Workload Proxies and Their Optimization for Healthcare Analytics

Major Project Report

Submitted in fulfillment of the requirements
for the degree of

Master of Technology
in
Electronics & Communication Engineering
(Embedded Systems)

By

Abhi Panchal

(17MECE11)

Electronics & Communication Engineering Department
Institute of Technology
Nirma University
Ahmedabad-382 481

May-2019

Workload Proxies and Their Optimization for Healthcare Analytics

Major Project Report

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology

in

Electronics & Communication Engineering

By

Abhi Panchal
(17MECE11)

Under the guidance of

External Project Guide:

Dr. Ramanathan Sethuraman

Silicon Architecture Engineer

Intel Technologies Pvt. Ltd.,

Bengaluru.

Internal Project Guide:

Dr. Nagendra Gajjar

Professor & PG Coordinator, Embedded Systems,

Institute of Technology,

Nirma University, Ahmedabad.

Electronics & Communication Engineering Department

Institute of Technology, Nirma University

Ahmedabad-382 481

May-2019

Declaration

This is to certify that

a. The thesis comprises my original work towards the degree of Master of Technology in Embedded Systems at Nirma University and has not been submitted elsewhere for a degree.

b. Due acknowledgment has been made in the text to all other material used.

- Abhi Panchal

17MECE11

Disclaimer

"The content of this thesis does not represent the technology, opinions, beliefs, or positions of Intel Technology India Pvt. Ltd., its employees, vendors, customers, or associates."

Certificate

This is to certify that the Major Project entitled **"Workload Proxies and Their Optimization for Healthcare Analytics"** submitted by **Abhi Panchal (17MECE11)**, towards the partial fulfillment of the requirements for the degree of Master of Technology in Embedded Systems, Nirma University, Ahmedabad is the record of work carried out by him under our supervision and guidance. In our opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of our knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Date: Place: Ahmedabad

Dr. N. P. Gajjar
Internal Guide

Dr. N. P. Gajjar
Program Coordinator

Dr. D. K. Kothari
Head, EC Department

Dr. Alka Mahajan
Director, ITNU

Certificate

This is to certify that the Major Project entitled **"Workload Proxies and Their Optimization for Healthcare Analytics"** submitted by **Abhi Panchal (17MECE11)**, towards the partial fulfillment of the requirements for the degree of Master of Technology in Embedded Systems, Nirma University, Ahmedabad is the record of work carried out by him under my supervision and guidance at **Intel Technology India Pvt. Ltd.** In my opinion, the submitted work has reached a level required for being accepted for examination.

Dr. Ramanathan Sethuraman
Silicon Architecture Engineer
Intel Technology India Pvt. Ltd.
Bengaluru.

Date: Place: Bengaluru

Statement of Originality

I, **Abhi Panchal**, Roll No. **17MECE11**, give undertaking that the Project Report on "**Workload Proxies and Their Optimization for Healthcare Analytics**" submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in **Electronics and communication (Embedded System)** of Institute of Technology, Nirma University, Ahmedabad, contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me as part of on-going research work in Intel India Technology Pvt. Ltd. and I give assurance that no attempt of plagiarism has been made. It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

Date:

Place:

Endorsed by

Dr. Nagendra Gajjar

Acknowledgements

I take this opportunity to express my profound gratitude and regards to my internship project guide **Dr. Nagendra Gajjar** for his exemplary guidance, monitoring and constant encouragement.

I express my gratitude to the staff of **Intel Technologies Pvt. Ltd., Bengaluru** for providing most encouraging working environment along with providing their valuable time and effort during implementation of this project.

I would like to express sincere gratitude to manager **Mr. Dheemanth Nagaraj**, Intel Fellow, Intel India Pvt. Ltd., for giving me an opportunity to work for Intel. I would also like to thank mentor **Dr. Ramanathan Sethuraman**, Silicon Architecture Engineer, Intel India Pvt. Ltd. for his encouragement and guidance. I would also like to cordially thank my buddy **Madhu Kumar**, System on a Chip (SoC) Design Engineer, Intel India Pvt. Ltd. for his valuable advices and support throughout the project. I thank **H. Lokesh**, Technician: Engineering Laboratory, Intel India Pvt. Ltd. for his support in setting up the experimental setup in labs.

I thank my parants, faculty members and colleagues for their constant support and encouragement during this project work.

- Abhi Panchal

17MECE11

Abstract

Workload analysis plays important role in understanding the problems in the design and development of server systems. This will help the architect to make predictions of the system behavior of the upcoming system or SoC. Today market is moving towards the workloads related to healthcare image processing because healthcare is on high priority sector and people expect highest level of care and services regardless of cost. This work primarily concentrates on the creation, characterization and optimization of workload proxies like U-Net, Xception and DenseNet for healthcare image processing on Intel Xeon Server Platform. In this thesis, for image segmentation U-Net topology is used and for image classification Xception and DenseNet topologies are used as source of the workload. With the help of few Intel internal tools and silicon data from the present generation platform, study is made to understand the behavior of the workload in present generation platform. We are able to measure the performance of different platforms for different workloads, to identify the bottlenecks in the performance of existing platforms or existing software, to find the reason for this bottleneck and predict the possible solution. So that this analysis will be used to propose SoC architecture features and important optimizations needed to support a new class of workloads efficiently in next generation platforms or SoCs. Open Visual Inferencing and Neural Network Optimization (OpenVINO) and Multi-instance implementation provided significant performance boost. Central Processing Unit (CPU) and Graphics Processing Unit (GPU) performance comparison results are also discussed in the last section. These workload proxies and performance numbers will be provided as proof of concept to core architects.

Abbreviation Notation and Nomenclature

OpenVINO Open Visual Inferencing and Neural Network Optimization

MKL-DNN Math Kernel Library for Deep Neural Networks

ACPI Advanced Configuration and Power Interface

U-Net-TC U-Net using Transposed Convolution

ASIC Application Specified Integrated Circuit

OPCM Open Performance Counter Monitor

API Application Programming Interface

FPGA Field Programmable Gate Array

U-Net-US U-Net using Up-Sampling

CNN Convolutional Neural Network

PCM Performance Counter Monitor

AVX Advanced Vector Extensions

GPU Graphics Processing Unit

CPU Central Processing Unit

SPS Studies Per Seconds

SUT System Under Test

PoC Proof of Concept

SoC System on a Chip

CV Computer Vision

I/O Input/Output

List of Figures

Contents

Chapter 1

Introduction

1.1 Motivation

Workload analysis plays critical role in understanding the problems in design and development of server systems.This will help the architect to make predictions of the system behavior of the upcoming system or SoCs.

Real workloads are not readily available. Need to create workload proxies which represents the real workload in terms of compute, bandwidth and Input/Output (I/O) operation becomes essential. It is important to predict future devices or SoCs and their associated workloads in understanding the target market. Here Workload Proxies plays a critical role. The workload proxies have been used in industry to benchmark current and future generation CPU/GPU platforms.

1.2 Objective

The main objectives of the project are as follows:

- Create, characterize and optimize the deep learning workload proxies for healthcare on Intel Xeon Server Platform.
- Proxies involves U-Net with up-sampling, Xception, U-Net with transposed

convolution, 3D U-Net with transposed convolution and DenseNet.

- Identify the performance bottlenecks in the existing platforms.
- Perform required software optimization.
- Provide as Proof of Concept (PoC) to core architects.

1.3 Requirements

The development and implementation of this project requires following:

- Knowledge of workload/topologies
- Understanding of processor performance metrics
- Knowledge of profiling tools
- Shell scriping
- Python scripting

1.4 Scope of Work

The main objective is to characterize and analyze the workload proxies and come up with the optimized solution which acts as a PoC to core architects and help them in defining new SoCs.

1.5 Gantt Charts

1.5.1 Project Work-Flow Phase-I

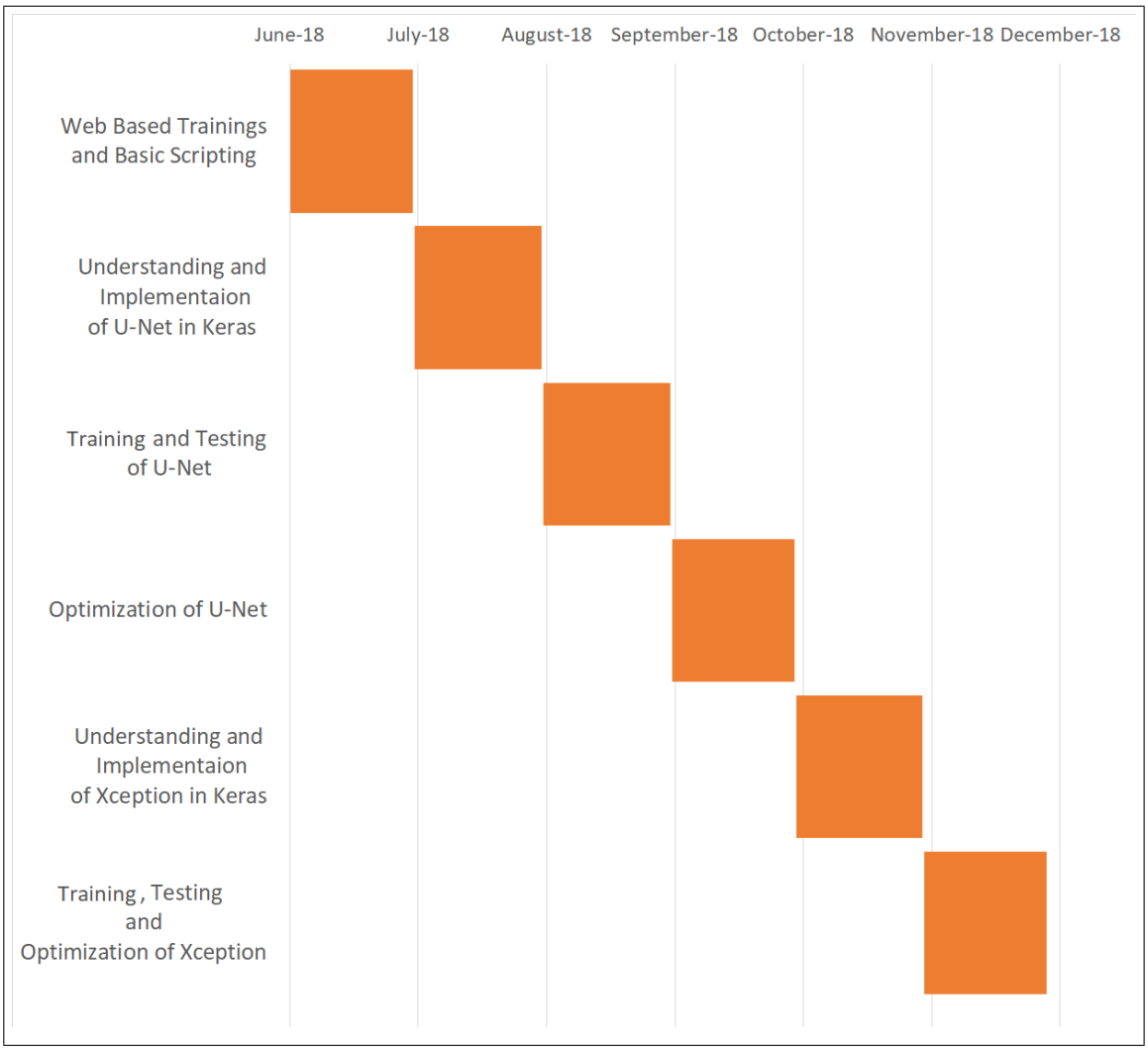

Figure 1.1: Project Work-Flow Phase-I

1.5.2 Project Work-Flow Phase-II

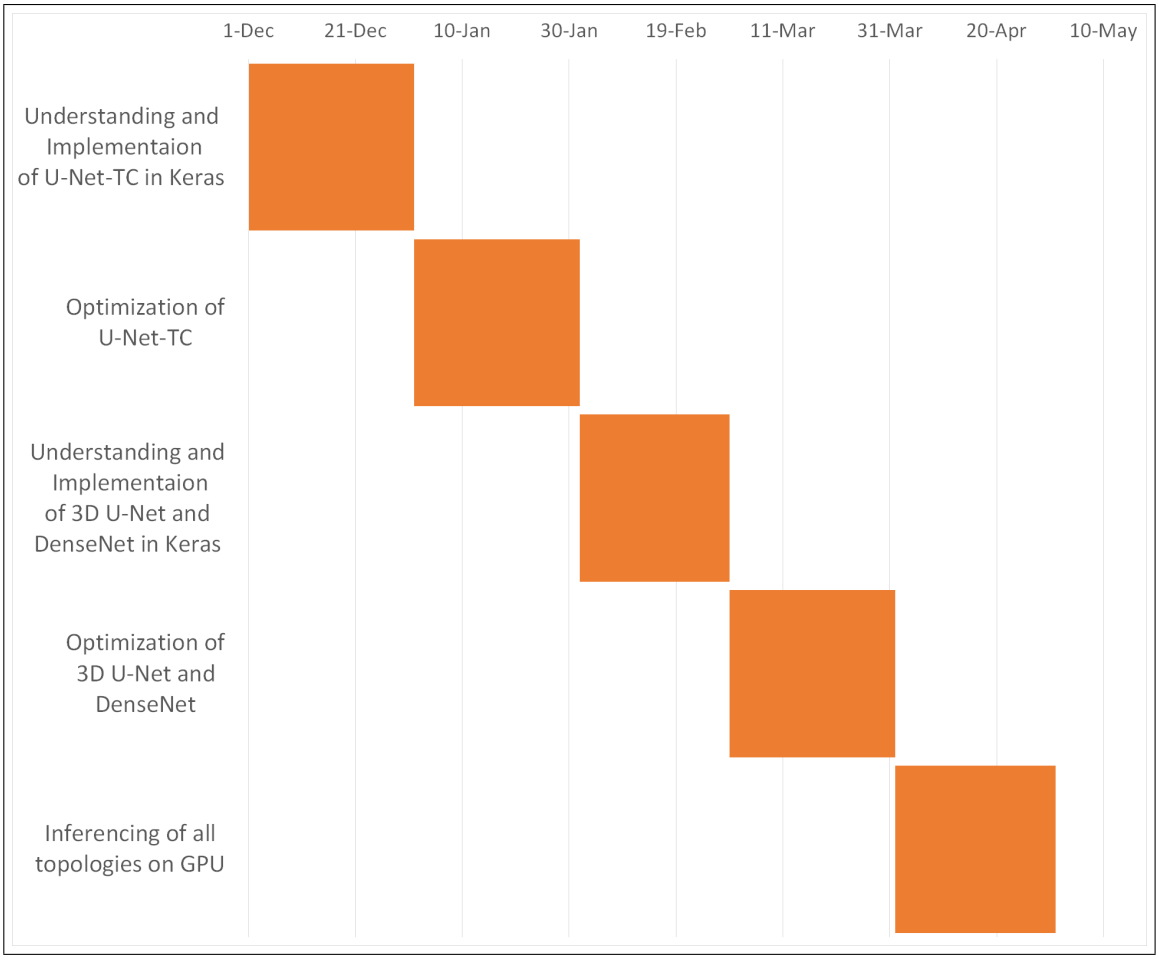

Figure 1.2: Project Work-Flow Phase-II

1.6 Thesis Outline

- **Chapter-1** contains the brief information about motivation and objective of the project along with the Gantt chart indicating project development work flow.
- **Chapter-2** describes the literature survey by providing the overview of workload, workload characterization and optimization.
- **Chapter-3** discusses about workload proxies.
- **Chapter-4** discusses about tools used and performance results of workload proxies.
- **Chapter-5** concludes the project report and discusses the future scope.

Chapter 2

Background Theory and Literature Survey

This chapter provides overview of workload characterization.

2.1 Background Theory and Literature Survey

Workload characterization becomes necessary before the silicon is ready, so that if any hardware changes are required those changes can be captured before silicon is manufactured. The study of workload characterization on the next generation platform before the actual silicon is available is of great importance. This helps to save cost and time.

These workloads are characterized in terms of CPU Utilization, CPU frequency, Core Scaling, Memory Bandwidth, I/O Bandwidth etc. It is necessary to understand CPU states in order to understand the behavior of any workload on any platform. CPU might get fully utilized even at maximum CPU frequency due to the high intensive workload. Similarly, a low profile workload utilizes minimum CPU frequency for a less fraction of time. This will also vary energy efficiency of the system and performance of the workload.

The workload should be repeatable in order to test multiple alternatives under

identical conditions. Generally a real-user environment is not repeatable and there is a need to study the real-user environments, observe and understand the key characteristics and develop a workload model, so that this model can be used repeatedly. This technique is called workload characterization. Once a workload characterization is done, the effect of changes in the workload and system can be studied in a controlled manner by simply changing the parameters. Workload characterization can be performed using several techniques like Averaging, Histogram etc[1].

The data measured from workload consist of the resource demands made on the system by user or number of users. The entity that makes the service requests at the System Under Test (SUT) is termed as user. The user may or may not be a human being. For example, if the processor is a SUT, then the users may be various batch jobs or programs. In workload characterization literature, the term workload unit or workload component is used instead of the user.

The workload characterization involves characterizing a typical workload unit or workload component or a user. The resource demands, service requests or measured quantity which is used to characterize the workload are called as workload features or workload parameters. Examples of workload parameters are CPU utilization, CPU frequency, memory bandwidth, I/O bandwidth, IPC, latency etc.

Machine Learning (ML) and Artificial Intelligence (AI) have progressed rapidly in recent years. Techniques of ML and AI have played important role in medical field like medical image processing, computer-aided diagnosis, image interpretation, image fusion, image registration, image segmentation, image-guided therapy, image retrieval and analysis Techniques of ML extract information from the images and represents information effectively and efficiently. The ML and AI facilitate and assist doctors that they can diagnose and predict accurate and faster the risk of diseases and prevent them in time. These techniques composed of deep learning algorithms like Support Vector Machine (SVM), Neural Network (NN), Convolutional Neural Network (CNN), Recurrent neural Network (RNN), Long Short term Memory (LSTM), Extreme Learning Model (ELM), Generative Adversarial Networks

(GANs) etc[2]. Models analyzed in this thesis are made up of CNN layers.

For optimizing the workload proxies, there are ways to optimize it at hardware level as well as software level. In this work, Intel Advanced Vector Extensions (AVX), Intel Math Kernel Library for Deep Neural Networks (MKL-DNN) and Model Optimizer are used for optimization.

Intel AVX is a set of instructions for doing Single Instruction Multiple Data (SIMD) operations on Intel architecture CPUs. Intel AVX is designed to support 512 or 1024 bits in future. Three-operand, nondestructive operations have been added in AVX. Memory alignment requirements for operands are relaxed. A few instructions take four-register operands, allowing smaller and faster code by removing unnecessary instructions[3].

The Intel MKL-DNN is a performance library for Deep Learning (DL) applications which is open source. It is intended for acceleration of DL frameworks on Intel architecture. It includes highly vectorized and threaded building blocks for implementation of Convolutional Neural Network (CNN)s and reccurent neural networks (RNNs) with C and C++ interfaces. The library provides optimized implementations for the most common computational functions (also called primitives) used in deep neural networks covering a wide range of applications, including image recognition, object detection, semantic segmentation, neural machine translation, and speech recognition[4].

Model optimization is explained in the section Model Optimizer. So by using AVX and MKL-DNN, it is possible to optimize the workload at hardware level as well as software level.

Chapter 3

Workload Proxies

3.1 U-Net

3.1.1 Introduction

In past few years, in several visual recognition tasks, the deep convolutional networks have outperformed the state of the art. The quintessential use of convolutional networks is on classification tasks, where the output to an image is a single class label. Though, in many visual tasks, chiefly in bio-medical image processing, the desired output should include localization, i.e., a class label is supposed to be assigned to each pixel. Furthermore, thousands of training images are usually beyond reach in bio-medical tasks.

The more elegant architecture is used here, which is the so-called/fully convolutional network. This architecture yields more precise segmentations with only using few training images. The main idea is to augment a usual contracting network by successive layers, where up-sampling operators are used. Therefore, the output resolution is increased by layers. Thus, the resolution of the output is increased by layers. So to localize, from the contracting path high resolution features are connected with the up-sampled output. After that a successive convolution layer can learn to bring together a more definite output based on this information. In

the up-sampling half we've an oversized range of feature channels, which permit the network to propagate context data to higher resolution layers. As a consequence, the expansive path is a lot of or less parallel to the contracting path, and yields a U-shaped architecture. The network doesn't have any fully connected layers and solely uses the valid a part of every convolution.To predict the pixels within the border region of the image, the missing context is figured by mirroring the input image. This covering strategy is vital to use the network to large pictures, since otherwise the resolution would be restricted by the system memory[5].

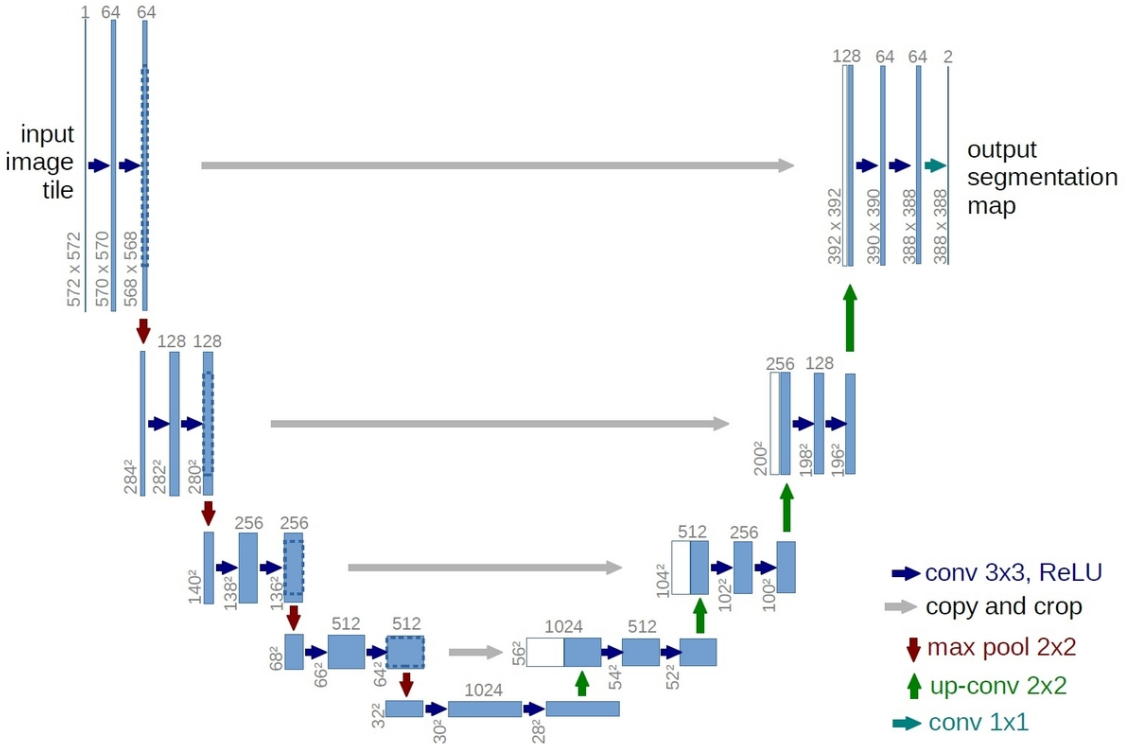

Figure 3.1: U-Net architecture[5]

3.1.2 Network Architecture

The network architecture is illustrated in this case with 2D in figure 3.1. It consists of a contracting path (left side) and an expansive path (right side). The contracting path follows the typical architecture of a convolutional network. It consists of the repeated application of two 3x3 convolutions (unpadded convolutions), each followed by a rectified linear unit (ReLU)[6] and a 2x2 max pooling operation with stride 2 for down-sampling. At each down-sampling step we double the number of feature channels.Every step in the expansive path consists of an up-sampling of the feature map followed by a 2x2 convolution that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU. The cropping is necessary due to the loss of border pixels in every convolution. At the final layer a 1x1 convolution is used to map each 64- component feature vector to the desired number of classes. In total the network has 23 convolutional layers.

3.1.3 Data Augmentation

When available number of training images are less, it is required to teach the network the robustness and invariance properties. It is called as data augmentation. For microscopical images we mainly need rotation and shift invariance along with robustness to deformations and gray value variations. If a segmentation network has very few annotated images in that case to train the network random elastic deformations of the training samples appear to be the best approach.

The U-Net architecture achieves very good performance on very different biomedical segmentation applications.

3.2 Xception

3.2.1 Introduction

Xception stands for Extreme version of Inception. With a modified depthwise separable convolution, it is even better than Inception-v3 for both ImageNet ILSVRC and JFT datasets.

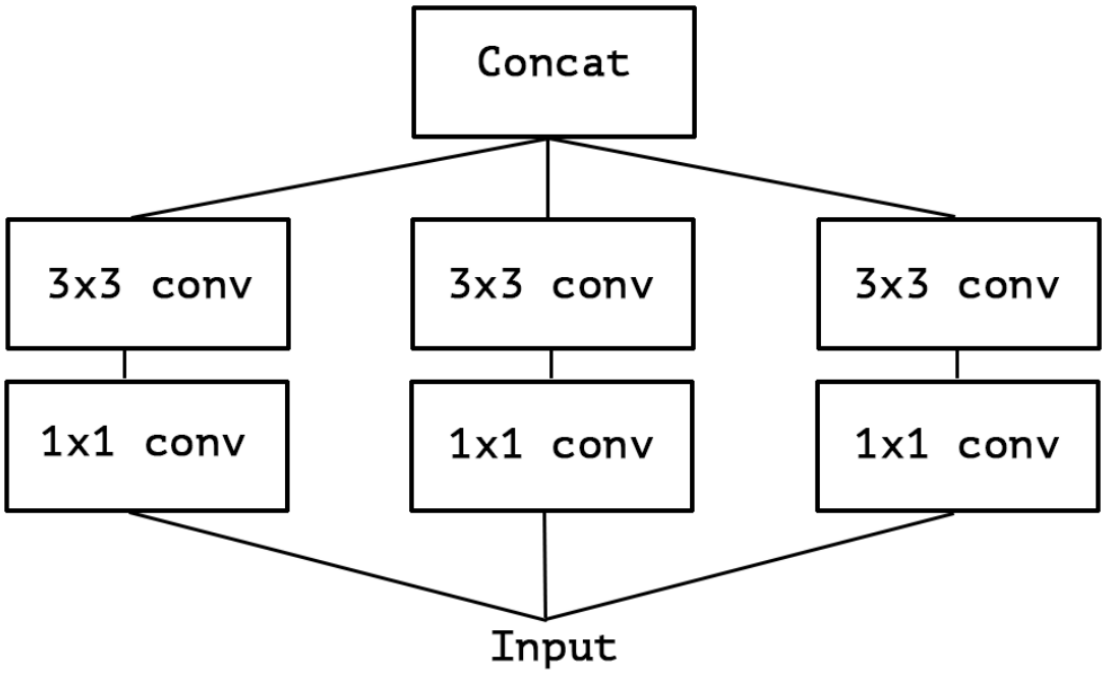

Figure 3.2: Depthwise Separable Convolution[7]

Depthwise convolution is the channel-wise n x n spatial convolution. Suppose in the figure above, we have 3 channels, then we will have 3 n x n spatial convolution. Pointwise convolution actually is the 1x1 convolution to change the dimension. Compared with conventional convolution, we do not need to perform convolution across all channels. That means the number of connections are fewer and the model is lighter[7].

3.2.2 Modified Depthwise Separable Convolution in Xception

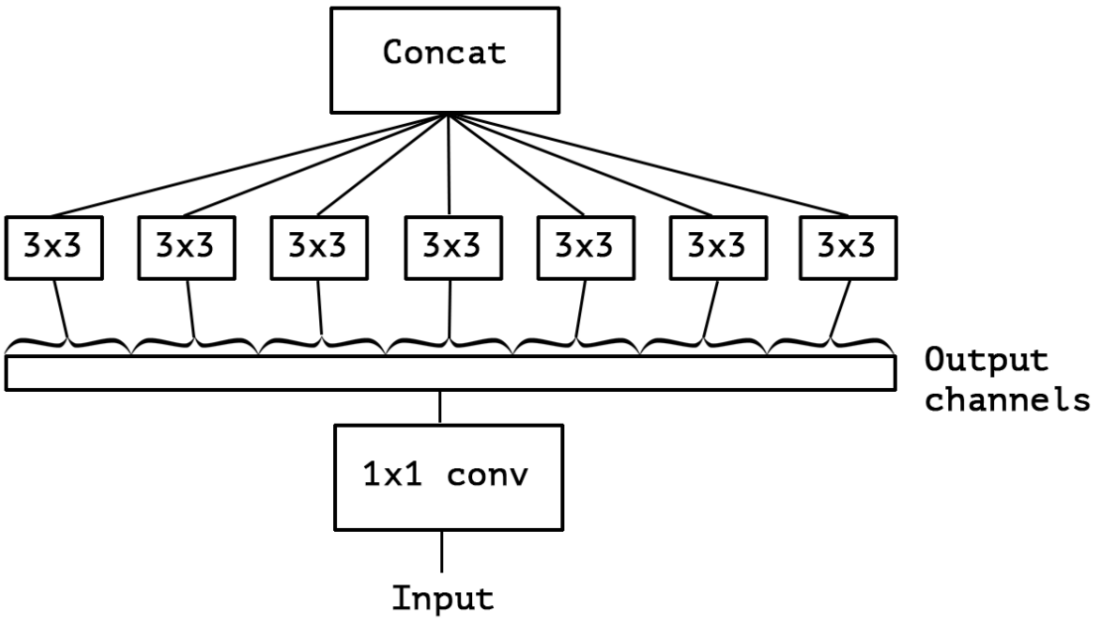

Figure 3.3: Modified Depthwise Separable Convolution in Xception[7]

The modified depthwise separable convolution is the pointwise convolution followed by a depthwise convolution. This modification is motivated by the inception module in Inception-v3 that 1x1 convolution is done first before any nxn spatial convolutions. Thus, it is a bit different from the original one. The original depthwise separable convolutions as usually implemented perform first channel-wise spatial convolution. After that it perform 1x1 convolution whereas the modified depthwise separable convolution perform 1x1 convolution first then channel-wise spatial convolution.

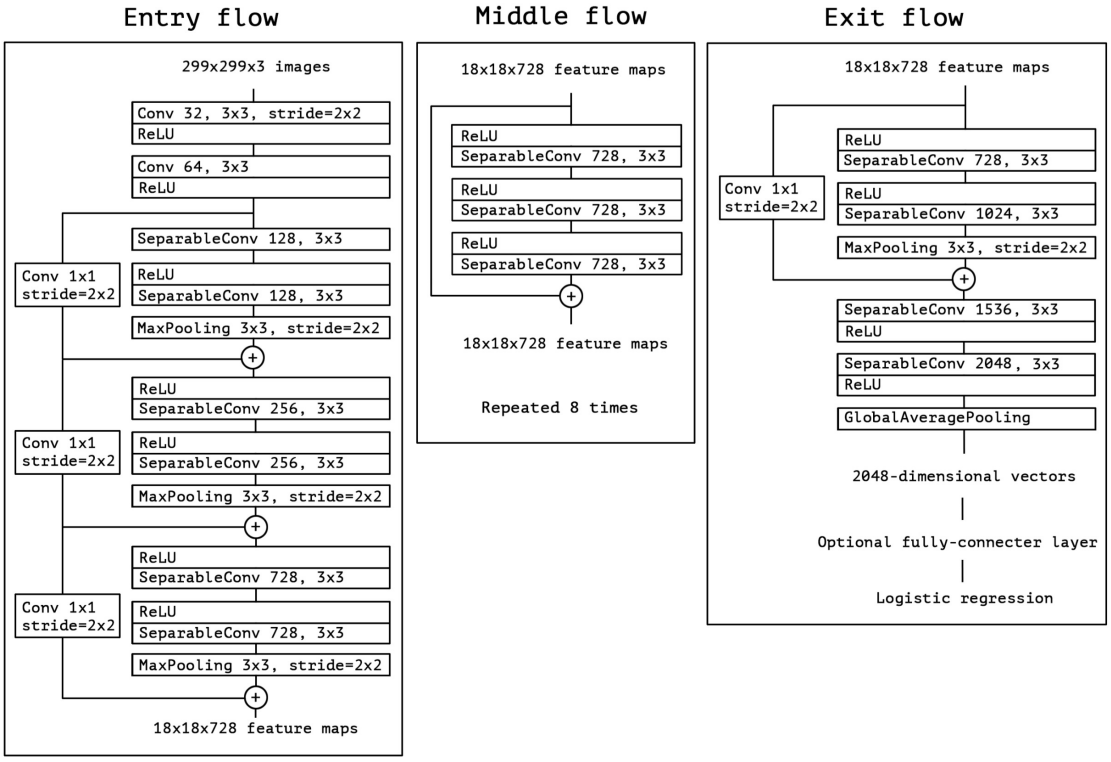

Figure 3.4: The Xception architecture[7]

As in the figure above, SeparableConv is the modified depthwise separable convolution. SeparableConvs are treated as Inception Modules and placed throughout the whole deep learning architecture. The architecture illustrated in figure 3.4 and discussed in thesis is 2D.

3.3 DenseNet

Improvement in computer hardware and network structure are made in last 20 years, which have enabled the training of truly deep CNNs only recently. The problems arise with CNNs when they go deeper. This is because the path for information from the input layer until the output layer becomes so big, that they can get vanished before reaching the other side. DenseNets simplify the connectivity pattern between layers introduced in other architectures. Another problem with

very deep networks was the problems to train. DenseNets solve this issue since each layer has direct access to the gradients from the loss function and the original input image. DenseNets do not sum the output feature maps of the layer with the incoming feature maps but concatenate them[8].

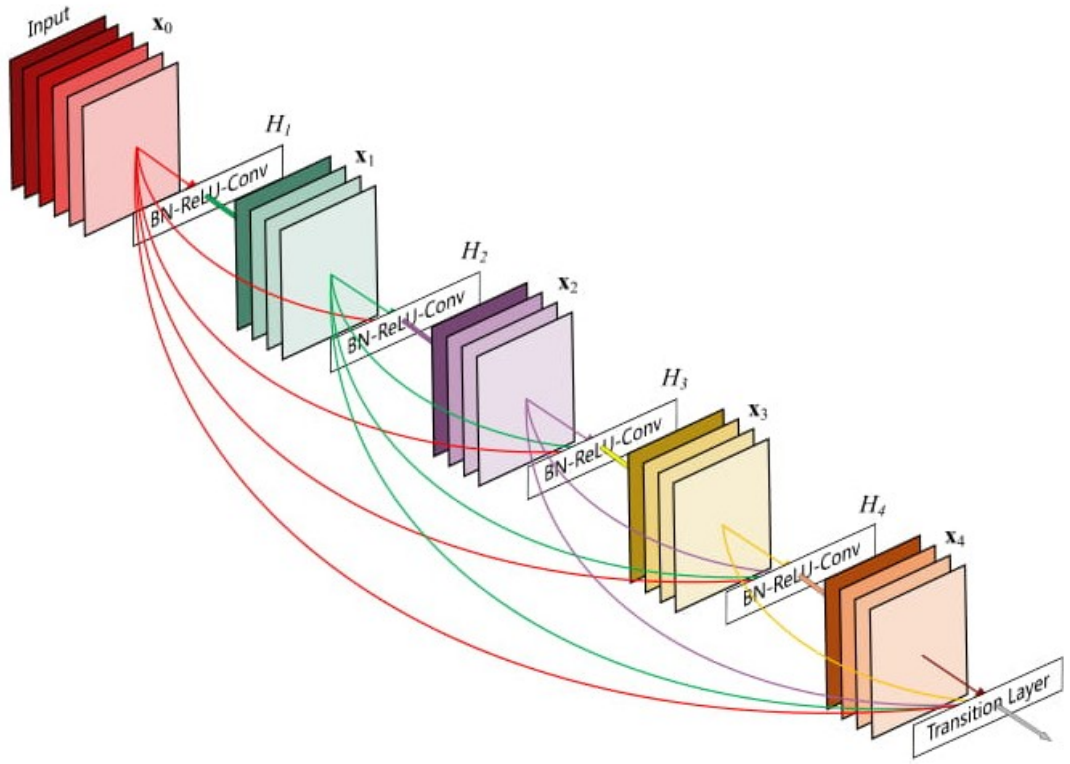

Figure 3.5: A 5-layer dense block with a growth rate of k = 4. Each layer takes all preceding feature-maps as input.[8]

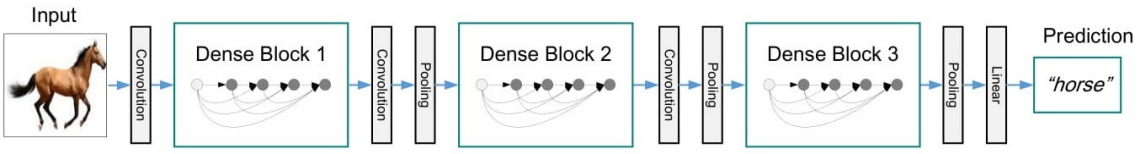

Figure 3.6: A deep DenseNet with three dense blocks.[8]

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48	[1 × 1 conv, 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Figure 3.7: DenseNet architectures for ImageNet. The growth rate for all the networks is k = 32. Note that each conv layer shown in the table corresponds the sequence BN-ReLU-Conv.

Dense Convolutional Network (DenseNet) discussed in this thesis is 2D, which connects each layer to every other layer in a feed-forward fashion. On the other hand,traditional convolutional networks with L layers have L connectionsone between each layer and its subsequent layerour network has L*(L+1)/ 2 direct connections.Biggest advantage of DenseNets is their improved flow of information and gradients throughout the network, which makes them easy to train alongside with better parameter efficiency. This helps in training of deeper network architectures.

Chapter 4

Tools/Frameworks and Results

This chapter describes various tools used during this project and results of the project.

4.1 Tools/Frameworks Used

- Python 2.7 and Python 3
- OpenVINO Toolkit
- OPCM
- Intel VTune Amplifier
- Intel Internal Tools
- Keras 2.2.2
- Tensorflow 1.12.0

4.1.1 Python 2.7 and Python 3

Python 2.7 is used to pre-process dataset, build the model and infer them. Python 3 is used to optimize and infer the model using OpenVINO Toolkit.

4.1.2 OpenVINO Toolkit

The Intel Distribution of OpenVINO toolkit is a comprehensive toolkit for quickly developing applications and solutions that emulate human vision. Based on CNN, the toolkit extends Computer Vision (CV) workloads across Intel hardware(including accelerators) and maximizes performance[9].

The Intel Distribution of OpenVINO toolkit:

- Enables CNN based deep learning inference on the edge.
- Supports heterogeneous execution across Intel's CV accelerators, using a common Application Programming Interface (API) for the CPU, Intel Integrated Graphics, Intel Movidius Neural Compute Stick, Intel Neural Compute Stick 2, and Intel Field Programmable Gate Array (FPGA).
- Speeds time-to-market through an easy-to-use library of CV functions and pre-optimized kernels.
- Includes optimized calls for CV standards, including OpenCV, OpenCL, and OpenVX.

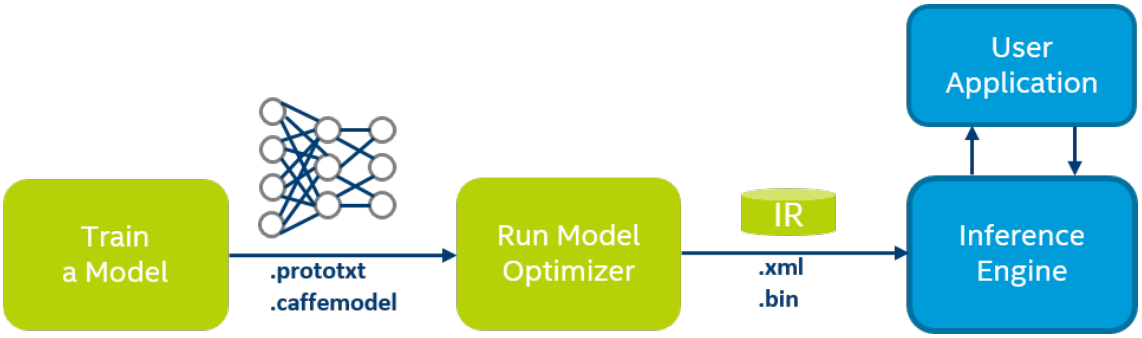

Figure 4.1: Deep Learning Inference Engine Workflow[9]

OpenVINO Model Optimizer is a cross-platform command-line tool that facilitates the transition between the training and deployment environment, performs

static model analysis, and adjusts deep learning models for optimal execution on end-point target devices[10].

Model Optimizer process assumes we have a network model trained using one of the supported frameworks. The Figure 4.1 illustrates the typical workflow for deploying a trained deep learning model:

A summary of the steps for optimizing and deploying a trained model:

- Configure the Model Optimizer for your framework.
- Convert a trained model to produce an optimized Intermediate Representation (IR) of the model based on the trained network topology, weights, and bias values.
- Test the model in the Intermediate Representation format using the Inference Engine in the target environment via provided Inference Engine validation application or sample applications.
- Integrate the Inference Engine into your application to deploy the model in the target environment.

Model Optimizer loads a model into memory, reads it, builds the internal representation of the model, optimizes it, and produces the Intermediate Representation. Intermediate Representation is the only format the Inference Engine accepts. Model Optimizer does not infer models. Model Optimizer is an offline tool that runs before the inference takes place. Model Optimizer produce a valid and an optimized Intermediate Representation.

Pretrained models contain layers that are important for training, such as the Dropout layer. These layers are useless during inference and might increase the inference time. In many cases, these layers can be automatically removed from the resulting Intermediate Representation. **However, if a group of layers can be represented as one mathematical operation, and thus as a single layer, the Model Optimizer recognizes such patterns and replaces these layers**

with one. The result is an Intermediate Representation that has fewer layers than the original model. This decreases the inference time.

4.1.3 OPCM

Open Performance Counter Monitor (PCM) is an API and a set of tools based on the API to monitor performance and energy metrics of Intel Core, Xeon, Atom and Xeon Phi processors. PCM works on Linux, Windows, Mac OS X, FreeBSD and DragonFlyBSD operating systems[11].

CPU utilization metric is calculated from the operating system. Metric is obtained from the OS based utility like UNIX top, UNIX htop, and Windows task manager. Metric calculated using this utility gives a good prediction of CPU utilization for architectures of 80s that had a much more uniform and predictable performance when compared to modern architectures which involves multi core, multi node, multithreading, pipelining and multi-level caches.

To overcome this problem Intel processors have already embedded the capability to monitor performance events with the processors. Intel processors consists of performance monitoring units (PMU). Collecting dynamic data from this unit helps in obtaining precise CPU resource utilization. Using this tool, data pertaining to core as well as uncore can be collected.Part of the processor that contains Intel Quick Path Interconnect (QPI) which connect to other processors, integrated memory controller and I/O hub is called uncore. Core data include core frequency including Intel turbo boost, elapsed core clock ticks, cache hits and misses, instruction retired and core residencies.

4.1.4 Intel VTune Amplifier

Intel VTune Amplifier is a performance analysis tool. It is for users developing serial and multithreaded applications. VTune Amplifier helps to analyze the algorithm choices. It also helps to identify where and how application can benefit from

available hardware resources[12].

Use the VTune Amplifier to locate or determine the following:

- The most time-consuming (hot) functions in your application and/or on the whole system
- Sections of code that do not effectively utilize available processor time
- The best sections of code to optimize for sequential performance and for threaded performance
- Synchronization objects that affect the application performance
- Whether, where, and why your application spends time on input/output operations
- The performance impact of different synchronization methods, different numbers of threads, or different algorithms
- Thread activity and transitions
- Hardware-related issues in your code such as data sharing, cache misses, branch misprediction, and others

4.1.5 Intel Internal Tools

By using Intel Internal Tools, Top-down Microarchitecture Analysis Method[13] metric is calculated.

4.2 Results

4.2.1 U-Net using Up Sampling(U-Net-US) Results

The comparison of U-Net-US baseline and optimized Studies Per Seconds (SPS) for N^{th} Generation of Intel Server is represented in figure 4.2. It is concluded that optimized prediction numbers are 5x better than baseline because of loop collapsing.

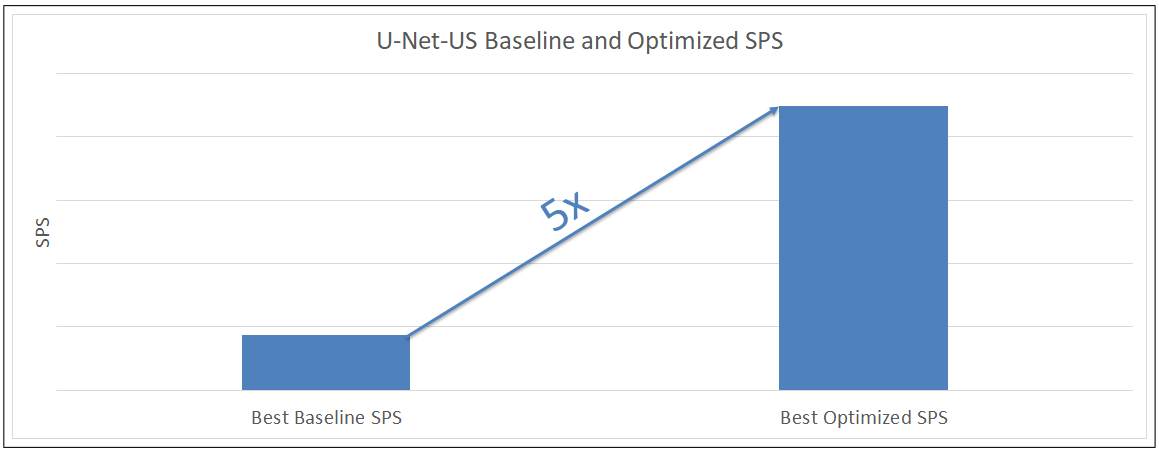

Figure 4.2: U-Net-US Baseline SPS

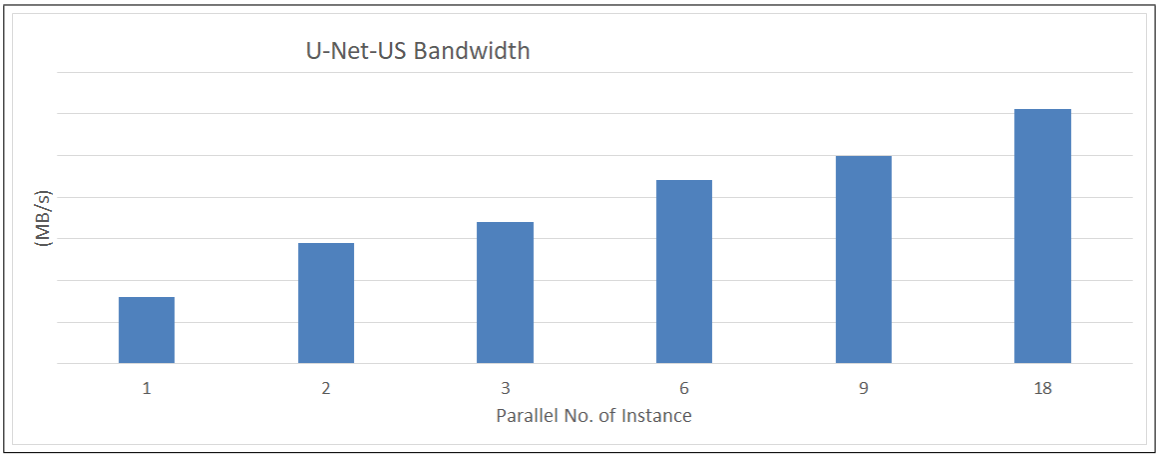

Figure 4.3: U-Net-US Bandwidth

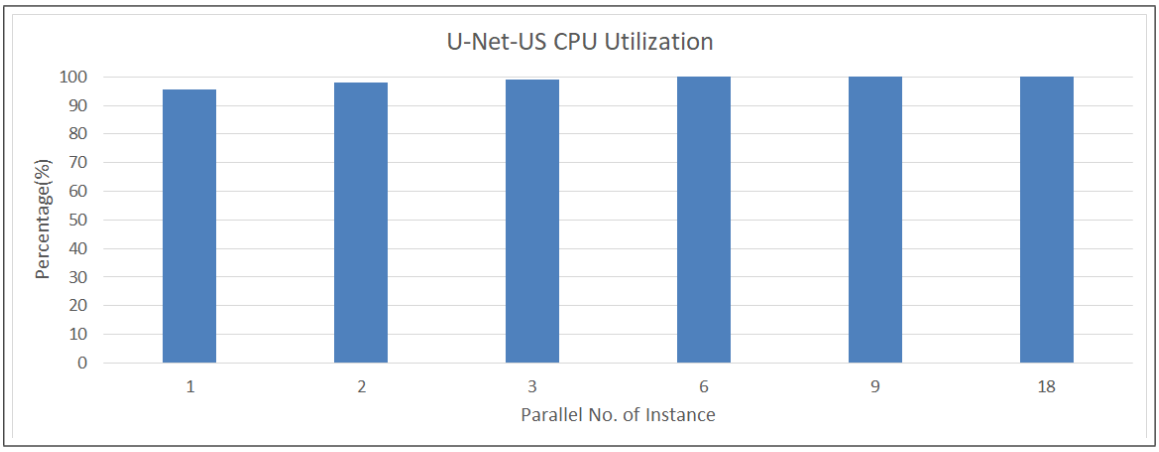

Figure 4.4: U-Net-US CPU Utilization

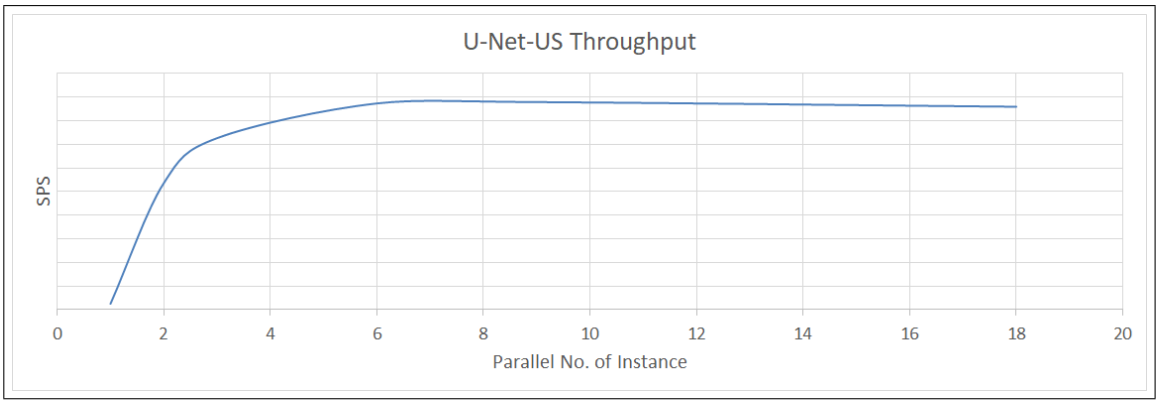

Figure 4.5: U-Net-US Throughput

U-Net-US bandwidth, CPU utilization and throughput data is represented in figure 4.3, 4.4 and 4.5 respectively. If there is no memory constrain then we can go for number of instances which yields peak performance, if not go with the number of instance which yields significant performance with optimal bandwidth. Also we can go for number of instances till acceptable latency. In this case at 6 number of instance per machine gives good CPU throughput at acceptable latency and optimal bandwidth. [1]

[1] Y-Axis in throughput chart is enlarged so that peak can easily be identified.

4.2.2 U-Net using Transposed Convolution(U-Net-TC) Results

The comparison of U-Net-TC baseline and optimized prediction numbers for N^{th} Generation of Intel Server is represented in figure 4.6. It is concluded that optimized prediction numbers are 4.5x better than baseline and 1.3x better than U-Net-US optimized numbers because of better hardware optimization using transposed convolution.

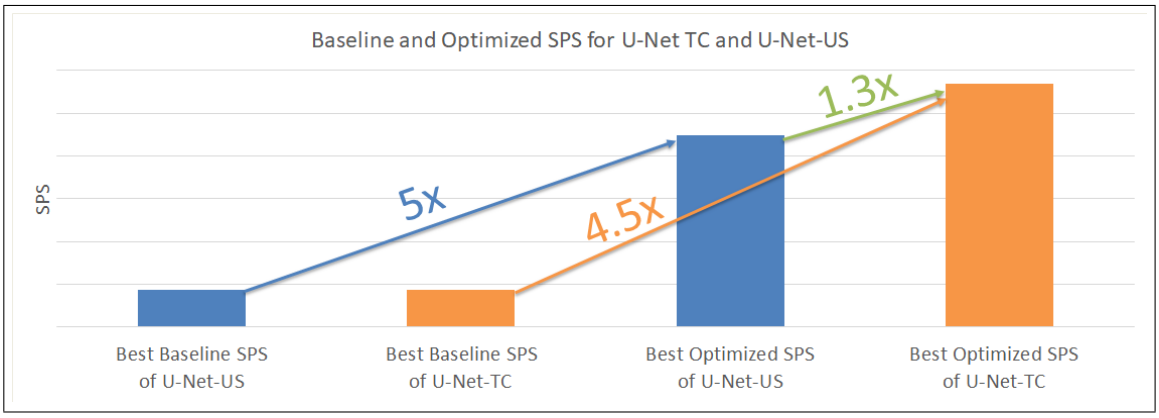

Figure 4.6: U-Net-TC Baseline SPS

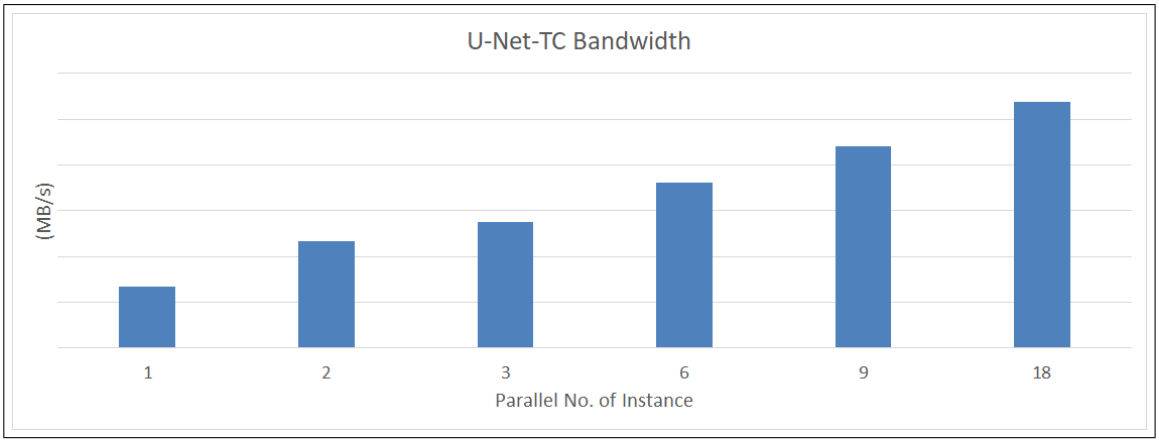

Figure 4.7: U-Net-TC Bandwidth

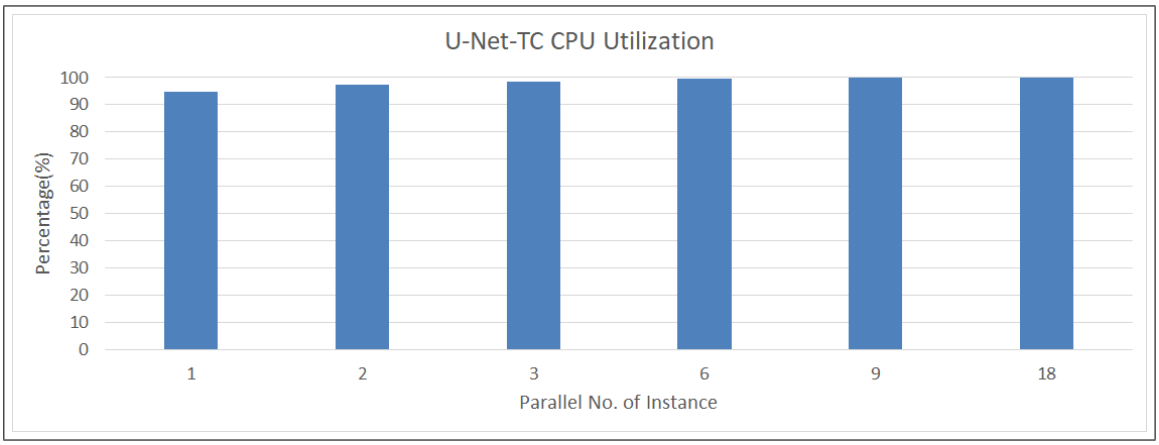

Figure 4.8: U-Net-TC CPU Utilization

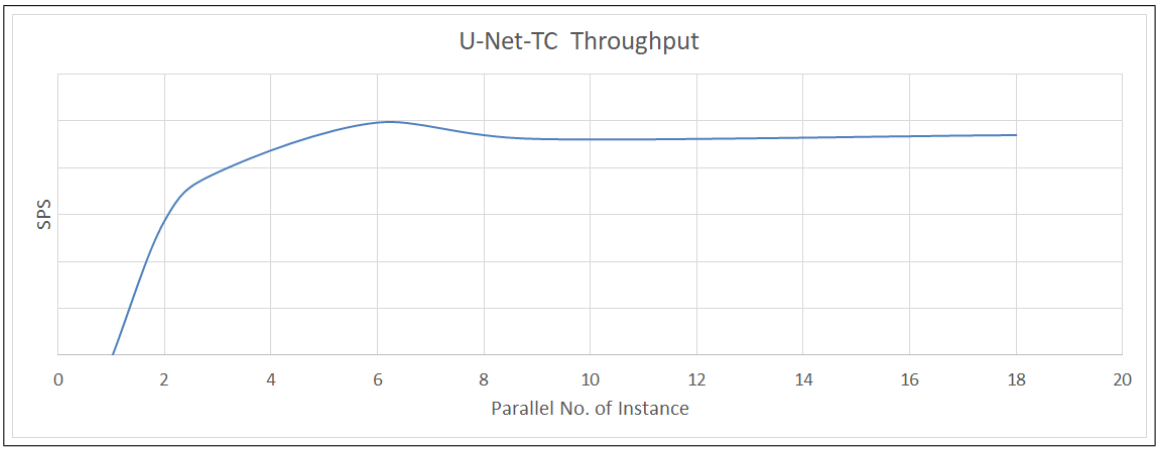

Figure 4.9: U-Net-TC Throughput

U-Net-TC bandwidth, CPU utilization and throughput data is represented in figure 4.7, 4.8 and 4.9 respectively. If there is no memory constrain then we can go for number of instances which yields peak performance, if not go with the number of instance which yields significant performance with optimal bandwidth. Also we can go for number of instances till acceptable latency. In this case at 6 number of instance per machine gives good CPU throughput at acceptable latency and optimal bandwidth. [2]

[2]Y-Axis in throughput chart is enlarged so that peak can easily be identified.

4.2.3 3D U-Net Results

The comparison of 3D U-Net baseline and optimized prediction time for N^{th} Generation of Intel Server is represented in figure 4.10. It is concluded that optimized prediction numbers are 4.5x better than baseline.

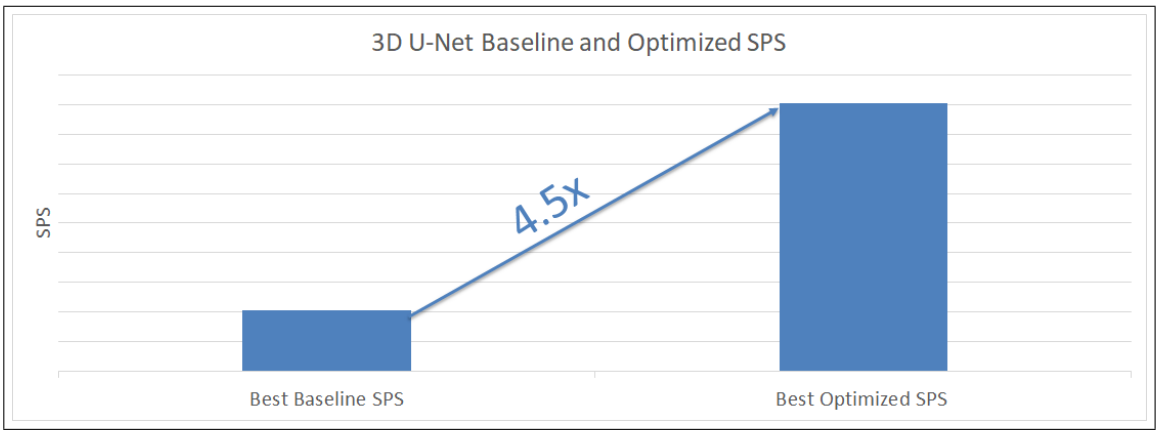

Figure 4.10: 3D U-Net Baseline Prediction Time for N^{th} Generation Intel Xeon Server

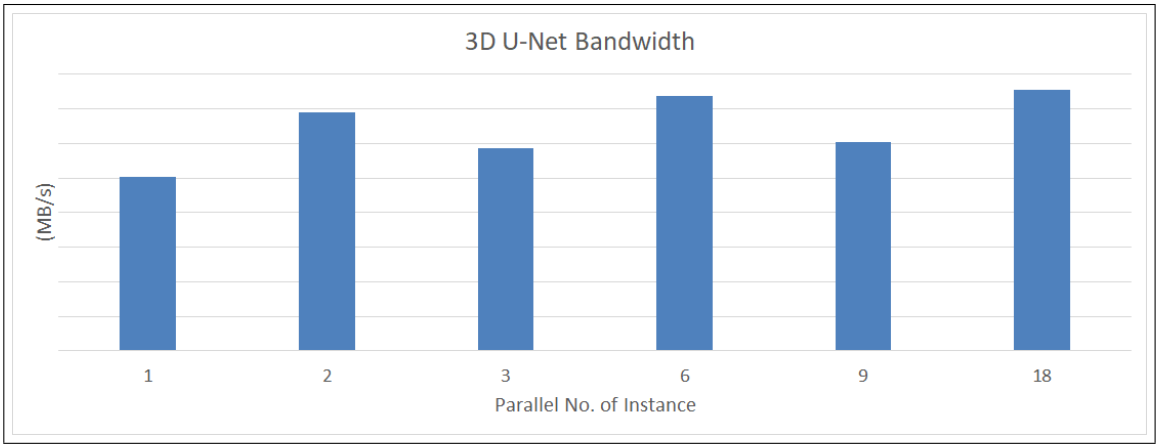

Figure 4.11: 3D U-Net Bandwidth

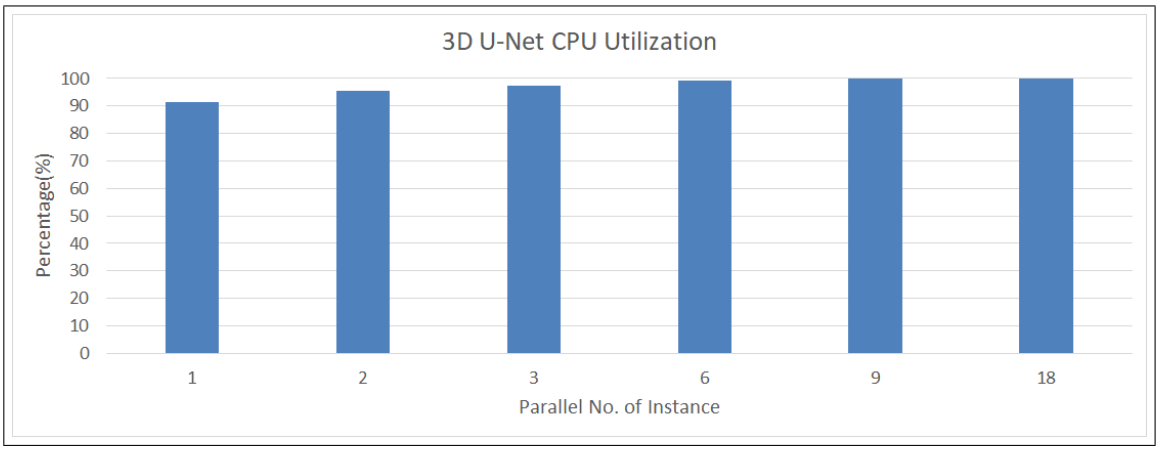

Figure 4.12: 3D U-Net CPU Utilization

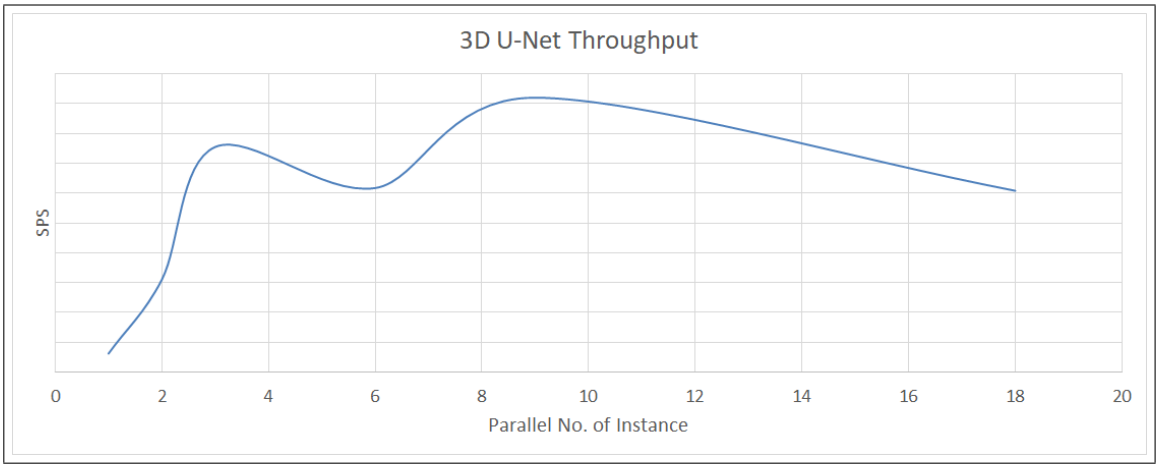

Figure 4.13: 3D U-Net Throughput

3D U-Net bandwidth, CPU utilization and throughput data is represented in figure 4.11, 4.12 and 4.13 respectively. If there is no memory constrain then we can go for number of instances which yields peak performance, if not go with the number of instance which yields significant performance with optimal bandwidth. Also we can go for number of instances till acceptable latency. In this case peak performance is only 3% increase compared to single instance performance. So it is better to go with single instance for 3D U-Net because core scaling is not giving good performance improvement. [3]

[3]Y-Axis in throughput chart is enlarged so that peak can easily be identified.

4.2.4 Xception Results

The comparison of Xception baseline and optimized SPS for N^{th} Generation of Intel Server is represented in figure 4.14. It is concluded that optimized prediction numbers are 6x better than baseline.

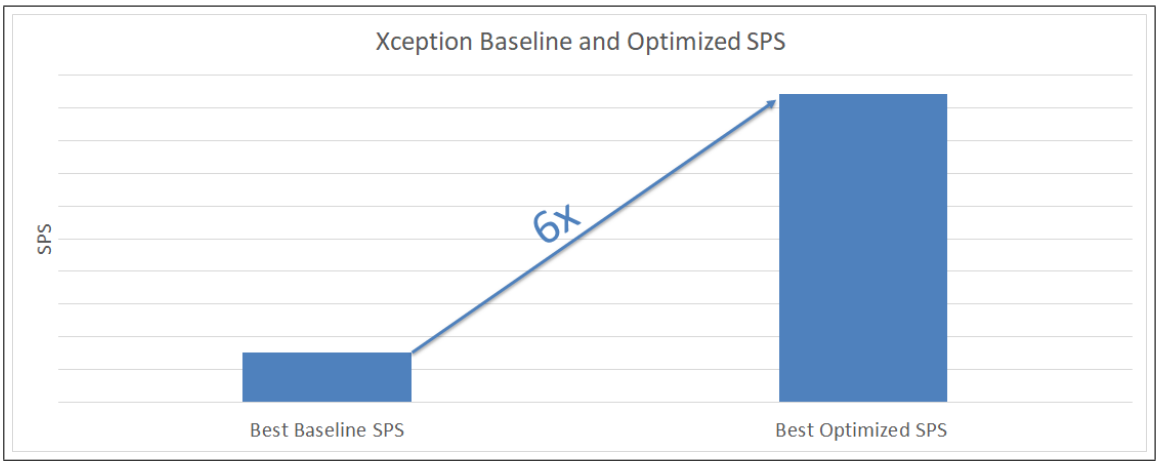

Figure 4.14: Xception Baseline SPS

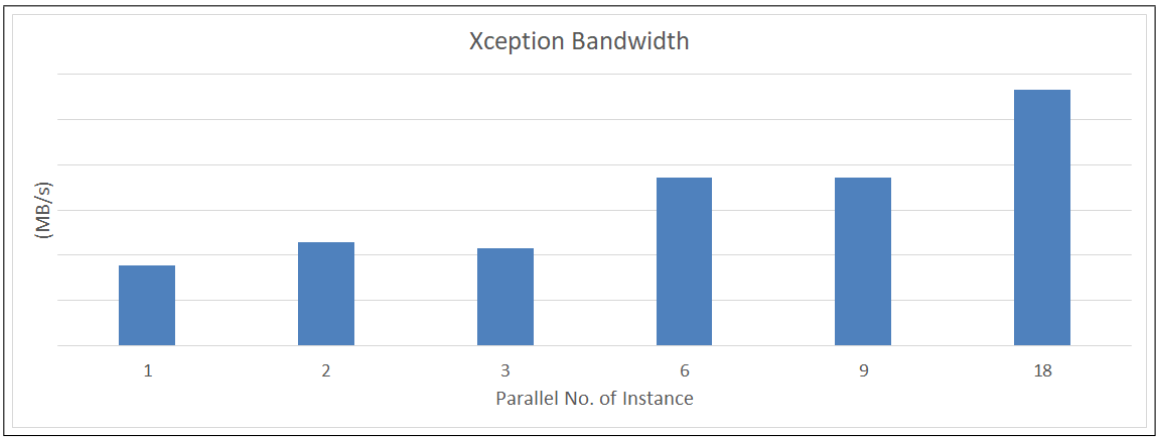

Figure 4.15: Xception Bandwidth

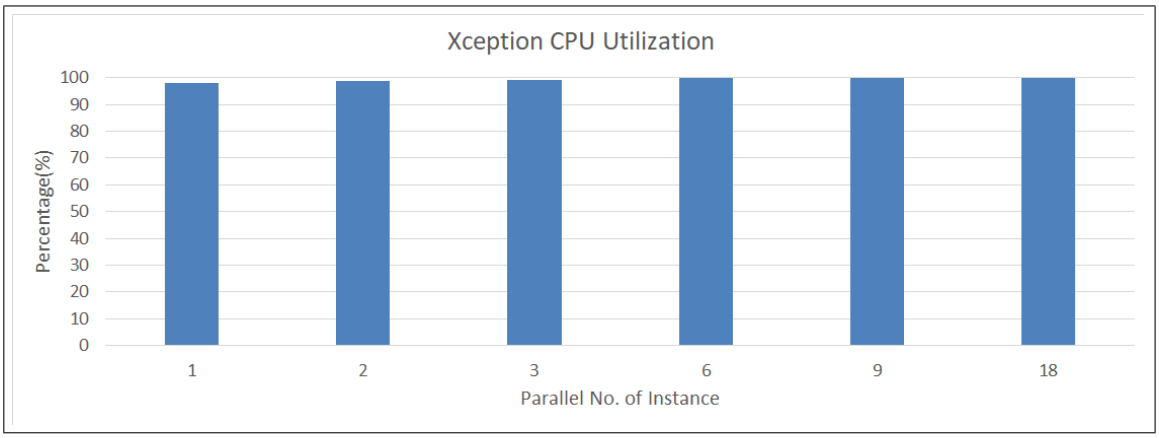

Figure 4.16: Xception CPU Utilization

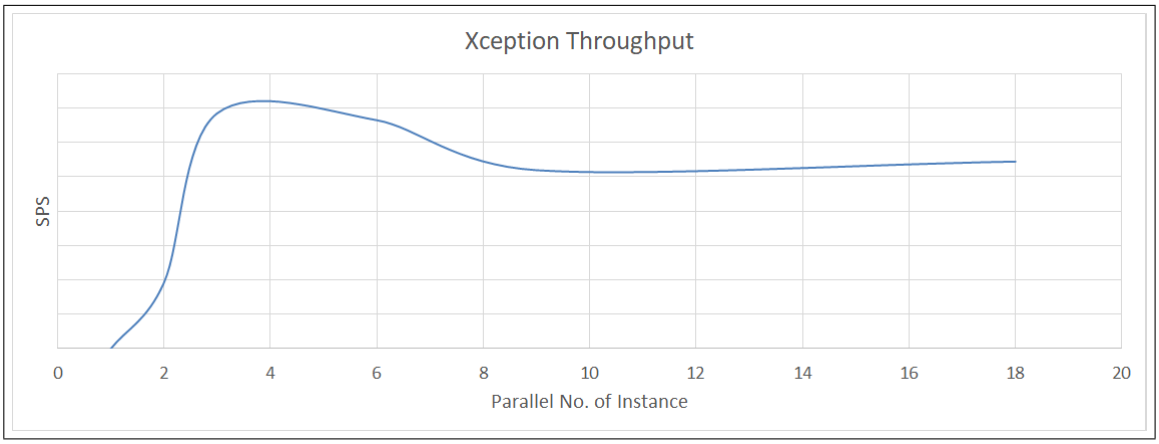

Figure 4.17: Xception Throughput

Xception bandwidth, CPU utilization and throughput data is represented in figure 4.15, 4.16 and 4.17 respectively. If there is no memory constrain then we can go for number of instances which yields peak performance, if not go with the number of instance which yields significant performance with optimal bandwidth. Also we can go for number of instances till acceptable latency. In this case at 3 number of instance per machine gives good CPU throughput at acceptable latency and optimal bandwidth. [4]

[4]Y-Axis in throughput chart is enlarged so that peak can easily be identified.

4.2.5 DenseNet

The comparison of DenseNet baseline and optimized studies per seconds for N^{th} Generation of Intel Server is represented in figure 4.18. It is concluded that optimized prediction numbers are 11x better than baseline because of loop collapsing.

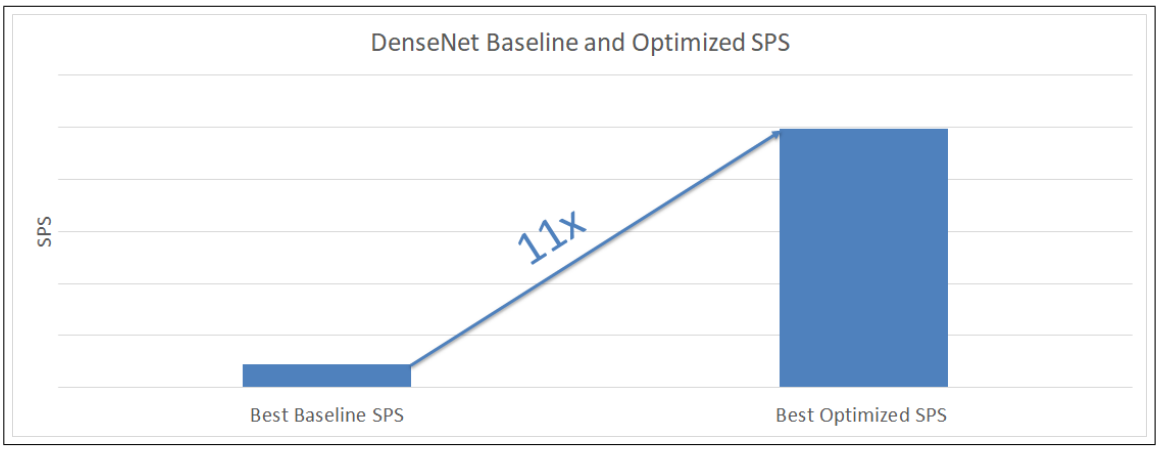

Figure 4.18: DenseNet Baseline Prediction Time for N^{th} Generation Intel Xeon Server

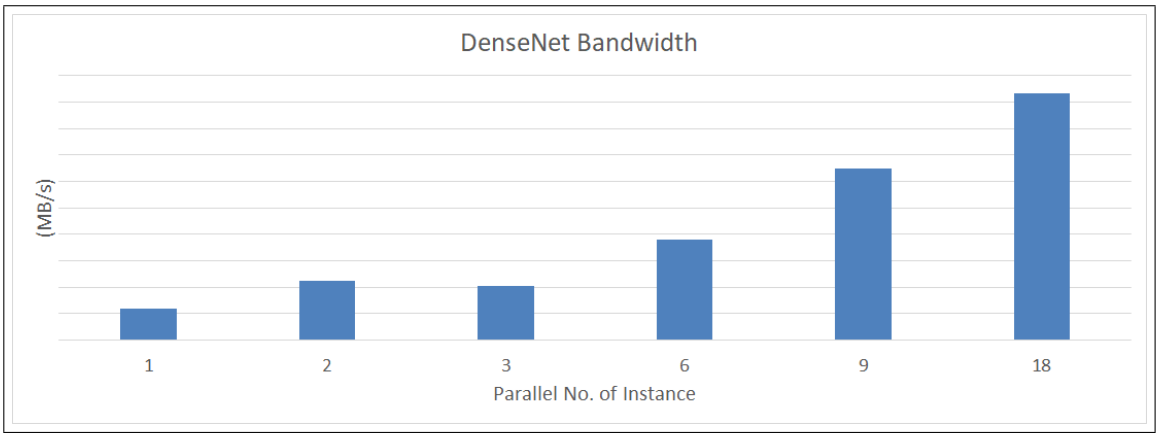

Figure 4.19: DenseNet Bandwidth

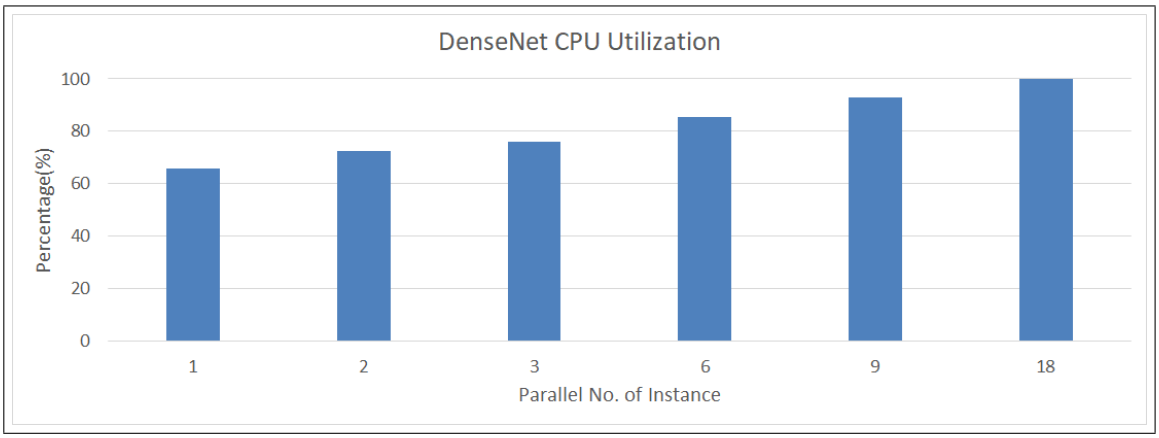

Figure 4.20: DenseNet CPU Utilization

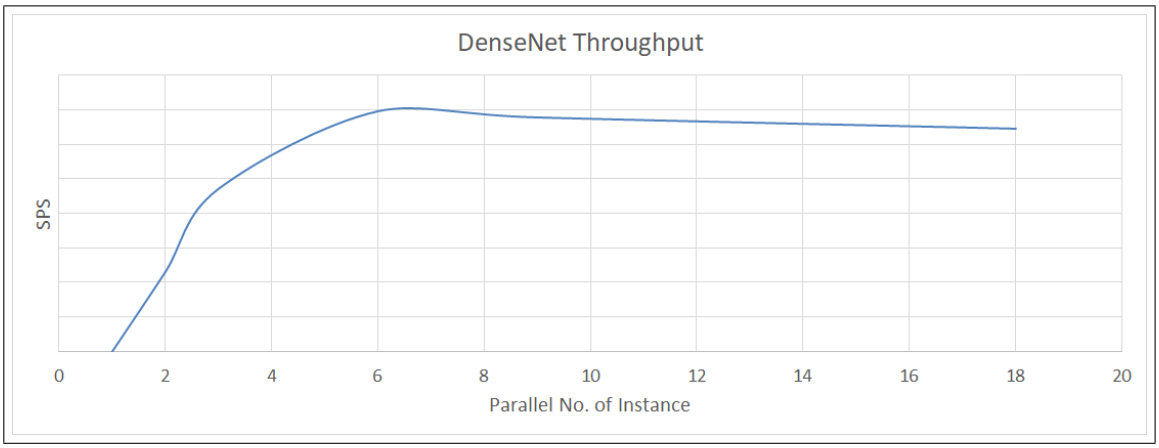

Figure 4.21: DenseNet Throughput

DenseNet bandwidth, CPU utilization and throughput data is represented in figure 4.19, 4.20 and 4.21 respectively. If there is no memory constrain then we can go for number of instances which yields peak performance, if not go with the number of instance which yields significant performance with optimal bandwidth. Also we can go for number of instances till acceptable latency. In this case at 6 number of instance per machine gives good CPU throughput at acceptable latency and optimal bandwidth. [5]

[5]Y-Axis in throughput chart is enlarged so that peak can easily be identified.

4.2.6 CPU vs GPU Performance

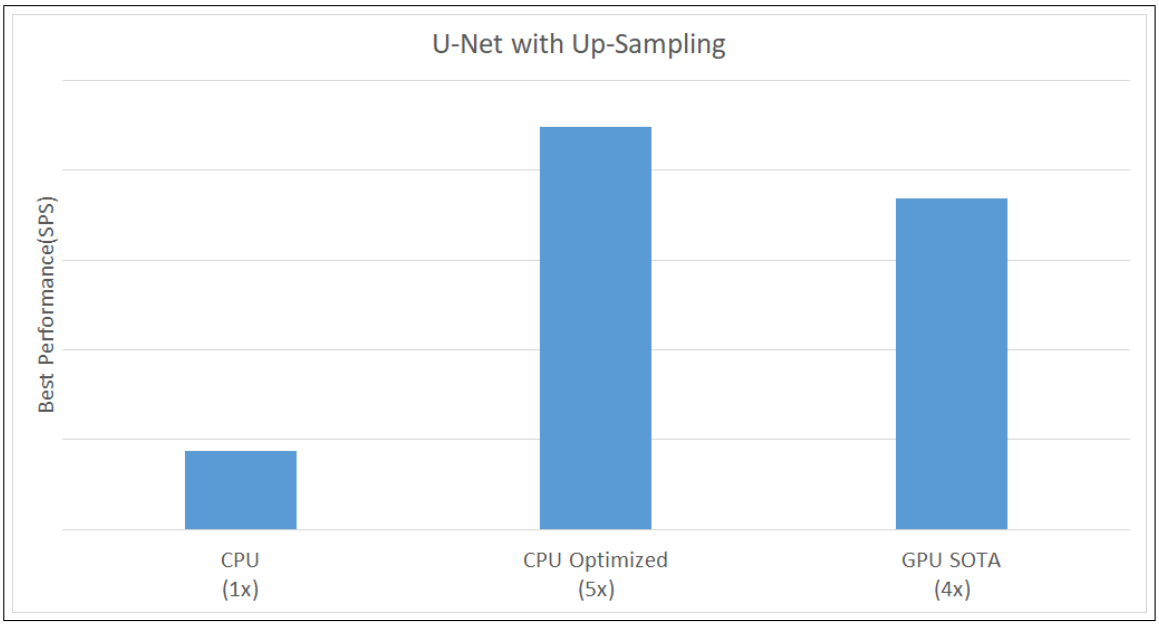

Figure 4.22: U-Net with Up-Sampling CPU, CPU Optimized and GPU Performance

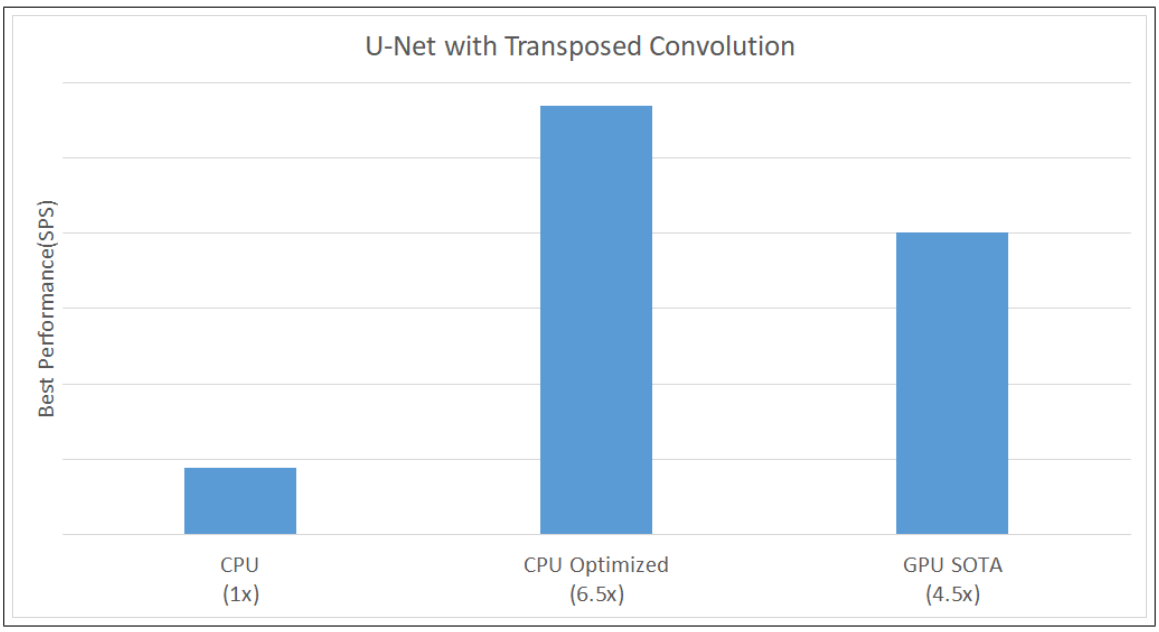

Figure 4.23: U-Net with Transposed Convolution CPU, CPU Optimized and GPU Performance

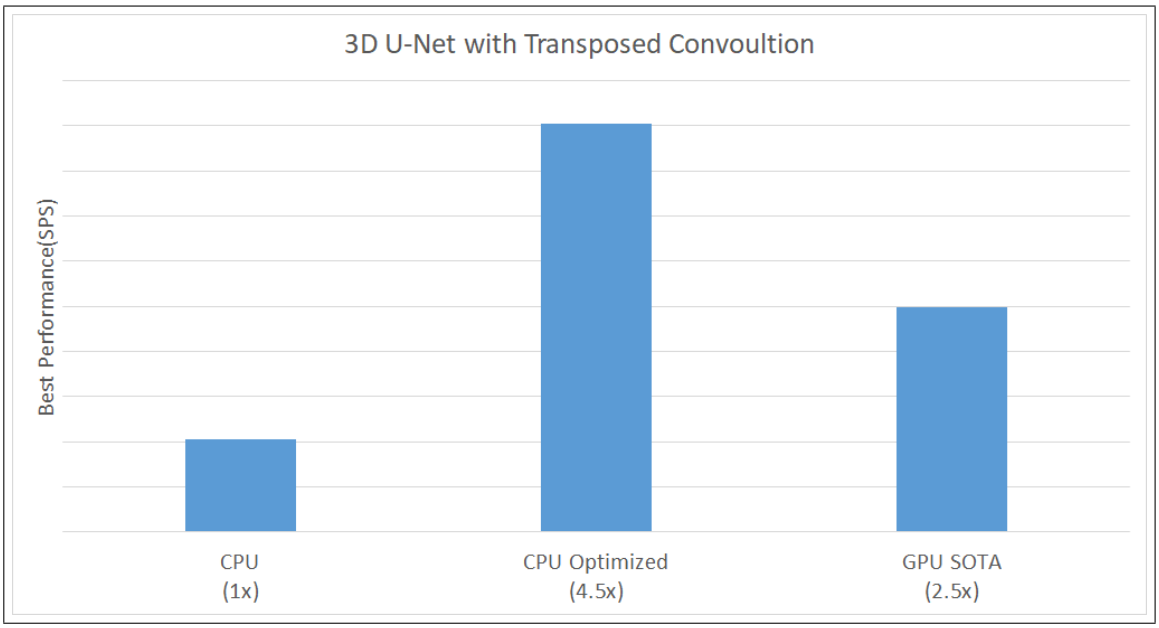

Figure 4.24: 3D U-Net with Transposed Convolution CPU, CPU Optimized and GPU Performance

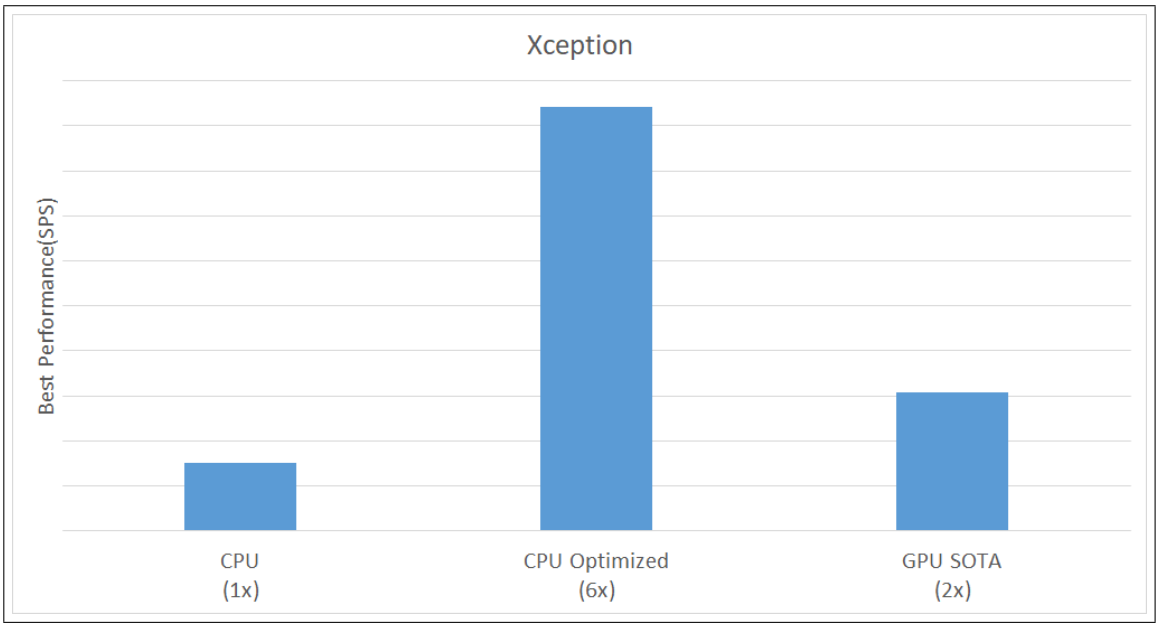

Figure 4.25: Xception CPU, CPU Optimized and GPU Performance

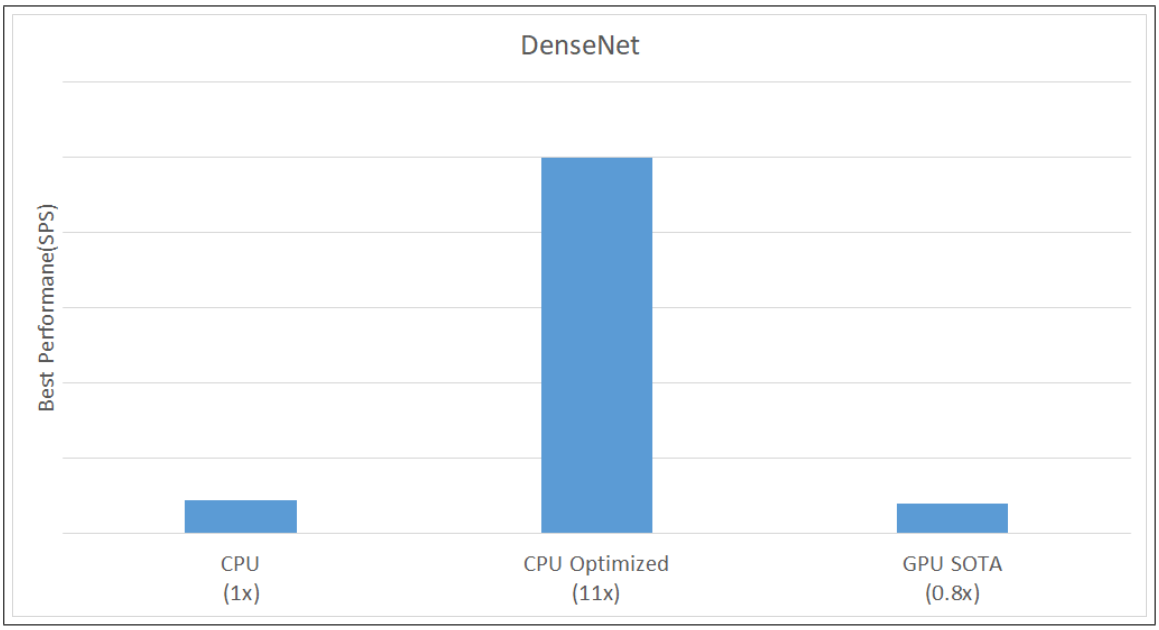

Figure 4.26: DenseNet CPU, CPU Optimized and GPU Performance

CPU, CPU optimized and GPU performance results are compared and shown in the above figures for all the topologies. Here, CPU and GPU performance numbers are with comparable peak performance TFLOPS (FP32) parts/devices. From the comparison, CPU optimized performance numbers showed improvement than CPU baseline and GPU SOTA performance numbers for given topologies.

Chapter 5

Conclusion and Future Scope

In this work, all workload proxies are successfully implemented and optimized using OpenVINO model optimizer.

5.1 Conclusion

In this thesis, performance analysis of Intel platforms by characterizing and optimizing workload proxies related to healthcare is carried out.

The experimental analysis related to U-Net-US, U-Net-TC, 3D U-Net, Xception and DenseNet showed improvement in performance numbers from baseline to optimized as below:

- U-Net-US - 5x
- U-Net-TC - 6.5x
- 3D U-Net - 4.5x
- Xception - 6x
- DenseNet - 11x

Performance analysis of CPU and GPU showed that CPU optimized performance is better than GPU SOTA for the given topologies.

By including the experimental results in the analysis, scalability study is carried out for next generation platform. The analysis also gave prediction on the maximum number of frames per second supported by the existing platform. These workload proxies are being used by many other teams inside the Intel.

5.2 Future Scope

In recent past years, deep learning has obtained a central position toward the automation of our daily life and provided considerable improvements as compared to traditional machine learning algorithms. Based on the enormous performance, most researchers believe that within next 15 years, deep learning based applications will take over human and most of the daily routine activities will be performed by autonomous machineries. Majority of deep learning methods focus on supervised deep learning. Although annotations of medical data especially image data is not always possible i.e. in case when rare disease or unavailability of qualified expert. To overcome, the issue of big data unavailability, the supervised deep learning field is required to shift to unsupervised or semi supervised. Acceptance of deep learning in health sector need clinical validation by medical expert. So far deep learning based application provided positive feedback. Because of the sensitivity of healthcare data and challenges, we should look more sophisticated deep learning methods that can deal complex healthcare data efficiently. Lastly we conclude that there are unlimited opportunities to improve healthcare system via Deep Learning methods.

References

[1] R. Jain, "The Art of Computer System performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling"John Wiley and Sons, Inc, 1991.

[2] Muhammad Imran Razzak, Saeeda Naz and Ahmad Zaib, "Deep Learning for Medical Image Processing: Overview, Challenges and Future,"

[3] Introduction to Intel Advanced Vector Extensions https://software.intel.com/en-us/articles/introduction-to-intel-advanced-vector-extensions

[4] Intel Math Kernel Library for Deep Learning Networks https://software.intel.com/en-us/articles/intel-mkl-dnn-part-1-library-overview-and-installation

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation,"18 May 2015

[6] Xavier Glorot, Antoine Bordes and Yoshua Bengio, "Deep Sparse Rectifier Neural Networks "

[7] Francois Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions,"4 Apr 2017

[8] Gao Huang, Zhuang Liu, Laurens van der Maaten"Densely Connected Convolutional Networks ,"28 Jan 2018

[9] Model Optimizer Developer Guide, Retrieved from https://software.intel.com/en-us/articles/OpenVINO-ModelOptimizer

[10] Release Notes for Intel Distribution of OpenVINO toolkit Guide, Retrieved from https://software.intel.com/en-us/openvino-toolkit

[11] Intel Performance Counter Monitor Guide, Retrieved from https://software.intel.com/en-us/articles/intel-performance-counter-monitor

[12] Intel VTune Amplifier 2019 User Guide, Retrieved from https://software.intel.com/en-us/vtune-amplifier-help

[13] Ahmad Yasin , "A Top-Down method for performance analysis and counters architecture ,"2014