, img_cols, 1))
    nb_filter = [64,128,256,512,1024]
    inputs = Input((img_rows, img_cols, 1))
# NBFolter 0= 64
    conv1_1 = Conv2D(nb_filter[0], (3, 3), activation='relu', padding='same')(inputs)
    conv1_1 = Conv2D(nb_filter[0], (3, 3), activation='relu', padding='same')(conv1_1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1_1)

    conv2_1 = Conv2D(nb_filter[1], (3, 3), activation='relu', padding='same')(pool1)
    conv2_1 = Conv2D(nb_filter[1], (3, 3), activation='relu', padding='same')(conv2_1)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2_1)

# ADDED:
    conv1_1 = concatenate([Conv2DTranspose(nb_filter[0], (2, 2), strides=(2, 2),
padding='same')(conv2_1), conv1_1])
    # up1_2 = Conv2DTranspose(nb_filter[0], (2, 2), strides=(2, 2), name='up12',
padding='same')(conv2_1)
    # conv1_2 = concatenate([up1_2, conv1_1], name='merge12', axis=bn_axis)
    #conv1_2 = standard_unit(conv1_2, stage='12', nb_filter=nb_filter[0])

    conv3_1 = Conv2D(nb_filter[2], (3, 3), activation='relu', padding='same')(pool2)
    conv3_1 = Conv2D(nb_filter[2], (3, 3), activation='relu', padding='same')(conv3_1)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3_1)

# ADDED
    conv2_1 = concatenate([Conv2DTranspose(nb_filter[1], (2, 2), strides=(2, 2),
padding='same')(conv3_1), conv2_1])

    conv4_1 = Conv2D(nb_filter[3], (3, 3), activation='relu', padding='same')(pool3)
    conv4_1 = Conv2D(nb_filter[3], (3, 3), activation='relu', padding='same')(conv4_1)
    pool4 = MaxPooling2D(pool_size=(2, 2))(conv4_1)
```

```
#ADDED
   conv3_1 = concatenate([Conv2DTranspose(nb_filter[2], (2, 2), strides=(2, 2), padding='same')(conv4_1), conv3_1])

   conv5_1 = Conv2D(nb_filter[4], (3, 3), activation='relu', padding='same')(pool4)
   conv5_1 = Conv2D(nb_filter[4], (3, 3), activation='relu', padding='same')(conv5_1)

#ADDED
   conv4_1 = concatenate([Conv2DTranspose(nb_filter[3], (2, 2), strides=(2, 2), padding='same')(conv5_1), conv4_1])

   up4_2 = concatenate([Conv2DTranspose(nb_filter[3], (2, 2), strides=(2, 2), padding='same')(conv5_1), conv4_1])
   conv4_2 = Conv2D(nb_filter[3], (3, 3), activation='relu', padding='same')(up4_2)
   conv4_2 = Conv2D(nb_filter[3], (3, 3), activation='relu', padding='same')(conv4_2)

   #up3_3 = concatenate([Conv2DTranspose(nb_filter[2], (2, 2), strides=(2, 2), padding='same')(conv4_2), conv3_1])

   up3_3 = concatenate([Conv2DTranspose(nb_filter[2], (2, 2), strides=(2, 2), padding='same')(conv4_2), conv3_1])
   conv3_3 = Conv2D(nb_filter[2], (3, 3), activation='relu', padding='same')(up3_3)
   conv3_3 = Conv2D(nb_filter[2], (3, 3), activation='relu', padding='same')(conv3_3)

   up2_4 = concatenate([Conv2DTranspose(nb_filter[1], (2, 2), strides=(2, 2), padding='same')(conv3_3), conv2_1])
   conv2_4 = Conv2D(nb_filter[1], (3, 3), activation='relu', padding='same')(up2_4)
   conv2_4 = Conv2D(nb_filter[1], (3, 3), activation='relu', padding='same')(conv2_4)

   up1_5 = concatenate([Conv2DTranspose(nb_filter[0], (2, 2), strides=(2, 2), padding='same')(conv2_4), conv1_1])
   conv1_5 = Conv2D(nb_filter[0], (3, 3), activation='relu', padding='same')(up1_5)
   conv1_5 = Conv2D(nb_filter[0], (3, 3), activation='relu', padding='same')(conv1_5)

   unet_output = Conv2D(1, (1, 1), activation='sigmoid')(conv1_5)

#   model = Model(inputs=[inputs], outputs=[conv10])
   model = Model(inputs=[inputs], outputs=[unet_output])

   # model.load_weights('weights.h5')

   model.compile(optimizer=Adam(lr=1e-5), loss=dice_coef_loss, metrics=[similarity, dice_coef])

   return model
```

```
def preprocess(imgs):
    imgs_p = np.ndarray((imgs.shape[0], img_rows, img_cols), dtype=np.uint8)
    for i in range(imgs.shape[0]):
        imgs_p[i] = resize(imgs[i], (img_cols, img_rows), preserve_range=True)

    imgs_p = imgs_p[..., np.newaxis]
    return imgs_p

def train_and_predict():
    print('-'*30)
    print('Loading and preprocessing train data...')
    print('-'*30)
    imgs_train, imgs_mask_train = load_train_data()

    imgs_train = preprocess(imgs_train)
    imgs_mask_train = preprocess(imgs_mask_train)

    imgs_train = imgs_train.astype('float32')
    mean = np.mean(imgs_train)  # mean for data centering
    std = np.std(imgs_train)  # std for data normalization

    imgs_train -= mean
    imgs_train /= std

    imgs_mask_train = imgs_mask_train.astype('float32')
    imgs_mask_train /= 255.  # scale masks to [0, 1]

    print('-' * 30)
    print('Loading and preprocessing validation data...')
    print('-' * 30)
    imgs_validate, imgs_mask_validate = load_validate_data()

    imgs_validate = preprocess(imgs_validate)
    imgs_mask_validate = preprocess(imgs_mask_validate)

    imgs_validate = imgs_validate.astype('float32')
    imgs_validate -= mean
    imgs_validate /= std

    imgs_mask_validate = imgs_mask_validate.astype('float32')
    imgs_mask_validate /= 255.  # scale masks to [0, 1]

    print('-'*30)
    print('Loading and preprocessing test data...')
    print('-'*30)
```

```
imgs_test, imgs_id_test, imgs_mask_test = load_test_data()

imgs_test = preprocess(imgs_test)
imgs_mask_test = preprocess(imgs_mask_test)

imgs_test = imgs_test.astype('float32')
imgs_test -= mean
imgs_test /= std

imgs_mask_test = imgs_mask_test.astype('float32')
imgs_mask_test /= 255.  # scale masks to [0, 1]


print('-'*30)
print('Creating and compiling model...')
print('-'*30)
model = get_unet()
# Different
# EA 2020Ar17
callbacks = [tf.keras.callbacks.ModelCheckpoint('weights.h5', monitor='val_loss', save_best_only=True),
        tf.keras.callbacks.EarlyStopping(monitor='val_dice_coef', patience=5, min_delta=0, mode='max'),
        tf.keras.callbacks.CSVLogger('model_1_logs.csv')]

model_checkpoint = ModelCheckpoint('weights.h5', monitor='val_loss', save_best_only=True)
#
print('-'*30)
print('Fitting model...')
print('-'*30)

# Fitting model; increment epochs appropriately (default epochs=300, default batch_size=8)
# EA -2020Apr17 history = model.fit(imgs_train, imgs_mask_train, batch_size=10, epochs=80, verbose=1, shuffle=False,
# history = model.fit(imgs_train, imgs_mask_train, batch_size=2, epochs=100, verbose=1, shuffle=False,
# 05152020 history=model.fit(imgs_train, imgs_mask_train, batch_size=2, epochs=20, verbose=1, shuffle=False,

history=model.fit(imgs_train, imgs_mask_train, batch_size=8, epochs=50, verbose=1, shuffle=False,
        validation_data=(imgs_test, imgs_mask_test),
        callbacks=[model_checkpoint])
```

```
#EA-Apr17 Evaluate the model
print('-' * 30)
print('Evaluating the model on test data...')
print('-' * 30)

score = model.evaluate(imgs_test, imgs_mask_test)
print('\nloss: ', score[0])
print('dice_coef: ', score[2])
print('similarity: ', score[1], '\n')


# list all data in history
print(history.history.keys())

plt.plot(history.history['dice_coef'])
plt.plot(history.history['val_dice_coef'])
plt.title('model accuracy')
plt.ylabel('dice_coef')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()

print('-'*30)
print('Loading saved weights...')
print('-'*30)
# model.load_weights('weights.h5')

print('-'*30)
print('Predicting masks on test data...')
print('-'*30)
imgs_mask_test_pred = model.predict(imgs_test, verbose=1)

# np.save('imgs_mask_test.npy', imgs_mask_test_pred)

print('-' * 30)
print('Saving predicted masks to files...')
print('-' * 30)
# pred_dir = 'preds'
```

```
    # 2020May15 Google COlab =GC
    # Google COlab home directory '/content/drive/My Drive/DPS2018AI/app/unet3/'
    pred_dir = '/content/drive/My Drive/DPS2018AI/app/UnetCTRL/preds'
    # pred_dir = '/content/drive/My Drive/DPS2018AI/app/UNetPP/preds'
    #if not os.path.exists(pred_dir):
    #    os.mkdir(pred_dir)
    for image, image_id in zip(imgs_mask_test_pred, imgs_id_test):
        image = (image[:, :, 0] * 255.).astype(np.uint8)
        imsave(os.path.join(pred_dir, str(image_id) + '_pred.png'), image)


if __name__ == '__main__':
#   model = UNetPlusPlus(96,96,1)
    model = get_unet()
    train_and_predict()
```

Appendix C

Data.py – For load the train, test and validation data

```
from __future__ import print_function

import os
import numpy as np

from skimage.io import imsave, imread

# path to images dataset. Recommended to write down entire path to avoid errors (i.e "C:/User/Desktop.../raw/)
# data_path = 'raw/'
# 2020May15 Google COlab mounted drive
# Use the next two lines to mount on Hooole COlab command line
#from google.colab import drive
#drive.mount('/content/drive')
#
data_path = '/content/drive/My Drive/DPS2018AI/app/unet3/raw/'

image_rows = 384
image_cols = 384


def create_train_data():
    train_data_path = os.path.join(data_path, 'train')
    images = os.listdir(train_data_path)
    total = len(images) // 2

    imgs = np.ndarray((total, image_rows, image_cols), dtype=np.uint8)
    imgs_mask = np.ndarray((total, image_rows, image_cols), dtype=np.uint8)

    i = 0
    print('-'*30)
    print('Creating training images...')
    print('-'*30)
    for image_name in images:
        if 'mask' in image_name:
            continue
        image_mask_name = image_name.split('.')[0] + '_mask.png'
        # img = imread(os.path.join(train_data_path, image_name), as_grey=True)
        # img_mask = imread(os.path.join(train_data_path, image_mask_name), as_grey=True)
```

```
        #
        # 2020May15 Google COlab
        img = imread(os.path.join(train_data_path, image_name), as_gray=True)
        img_mask = imread(os.path.join(train_data_path, image_mask_name),
as_gray=True)

        img = np.array([img])
        img_mask = np.array([img_mask])

        imgs[i] = img
        imgs_mask[i] = img_mask

        if i % 100 == 0:
            print('Done: {0}/{1} images'.format(i, total))
        i += 1
    print('Loading done.')

    np.save('imgs_train.npy', imgs)
    np.save('imgs_mask_train.npy', imgs_mask)
    print('Saving to .npy files done.')


def load_train_data():
    imgs_train = np.load('imgs_train.npy')
    imgs_mask_train = np.load('imgs_mask_train.npy')
    return imgs_train, imgs_mask_train


def create_validate_data():
    validate_data_path = os.path.join(data_path, 'validate')
    images = os.listdir(validate_data_path)
    total = len(images) // 2

    imgs = np.ndarray((total, image_rows, image_cols), dtype=np.uint8)
    imgs_mask = np.ndarray((total, image_rows, image_cols), dtype=np.uint8)

    i = 0
    print('-'*30)
    print('Creating validation images...')
    print('-'*30)
    for image_name in images:
        if 'mask' in image_name:
            continue
        ##OK## image_mask_name = image_name.split('.')[0] + '_mask.tiff'
        ##OK## image_mask_name = image_name.split('.')[0] + '_mask.png'
        image_mask_name = image_name.split('.')[0] + '_mask.tiff'
```

```
        img_val_id = int(image_name.split('.')[0])
        #img = imread(os.path.join(validate_data_path, image_name), as_grey=True)
        #img_mask = imread(os.path.join(validate_data_path, image_mask_name), as_grey=True)
        #
        # 2020May15 Google COlab
        img = imread(os.path.join(validate_data_path, image_name), as_gray=True)
        img_mask = imread(os.path.join(validate_data_path, image_mask_name), as_gray=True)

        img = np.array([img])
        img_mask = np.array([img_mask])

        imgs[i] = img
        imgs_mask[i] = img_mask

        if i % 10 == 0:
            print('Done: {0}/{1} images'.format(i, total))
        i += 1
    print('Loading done.')

    np.save('imgs_validate.npy', imgs)
    np.save('imgs_mask_validate.npy', imgs_mask)
    print('Saving to .npy files done.')


def load_validate_data():
    imgs_validate = np.load('imgs_validate.npy')
    imgs_mask_validate = np.load('imgs_mask_validate.npy')
    return imgs_validate, imgs_mask_validate


def create_test_data():
    test_data_path = os.path.join(data_path, 'test')
    images = os.listdir(test_data_path)
    total = len(images) // 2

    imgs = np.ndarray((total, image_rows, image_cols), dtype=np.uint8)
    imgs_mask = np.ndarray((total, image_rows, image_cols), dtype=np.uint8)
    imgs_id = np.ndarray((total, ), dtype=np.int64)

    i = 0
    print('-'*30)
    print('Creating testing images...')
    print('-'*30)
    for image_name in images:
```

```
        if 'mask' in image_name:
            continue
        image_mask_name = image_name.split('.')[0] + '_mask.tiff'
        ##OK## image_mask_name = image_name.split('.')[0] + '_mask.png'
        img_id = int(image_name.split('.')[0])
        #img = imread(os.path.join(test_data_path, image_name), as_grey=True)
        #img_mask = imread(os.path.join(test_data_path, image_mask_name), as_grey=True)
        #
        # 2020May15 Google COlab
        img = imread(os.path.join(test_data_path, image_name), as_gray=True)
        img_mask = imread(os.path.join(test_data_path, image_mask_name), as_gray=True)

        img = np.array([img])
        img_mask = np.array([img_mask])

        imgs[i] = img
        imgs_mask[i] = img_mask
        imgs_id[i] = img_id

        if i % 10 == 0:
            print('Done: {0}/{1} images'.format(i, total))
        i += 1
    print('Loading done.')

    np.save('imgs_test.npy', imgs)
    np.save('imgs_mask_test.npy', imgs_mask)
    np.save('imgs_id_test.npy', imgs_id)
    print('Saving to .npy files done.')


def load_test_data():
    imgs_test = np.load('imgs_test.npy')
    imgs_mask_test = np.load('imgs_mask_test.npy')
    imgs_id = np.load('imgs_id_test.npy')
    return imgs_test, imgs_id, imgs_mask_test

if __name__ == '__main__':
    create_train_data()
    create_validate_data()
    create_test_data()
```

References

[1] Ambellan, F., et al. (2019). "Automated segmentation of knee bone and cartilage combining statistical shape knowledge and convolutional neural networks: Data from the Osteoarthritis Initiative." Medical image analysis 52: 109-118.

[2] Aprovitola, A. and L. Gallo (2016). "Knee bone segmentation from MRI: A classification and literature review." Biocybernetics and Biomedical Engineering 36(2): 437-449.

[3] Berthelot, D., et al. (2017). "BEGAN: Boundary equilibrium generative adversarial networks." arXiv preprint arXiv:1703.10717.

[4] Bezdek, J. C., Hall, L. O., & Clarke, L. (1993). Review of MR image segmentation techniques using pattern recognition. MEDICAL PHYSICS-LANCASTER PA-, 20, 1033-1033.

[5] Bowes, M. A., et al. (2015). "A novel method for bone area measurement provides new insights into osteoarthritis and its progression." Annals of the rheumatic diseases 74(3): 519-525.

[6] Centers for Disease Control and Prevention. 2003 National Health Interview Survey; 2030 Census projected population. Available at https://www.cdc.gov/arthritis/data_statistics/national-statistics.html. Accessed June 19, 2018.

[7] Cheung, W. and G. Hamarneh (2007). N-sift: N-dimensional scale invariant feature transform for matching medical images. 2007 4th IEEE International Symposium on Biomedical Imaging: From Nano to Macro, IEEE.

[8] Chu, C., et al. (2013). Multi-organ segmentation based on spatially-divided probabilistic atlas from 3D abdominal CT images. International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer.

[9] Coleman, G. B., & Andrews, H. C. (1979). Image segmentation by clustering. Proceedings of the IEEE, 67(5), 773-785.

[10] Dalvi R, Abugharbieh R, Wilson D, Wilson DR. Multicontrast MR for enhanced bone imaging and segmentation. Proceedings of IEEE International Conference Engineering in Medicine & Biology Society; 2007. p. 5620–3.

[11] Deniz, C. M., Xiang, S., Hallyburton, S., Welbeck, A., Honig, S., Cho, K., & Chang, G. (2017). Segmentation of the Proximal Femur from MR Images using Deep Convolutional Neural Networks. *arXiv preprint arXiv:1704.06176*.

[12] Ding, Y., et al. (2019). "A stacked multi-connection simple reducing net for brain tumor segmentation." IEEE Access 7: 104011-104024.

[13] Do, N.-T., et al. (2019). Knee Bone Tumor Segmentation from radiographs using Seg-Unet with Dice Loss. Proc. of International Workshop on Frontiers in Computer Vision (IWFCV 2019).

[14] Du, Y., Almajalid, R., Shan, J., & Zhang, M. (2018). A Novel Method to Predict Knee Osteoarthritis Progression on MRI Using Machine Learning Methods. *IEEE Transactions on NanoBioscience*.

[15] Du, Y., et al. (2017). Knee osteoarthritis prediction on MR images using cartilage damage index and machine learning methods. Bioinformatics and Biomedicine (BIBM), 2017 IEEE International Conference on, IEEE.

[16] Du, Y., et al. (2018). "A Novel Method to Predict Knee Osteoarthritis Progression on MRI Using Machine Learning Methods." IEEE Transactions on NanoBioscience.

[17] Dunn, J. C. (1973). A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters..

[18] Eckstein, F., Cicuttini, F., Raynauld, J. P., Waterton, J. C., & Peterfy, C. (2006). Magnetic resonance imaging (MRI) of articular cartilage in knee osteoarthritis (OA): morphological assessment. Osteoarthritis and cartilage, 14, 46-75.

[19] Eckstein, F., et al. (2002). "Long-term and resegmentation precision of quantitative cartilage MR imaging (qMRI)." Osteoarthritis and cartilage 10(12): 922-928.

[20] El Jurdi, R., et al. (2020). "BB-UNet: U-Net with Bounding Box Prior." IEEE Journal of Selected Topics in Signal Processing.

[21] Enokiya, Y., et al. (2018). "Automatic Liver Segmentation Using U-Net with Wasserstein GANs." Journal of Image and Graphics 6(2).

[22] Everingham, M. and J. Winn (2011). "The pascal visual object classes challenge 2012 (VOC2012) development kit." Pattern Analysis, Statistical Modelling and Computational Learning, Tech. Rep 8.

[23] Fan, X., Guo, H., Zheng, K., Feng, W., & Wang, S. (2018). Object Detection with Mask-based Feature Encoding. *arXiv preprint arXiv:1802.03934*.

[24] Freeman, W. T., & Roth, M. (1995, June). Orientation histograms for hand gesture recognition. In International workshop on automatic face and gesture recognition (Vol. 12, pp. 296-301).

[25] Fripp J, Crozier S, Warfield SK, Ourselin S (2007) Automatic segmentation of the bone and extraction of the bone–cartilage interface from magnetic resonance images of the knee. Phys Med Biol 52:1617–1631. https ://doi.org/10.1088/0031-9155/52/6/005

[26] Fripp, J., et al. (2007). "Automatic segmentation of the bone and extraction of the bone–cartilage interface from magnetic resonance images of the knee." Physics in Medicine & Biology 52(6): 1617.

[27] Fripp, J., et al. (2009). Automated segmentation of the menisci from MR images. 2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro, IEEE.

[28] Fukushima, K. (1975). "Cognitron: A self-organizing multilayered neural network." Biological cybernetics 20(3-4): 121-136.

[29] Fukushima, K. (1981). "Neocognitron--a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position." NHK 放送科学基礎研究所報告(15): p106-115.

[30] Fukushima, K. and N. Wake (1992). Improved Neocognitron with bend-detecting cells. IEEE/INNS International Joint Conference on Neural Networks, IEEE Press.

[31] Gaj, S., et al. (2020). "Automated cartilage and meniscus segmentation of knee MRI with conditional generative adversarial networks." Magnetic resonance in medicine 84(1): 437-449.

[32] Gan, H. S., Ramlee, M. H., Wahab, A. A., Lee, Y. S., & Shimizu, A. (2020). From classical to deep learning: review on cartilage and bone segmentation techniques in knee osteoarthritis research. Artificial Intelligence Review, 1-50.

[33] Giebel, H. (1971). Feature extraction and recognition of handwritten characters by homogeneous layers. Zeichenerkennung durch biologische und technische systeme/Pattern recognition in biological and technical systems, Springer: 162-169.

[34] Girshick, R. (2015). "Fast R-CNN." arXiv preprint arXiv:1504.08083.

[35] Girshick, R., et al. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition.

[36] Girshick, R., et al. (2015). "Region-based convolutional networks for accurate object detection and segmentation." IEEE transactions on pattern analysis and machine intelligence 38(1): 142-158.

[37] Gonzalez, R. C., & Woods, R. E. (2008). Digital Image Processing 4th Edition Pearson Prentice Hall.

[38] Goodfellow, I., et al. (2014). Generative Adversarial Nets (GAN). Advances in neural information processing systems.

[39] Gornale, S. S., Patravali, P. U., & Manza, R. R. A Survey on Exploration and Classification of Osteoarthritis Using Image Processing Techniques. International Journal of Scientific and Engineering Research (IJSER), ISSN, 2229-5518.

[40] Grau, V., Mewes, A. U. J., Alcaniz, M., Kikinis, R., & Warfield, S. K. (2004). Improved watershed transforms for medical image segmentation using prior information. IEEE transactions on medical imaging, 23(4), 447-458.

[41] Guo, Y., et al. (2011). Distribution-based active contour model for medical image segmentation. 2011 Sixth International Conference on Image and Graphics, IEEE.

[42] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017, October). Mask R-CNN. In *Computer Vision (ICCV), 2017 IEEE International Conference on* (pp. 2980-2988). IEEE.

[43] He, P. and J. Zheng (2001). Segmentation of tibia bone in ultrasound images using active shape models. 2001 Conference Proceedings of the 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society, IEEE.

[44] Heidarkhan-Tehrani, A., Singh, S., Xiao, Y., & Oloyede, A. (2012). Anisotropy of articular cartilage reflects the ECM gradient architecture: Hough-Radon Transform

Analysis. In Proceedings of the IASTED International Symposia on Imaging and Signal Processing in Health Care and Technology (ISPHT 2012) (pp. 64-70).

[45] Heimann, T., et al. (2010). Segmentation of knee images: a grand challenge. Proc. MICCAI Workshop on Medical Image Analysis for the Clinic.

[46] Ho, N.-H., et al. (2019). "Regenerative Semi-Supervised Bidirectional W-Network-Based Knee Bone Tumor Classification on Radiographs Guided by Three-Region Bone Segmentation." IEEE Access 7: 154277-154289.

[47] Hodge, J. A., Mishra, K. V., & Zaghloul, A. I. (2019, October). Joint multi-layer GAN-based design of tensorial RF metasurfaces. In 2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP) (pp. 1-6). IEEE..

[48] http://www.image-net.org/challenges/LSVRC/2012/index. ILSVRC2012 datasets. Accessed on August 12th 2019.

[49] Hubel, D. H. and T. N. Wiesel (1962). "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex." The Journal of physiology 160(1): 106.

[50] Işın, A., et al. (2016). "Review of MRI-based brain tumor image segmentation using deep learning methods." Procedia Computer Science 102: 317-324.

[51] Jin, Q., et al. (2018). "RA-UNet: A hybrid deep attention-aware network to extract liver and tumor in CT scans." arXiv preprint arXiv:1811.01328.

[52] Johnson, J. M. (2013). "Analysis, Segmentation and Prediction of Knee Cartilage using Statistical Shape Models."

[53] Kashyap, S., et al. (2017). "Learning-based cost functions for 3-D and 4-D multi-surface multi-object segmentation of knee MRI: Data from the osteoarthritis initiative." IEEE transactions on medical imaging 37(5): 1103-1113.

[54] Kass, M., et al. (1988). "Snakes: Active contour models." International journal of computer vision 1(4): 321-331.

[55] Ke, Y. and R. Sukthankar (2004). PCA-SIFT: A more distinctive representation for local image descriptors. Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004., IEEE.

[56] Kellgren, J. and J. Lawrence (1957). "Radiological assessment of osteo-arthrosis." Annals of the rheumatic diseases 16(4): 494.

[57] Kitney, R., et al. (1998). Fast automated segmentation and visualisation methods for MR images of the knee joint in arthritis. Proceedings of the 20th Annual International Conference of the IEEE Engineering in Medicine and Biology Society. Vol. 20 Biomedical Engineering Towards the Year 2000 and Beyond (Cat. No. 98CH36286), IEEE.

[58] LeCun, Y., et al. (1995). "Learning algorithms for classification: A comparison on handwritten digit recognition." Neural networks: the statistical mechanics perspective 261: 276.

[59] LeCun, Y., Jackel, L. D., Bottou, L., Brunot, A., Cortes, C., Denker, J., ... & Simard, P. (1995, October). Comparison of learning algorithms for handwritten digit recognition. In International conference on artificial neural networks (Vol. 60, pp. 53-60).

[60] Lee, H., et al. (2018). BCD-NET: A novel method for cartilage segmentation of knee MRI via deep segmentation networks with bone-cartilage-complex modeling. Biomedical Imaging (ISBI 2018), 2018 IEEE 15th International Symposium on, IEEE..

[61] Li, R., Liu, W., Yang, L., Sun, S., Hu, W., Zhang, F., & Li, W. (2017). DeepU-Net: A Deep Fully Convolutional Network for Pixel-level Sea-Land Segmentation. *arXiv preprint arXiv:1709.00201*.

[62] Liu, F. (2019). "SUSAN: segment unannotated image structure using adversarial network." Magnetic resonance in medicine 81(5): 3330-3345.

[63] Liu, F., et al. (2018). "Deep convolutional neural network and 3D deformable approach for tissue segmentation in musculoskeletal magnetic resonance imaging." Magnetic resonance in medicine 79(4): 2379-2391.

[64] Liu, F., et al. (2018). "Deep learning approach for evaluating knee MR images: achieving high diagnostic performance for cartilage lesion detection." Radiology 289(1): 160-169.

[65] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). SSD: Single Shot Multibox Detector. In *European conference on computer vision* (pp. 21-37). Springer, Cham.

[66] Liu, W., et al. (2016). SSD: Single Shot Multibox Detector. European conference on computer vision, Springer.

[67] Long, J., et al. (2015). Fully convolutional networks for semantic segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition.

[68] Lowe, D. G. (2004). "Distinctive image features from scale-invariant keypoints." International journal of computer vision 60(2): 91-110.

[69] Ma, L., Shuai, R., Ran, X., Liu, W., & Ye, C. (2020). Combining DC-GAN with ReS-UNet for blood cell image classification. Medical & Biological Engineering & Computing, 1-14.

[70] Mahendrakar, P., et al. (2018). A Survey on Morphological Assessment of Knee Articular Cartilage from MR Images. 2018 International Conference on Smart Systems and Inventive Technology (ICSSIT), IEEE.

[71] Man, G. and G. Mologhianu (2014). "Osteoarthritis pathogenesis–a complex process that involves the entire joint." Journal of medicine and life 7(1): 37.

[72] Man, G. and G. Mologhianu (2014). "Osteoarthritis pathogenesis–a complex process that involves the entire joint." Journal of medicine and life 7(1): 37.

[73] Mao, X., et al. (2017). Least Squares Generative Adversarial Networks. Proceedings of the IEEE International Conference on Computer Vision.

[74] Marques, J. (2012). Osteoarthritis imaging by quantification of tibial trabecular bone, University of Copenhagen, Faculty of Science, Department of Computer Science.

[75] Mazurowski, M. A., et al. (2018). "Deep learning in radiology: an overview of the concepts and a survey of the state of the art." arXiv preprint arXiv:1802.08717.

[76] Memari, N. and M. Moghbel (2020). Computer-Aided Diagnosis (CAD) of Knee Osteoarthritis based on Magnetic Resonance Imaging for Quantitative Pathogenesis Analysis and Visualization. 2020 IEEE 10th Symposium on Computer Applications & Industrial Electronics (ISCAIE), IEEE.

[77] Menze, B. H., et al. (2014). "The multimodal brain tumor image segmentation benchmark (BRATS)." IEEE transactions on medical imaging 34(10): 1993-2024.

[78] Millington, S., Li, K., Wu, X., Hurwitz, S., & Sonka, M. (2005). Automated simultaneous 3D segmentation of multiple cartilage surfaces using optimal graph searching on MRI images. Osteoarthritis Cartilage, 13(Suppl A), S130.

[79] Minsky, M. and S. Papert (1969). "An introduction to computational geometry." Cambridge tiass., HIT.

[80] Nagaosa, Y., P. Lanyon, and M. Doherty, Characterization of size and direction of osteophyte in knee osteoarthritis: a radiographic study. Annals of the rheumatic diseases, 2002. 61(4): p. 319-324.

[81] Ning, Y., et al. (2018). Automated Pancreas Segmentation Using Recurrent Adversarial Learning. 2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), IEEE.

[82] Norman, B., Pedoia, V., & Majumdar, S. (2018). Use of 2D U-Net Convolutional Neural Networks for Automated Cartilage and Meniscus Segmentation of Knee MR Imaging Data to Determine Relaxometry and Morphometry. *Radiology*, 172322.

[83] Pang, J., Driban, J. B., Destenaves, G., Miller, E., Lo, G. H., Ward, R. J., ... & McAlindon, T. E. (2013). Quantification of bone marrow lesion volume and volume change using semi-automated segmentation: data from the osteoarthritis initiative. BMC musculoskeletal disorders, 14(1), 1.

[84] Peake, E., et al. (2020). "Ensemble learning for robust knee cartilage segmentation: data from the osteoarthritis initiative." BioRxiv.

[85] Peterfy, C., et al. (2004). "Whole-organ magnetic resonance imaging score (WORMS) of the knee in osteoarthritis." Osteoarthritis and cartilage 12(3): 177-190.

[86] Pham DL, Xu C, Prince JL. Current methods in medical image segmentation. Annu Rev Biomed Eng 2000;2:315–37.

[87] Prasoon, A., Petersen, K., Igel, C., Lauze, F., Dam, E., & Nielsen, M. (2013, September). Deep feature learning for knee cartilage segmentation using a triplanar convolutional neural network. In *International conference on medical image computing and computer-assisted intervention* (pp. 246-253). Springer, Berlin, Heidelberg.

[88] Raj, A., et al. (2018). Automatic knee cartilage segmentation using fully volumetric convolutional neural networks for evaluation of osteoarthritis. Biomedical Imaging (ISBI 2018), 2018 IEEE 15th International Symposium on, IEEE.

[89] Redmon, J. and A. Farhadi (2016). "YOLO9000: Better, faster, stronger. arXiv 2016." arXiv preprint arXiv:1612.08242.

[90] Redmon, J. and A. Farhadi (2017). YOLO9000: better, faster, stronger. Proceedings of the IEEE conference on computer vision and pattern recognition.

[91] Redmon, J. and A. Farhadi (2018). "Yolov3: An incremental improvement." arXiv preprint arXiv:1804.02767.

[92] Ren, S., et al. (2017). "Faster R-CNN: towards real-time object detection with region proposal networks." IEEE transactions on pattern analysis and machine intelligence 39(6): 1137-1149.

[93] Ronneberger, O., et al. (2015). U-net: Convolutional networks for biomedical image segmentation. International Conference on Medical image computing and computer-assisted intervention, Springer.

[94] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, 65(6), 386.

[95] Rosenblatt, F. (1962). "Principles of neurodynamics: Perceptions and the theory of brain mechanisms."

[96] Roth, H. R., et al. (2018). "Spatial aggregation of holistically-nested convolutional neural networks for automated pancreas localization and segmentation." Medical image analysis 45: 94-10

[97] Schmid, J. and N. Magnenat-Thalmann (2008). MRI bone segmentation using deformable models and shape priors. International conference on medical image computing and computer-assisted intervention, Springer.

[98] Shan, L., et al. (2014). "Automatic atlas-based three-label cartilage segmentation from MR knee images." Medical image analysis 18(7): 1233-1246.

[99] Shim, H., Kwoh, C. K., Yun, I. D., Lee, S. U., & Bae, K. (2009, March). Simultaneous 3D segmentation of three bone compartments on high resolution knee MR images from osteoarthritis initiative (OAI) using graph cuts. In Medical Imaging 2009: Image Processing (Vol. 7259, p. 72593P). International Society for Optics and Photonics.

[100] Stammberger, T., Eckstein, F., Englmeier, K. H., & Reiser, M. (1999). Determination of 3 D cartilage thickness data from MR imaging: Computational method and reproducibility in the living. Magnetic resonance in medicine, 41(3), 529-536.

[101] Sun Y, Teo EC, Zhang QH. Discussions of Knee joint segmentation. International Conference on Biomedical and Pharmaceutical Engineering-ICBPE; 2006.

[102] Tamez-Pena, J. G., et al. (2012). "Unsupervised segmentation and quantification of anatomical knee features: data from the Osteoarthritis Initiative." IEEE Transactions on Biomedical Engineering 59(4): 1177-1186.

[103] Tamez-Peña, J., et al. (2011). Atlas based method for the automated segmentation and quantification of knee features: Data from the osteoarthritis initiative. 2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro, IEEE.

[104] Tan, C., Zhao, L., Yan, Z., Li, K., Metaxas, D., & Zhan, Y. (2018, April). Deep multi-task and task-specific feature learning network for robust shape preserved organ segmentation. Biomedical Imaging (ISBI 2018), 2018 IEEE 15th International Symposium on, IEEE.

[105] Tappert, C.C. (2019). "Who is the Father of Deep Learning?" Proc. 6th Annual Conf. Comp. Science & Comp. Intelligence: Artificial Intelligence (CSCI-ISAI).

[106] Tianhu, L., & Sewchand, W. (1992). Statistical approach to X-ray CT imaging and its applications in image analysis. II. A new stochastic model-based image segmentation technique for X-ray CT image. IEEE Transactions on Medical Imaging, 11(1), 62-69.

[107] Tong, T., et al. (2015). "Discriminative dictionary learning for abdominal multi-organ segmentation." Medical image analysis 23(1): 92-104.

[108] Wang, G., Li, W., Zuluaga, M. A., Pratt, R., Patel, P. A., Aertsen, M., ... & Vercauteren, T. (2018). Interactive medical image segmentation using deep learning with image-specific fine-tuning. *IEEE Transactions on Medical Imaging*.

[109] Wang, Q., et al. (2013). Semantic context forests for learning-based knee cartilage segmentation in 3D MR images. International MICCAI Workshop on Medical Computer Vision, Springer.

[110] Wang, Y.-Y., et al. (2013). "A scale invariant feature transform based method." Journal of Information Hiding and Multimedia Signal Processing 4(2): 73-89.

[111] Ward, I. R., Laga, H., & Bennamoun, M. (2019). RGB-D image-based Object Detection: From traditional methods to deep learning techniques. In RGB-D Image Analysis and Processing (pp. 169-201). Springer, Cham.

[112] William, F. T., & Roth, M. (1995, June). Orientation histograms for hand gesture recognition. In International workshop on automatic face and gesture recognition (Vol. 12, pp. 296-301).

[113] Williams, T. G., et al. (2010). Automatic segmentation of bones and inter-image anatomical correspondence by volumetric statistical modelling of knee MRI. 2010 IEEE International Symposium on Biomedical Imaging: From Nano to Macro, IEEE.

[114] Williams, T. G., Holmes, A. P., Bowes, M., Vincent, G., Hutchinson, C. E., Waterton, J. C., ... & Taylor, C. J. (2014). Measurement and visualization of focal

cartilage thickness change by MRI in a study of knee osteoarthritis using a novel image analysis tool. The British journal of radiology.

[115] Wu, C. W., Morrell, M. R., Heinze, E., Concoff, A. L., Wollaston, S. J., Arnold, E. L., ... & Moreland, L. W. (2005, December). Validation of American College of Rheumatology classification criteria for knee osteoarthritis using arthroscopically defined cartilage damage scores. In Seminars in arthritis and rheumatism (Vol. 35, No. 3, pp. 197-201). WB Saunders.

[116] Wu, D., Sofka, M., Birkbeck, N., & Zhou, S. K. (2014, September). Segmentation of multiple knee bones from CT for orthopedic knee surgery planning. In International Conference on Medical Image Computing and Computer-Assisted Intervention (pp. 372-380). Springer, Cham.

[117] Wu, J., et al. (2013). "A Comparative Study of SIFT and its Variants." Measurement science review 13(3): 122-131.

[118] Xue, Y., et al. (2018). "SeGAN: Adversarial network with multi-scale l 1 loss for medical image segmentation." Neuroinformatics 16(3-4): 383-392.

[119] Yan, W., et al. (2019). The Domain Shift Problem of Medical Image Segmentation and Vendor-Adaptation by Unet-GAN. International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer.

[120] Yang, R. and R. Yang (2014). Action segmentation and recognition based on depth hog and probability distribution difference. International Conference on Intelligent Computing, Springer.

[121] Yin, M. (2016). MRI software measurement of osteophyte volume in knee osteoarthritis: a longitudinal validation study.

[122] Zhengrong. L., MacFall, J. R., & Harrington, D. P. (1994). Parameter estimation and tissue segmentation from multispectral MR images. IEEE transactions on Medical Imaging, 13(3), 441-449.

[123] Zeng, G. and G. Zheng (2018). Multi-stream 3D FCN with multi-scale deep supervision for multi-modality isointense infant brain MR image segmentation. Biomedical Imaging (ISBI 2018), 2018 IEEE 15th International Symposium on, IEEE.

[124] Zhang, M., Driban, J. B., Price, L. L., Lo, G. H., & McAlindon, T. E. (2015). Magnetic Resonance Image Sequence Influences the Relationship between Bone Marrow Lesions Volume and Pain: Data from the Osteoarthritis Initiative. BioMed research international, 2015.

[125] Zhou, Z., et al. (2018). Unet++: A nested u-net architecture for medical image segmentation. Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support, Springer: 3-11.

[126] Zhou, Z., et al. (2019). "Unet++: Redesigning skip connections to exploit multiscale features in image segmentation." IEEE transactions on medical imaging 39(6): 1856-1867.

[127] Zhu, Y., et al. (2020). Machine Learning Approaches for Cancer Bone Segmentation from Micro Computed Tomography Images. 2020 IEEE 23rd International Conference on Information Fusion (FUSION), IEEE.