A DEEP LEARNING APPROACH FOR COW UDDER DETECTION

ARIF ZAMAN
01-242172-008

A thesis submitted in fulfilment of the
requirements for the award of the degree of
MS (Computer Engineering)

DEPARTMENT OF COMPUTER ENGINEERING

BAHRIA UNIVERSITY ISLAMABAD

FEBRUARY 2020

THESIS COMPLETION CERTIFICATE

Student's Name: ARIF ZAMAN Registration No. 53357 Programme of Study: MS-CE

Thesis Title: A DEEP LEARNING APPROACH FOR COW UDDER DETECTION

It is to certify that the above student's thesis has been completed to my satisfaction and, to my belief, its standard is appropriate for submission for Evaluation. I have also conducted plagiarism test of this thesis using HEC prescribed software and found similarity index at 9% that is within the permissible limit set by the HEC for the MS/MPhil degree thesis. I have also found the thesis in a format recognized by the BU for the MS/MPhil thesis.

Principal Supervisor's Signature: _____

Date:_____ **Name: Dr. Khalid Javed**

APPROVAL FOR EXAMINATION

Scholar's Name: ARIF ZAMAN Registration No. 53357 Programme of Study: MS (CE) ThesisTitle: A DEEP LEARNING APPROACH FOR COW UDDER DETECTION.

It is to certify that the above scholar's thesis has been completed to my satisfaction and, to my belief, its standard is appropriate for submission for examination. I have also conducted plagiarism test of this thesis using HEC prescribed software and found similarity index 9% that is within the permissible limit set by the HEC for the MS degree thesis. I have also found the thesis in a format recognized by the BU for the MS thesis.

Principal Supervisor's Signature _____

Date: _____

Name: _____

DECLARATION

I, ARIF ZAMAN hereby state that my MS thesis titled A DEEP LEARNING APPROACH FOR COW UDDER DETECTION is my own work and has not been submitted previously by me for taking any degree from this university BAHRIA UNIVERSITY ISLAMABAD or anywhere else in the country/world.

At any time if my statement is found to be incorrect even after my graduation, the University has the right to withdraw/cancel my MS degree.

Name of scholar: ARIF ZAMAN

Date: 07 FEBRUARY 2020

PLAGIARISM UNDERTAKING

I, solemnly declare that research work presented in the thesis titled "A DEEP LEARNING APPROACH FOR COW UDDER DETECTION" is solely my research work with no significant contribution from any other person. Small contribution / help wherever taken has been duly acknowledged and that complete thesis has been written by me.

I understand the zero tolerance policy of the HEC and Bahria University towards plagiarism. Therefore I as an Author of the above titled thesis declare that no portion of my thesis has been plagiarized and any material used as reference is properly referred / cited.

I undertake that if I am found guilty of any formal plagiarism in the above titled thesis even after award of MS degree, the university reserves the right to withdraw / revoke my MS degree and that HEC and the University has the right to publish my name on the HEC / University website on which names of scholars are placed who submitted plagiarized thesis.

Scholar / Author's Sign: _____

Name of the Scholar: ARIF ZAMAN

In memory of my father

To my beloved mother with love and eternal appreciation

ACKNOWLEDGEMENT

In preparing this thesis, I was in contact with many people, researchers, academicians, and practitioners. They have contributed towards my understanding and thoughts. In particular, I wish to express my sincere appreciation to my thesis supervisor, Professor Dr. Shujjat Khan, for encouragement, guidance, critics and friendship. I am also very thankful to my principal supervisor Professor Dr. Khalid Javed. Without their continued support and interest, this thesis would not have been the same as presented here.

My fellow postgraduate students should also be recognized for their support. My sincere appreciation also extends to all my colleagues and others who have provided assistance at various occasions. Their views and tips are useful indeed. Unfortunately, it is not possible to list all of them in this limited space. I am grateful to all my family members.

ABSTRACT

Image recognition with deep learning focuses on identifying one or more specific objects, class or feature in a specified image or video frame. Deep learning has received a lot of attention recently and is yielding unprecedented results. Image recognition is used in different applications, such as Medical diagnosis, Automatic inspection of manufacturing products and tasks like pedestrian detection. In the dairy industry, teat spraying is essential in order to prevent the spread of diseases like mastitis. However, the current automatic teat spraying/cup attachment methods have a lot of issues. For example, teat location coordinates are assumed to be fixed and are pre-stored while in reality, it is more likely that the cow might be more mobile, hence it is a flawed approach. Secondly, IR based systems are inconsistent due to the light absorption property of IR which has no guidance for exact detection. To address this problem, it is required to attach the cup or spray the teats only when the udder is detected. This thesis makes use of deep learning for the detection of cow udder which can be of great importance in the dairy industry which is currently relying on conventional methods that are inefficient and inconsistent. Though Deep Convolutional Neural Network is the most advanced technique for image recognition but it demands a great deal of training time and computational power. Focusing on the problem of image recognition with limited time for training and restricted computing power in mind, a technique called “Transfer Learning” is used in this thesis. This technique makes use of the pre-trained model in order to accelerate the process of learning. Trainable parameters from Xception Deep Convolution Neural Network (DCNN) model were transferred for identifying cow udder. The model training was done in three stages; the data collection or dataset building stage which includes acquiring cow udder images and manual segmentation then training stage and finally the validation/testing phase. The model is trained on our custom made cow udder dataset. With overall miou_1.0 of 0.88. The model was successful in accurately detecting udder.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVATIONS

ANN	ARTIFICIAL NEURAL NETWORK
CNN	CONVOLUTIONAL NEURAL NETWORK
DCNN	DEEP CONVOLUTIONAL NEURAL NETWORK
RNN	RECURRENT NEURAL NETWORK
GPU	GRAPHICS PROCESSING UNIT
CRF	CONDITIONAL RANDOM FIELD
SPP	SPATIAL PYRAMID POOLING
ASPP	ARTROUS SPATIAL PYRAMID POOLING
ReLU	RECTIFIER LINEAR UNIT
TPU	TENSOR PROCESSING UNIT
MIOU	MEAN INTERSECTION OVER UNION

CHAPTER 1

1 INTRODUCTION

This chapter presents the problem statement, objective and structure of the thesis.

1.1 Overview

Machines are becoming intelligent with time as humans have always tried to bring computers up to the mark of human efficiency. In order to behave like humans, computers need to understand language, memorize, learn and deduct. All of this is achieved under the realm of artificial intelligence[1, 2]. With the advancement in computer technology, artificial intelligence has become one of the leading branch of computer studies due to its exceptional results in terms of performance and efficiency in variety of fields.

Deep learning is a subset of machine learning. It uses multi-dimensional non-linear structure for analyzing data[2, 3]. Deep learning makes use of hidden patterns from the training set with the aim of processing it like a human. Number of deep learning architectures are available like deep neural network, deep belief network and deep convolution neural network for ranges of tasks like speech recognition, bioinformatics processing, computer visualization and natural language processing, all of which are progressing exceptionally well [4, 5].

Deep learning for image recognition is also made to work as a human, starting from observation, gaining information from the observation and presenting it in intensity value, region or a specific shape. Models are created from large scale data in an attempt to make accurate representation[6].

1.2 Problem Statement

In mechanized dairy farms, inefficient cow teat cupping and spraying is one of the main problem. Current attachment/spraying systems have severe limitations in udder

detection consistency because these approaches employ conventional sensor or rigid methods. These detection methods for udder requires a controlled environment and can be disturbed by slight movement or minor object in front of them. Furthermore these detection methods do not perform well with changing light, cow tones, udder shape and position of the cow. Deep learning in the recent past has improved greatly and has achieved high accuracy in image recognition tasks. The dairy industry can benefit from the advancing deep learning technology by having a generalized system that can adapt to ambient conditions.

1.3 Objective

The main goal of this thesis was to segment the cow udder using deep learning technique. The main objectives of this research work are as follow.

- Training a deep learning model for reliable udder detection, in order to improve the current teat spraying/cup attachment system in the dairy industry.
- To find and make the udder dataset which can be used in future too.
- Building a model which in future can be employed for udder anomaly detection.

1.4 Methodology

The research work focuses on detecting cow udder using technique of “Transfer Learning”. For better use of the technique, a range of pre-trained models were studied. Along with it, research specific dataset was compiled by collecting images from different dairy farms and some taken from online sources such as Google database; because of its global access. The acquired images were then carefully labeled using computer software “Lableme”. The dataset was split into two sets i-e training and testing/evaluation. The training set contains 90% of the images while 10% goes to testing/evaluation set. For fair comparison and to reduce overfitting of the model the testing/evaluation images were not presented during training. The process was concluded in four major steps. Starting from acquiring images for the dataset, then the images were labeled properly, in the next step pre-trained model was selected and was trained on our custom dataset and finally, the

model was tested/evaluated on test data. Further details are discussed in later sections of this thesis.

1.5 Structure of the report

The thesis is presented in five chapters. It opens with the first chapter titled **"Introduction",** followed by a chapter titled **"Background"** which consults existing research to inform and describe the concepts and terminologies which are later used in this thesis. The **"Literature review**" chapter includes different relevant researches along with the critique of different existing models. It is followed by the **"Implementation"** chapter, which discusses the relevant methods and enlists detail of all the experiment related steps for the proposed solution, then the **"Results"** and finally the **"Discussion"** chapter presents comments and analysis of the proposed method.

CHAPTER 2

2 BACKGROUND

For almost a century there has been a practice of automating the dairy industry to decrease the labor cost. There has been considerable advancement in the automation of the dairy industry and number of tasks has been automated. The process of milking in dairy farms is repetitive and will benefit significantly from automation. Several automatic milking systems have been developed with the aim of reducing the labor cost. However, the development of a system for automation of the milking process which includes cup attachment has posed some technical considerations. Accurately positioning the robotic arms have some major problems, like tail may come in way of the arm, cow may be in a different position, or teat of cow may not be in the exact same position. The said problems have been looked into for many years and different systems has been developed for it, like an automatic cup attachment system invented by Notsuki and Ueno[7] which requires position of individual cow teat to be stored in the memory of the system, the said system also requires the cow to be always in the same position. One similar system invented by Akerman [8] uses sensors that require physically approaching and touching the teat for identification with no information about cow positioning. An infrared-based system also exists for detection of the teat, as the temperature difference in cow udder and teat is slightly different though not different enough for reliable detection, however, the temperature gradient was used by adjusting the sensor to the different temperature gradient and creating different level of sensitivity so that the teat can be detected. All the detection systems in current commercial automatic milking market uses laser systems, in which the camera is used for detection of the location of laser stripe. To triangulate the position of the teat stripe in the camera, image is coupled with relative distance between camera and laser's incidence angle[9]. The laser-based system fails with even a small obstacle in front of it. Deep Neural Network is said to be the most disruptive class of technology in the next decade, however, the technology still waits to be in use for commercial purposes in the dairy industry. Machine vision is of great importance in automation, as the technology has seen tremendous improvement in the recent past.

2.1 Deep Learning

Machine learning has always been limited in processing natural data in its original form. It requires careful engineering to transform raw data into suitable feature vector, from which classifier can classify patterns[10]. The solution to this problem is deep learning which allows building complex non-linear functions whose weights can be changed during training for learning a specific task. These features are part of the learning process and are organized in a hierarchy. Each layer's output is input for its successor layer. The model becomes deeper as the number of layers increases hence called deep learning. Deep learning is evolving with a fast pace as new architectures are regularly designed[11]. These architectures includes the standard architecture which is the fully connected multi-layer network, the Recurrent Neural Network (RNN) and Convolution Neural Network (CNN) which are further discussed in section 2.3. Deep learning can either be supervised or unsupervised. The dataset used in this thesis was labeled data with no preprocessing hence making it supervised learning.

2.2 Artificial Neural Network

The construction of Artificial Neural Network (ANN) was inspired by the human brain. ANNs are made up of neurons, which just like human nerve cells are connected to each other[6]. ANNs comprises of nodes which have weighted inputs and outputs along with activation function. The basic structure of neuron can be seen in figure 2.1

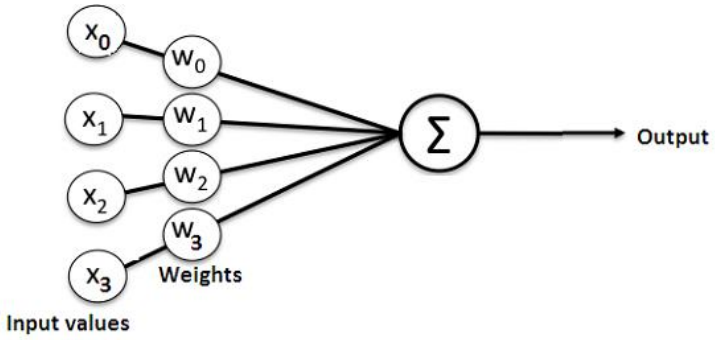

Figure 2.1 Structure of single Neuron of a Neural Model

As given in the figure the output of the neural model is function of input x_i and weight w_i. The output is computed as follow:

$$\text{Output} = \mathrm{f}\left(\sum_{i=0}^{3}(w_i x_i + b)\right) \qquad (2.1.1)$$

Every input value is weighted, multiplied by corresponding weight along with the addition of bias b and is then passed through an activation function.

A neural network is a combination of large number of such neurons with number of layers. Each layer in a neural network is connected to the one following it which means the output of each layer becomes the input for the upcoming layer. Figure 2.2 shows a simple feed-forward neural network known as multi-layer perception[12].

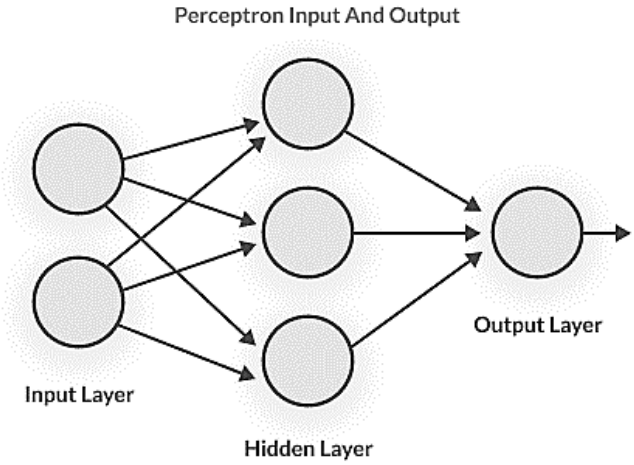

Figure 2.2 Multi-Layer Perceptron

As obvious in figure 2.2, neurons are grouped in each layer and every layer is connected to its subsequent layer. Each layer has its own bias parameter used for computing output of the given neuron. The layer to the very left is called the input layer, the one in the middle is known as the hidden layer, whose value cannot be seen during the training process, while the layer to the rightmost is known as output layer[13]. The correct size of the hidden layer is of great importance in the implementation of ANNs. If a proper amount of neurons are not determined in the hidden layer, the system may not be able to generalize the unseen instances. If too many nodes are used in the hidden layer it may cause over-fitting and the system may not find the desired output as discussed by Kon and Plaskota [14]. ANNs have a main advantage in processing data with high dimensional features like images but this comes with a cost, that is the requirement of high processing power and its complexity for an average user[15, 16].

2.3 Convolutional Neural Network

A Convolutional Neural Network (CNN) is a common deep neural network made of auto optimizing neurons that were first introduced in [17]. With the growth of the computational power of computers (use of GPUs) CNNs are well-spread. Compared to ANN, Convolutional neural network has a wide range of application in pattern recognition in images. As it encodes specific features of images into the network architecture, which makes it more suitable for image feature learning[18]. CNN consists of five basic elements; the input layer which holds the pixel value of the image, the Convolutional Layer calculates convolution of the input layer, the non-linear layer which applies non-linear transformation on previous layers output, the pooling layer reduces the number of parameter of activation by performing sampling operation and finally the fully connected layer show score of class from activations, which are used for classification[19]. Further explained below.

2.3.1 Convolution Layer

Convolution is the front layer of CNN and it performs heavy calculation of the CNN operation[20]. As a convolution neural network extracts various features from the input, the convolution layer is assigned to extract low-level features from the image, like lines, corners and edges. One of the key feature of the convolution layer is the usage of kernel. Kernels are also known as filters or feature detectors. Generally kernels are of small spatial dimension but can spread along the whole input[21, 22]. The convolution layer convolves each filter to the overall input and generates a two-dimensional activation map. Different activation map for every kernel is stacked along the depth. Therefore it is necessary to have same depth of input and filter[23, 24]. The basic operation of the kernel is visualized in Figure 2.3. The kernel glides through the entire vector after which the output generated which is the scalar product of every value inside the kernel.

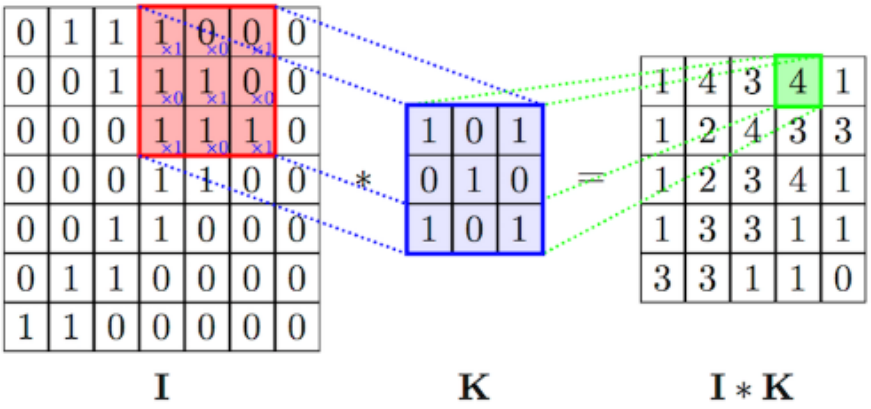

Figure 2.3 Kernel Calculation in the Convolution layer[11]

Commonly kernel operation starts from the top left of the image and keeps sliding around the whole pixels of the image and multiplies the value of the kernel with image pixel value which is then summed up. Every input part generates its own number. When a kernel is passed through all parts of the image, an output array is generated called the feature map. CNN compared to ANN reduces the complexity of the model using a kernel. Depth, stride and zero-padding need great consideration in designing the convolution layer. Depth corresponds to the number of kernels, by reducing depth we can reduce the number of neurons in the network which can degrade the performance of CNN. Stride corresponds to steps of the sliding kernel if set as 1 the kernel will move just one pixel at a time. By setting stride to higher value amount of overlapping area will be reduced. Filling zeros around the borders is called zero-padding, which gives us control over the spatial size of the output. All of these parameters can either increase or decrease the spatial dimensionality of output [25].

2.3.2 Non-Linear Layer

In order to identify features from a hidden layer, a convolution neural network uses non-linear transformation. In ANN the non-linear transformation functions are hyperbolic tangent or sigmoid. In image processing, the greater the sparsity of data, the better the results.

2.3.3 Pooling Layer

The pooling layer is used to reduce computational complexity, dimensionality and number of parameters. It also helps in making features strong against distortion. Two classic pooling functions are average pooling and max pooling.

2.3.4 Fully Connected Layer

Previous layers are repeated several times after which data comes to a fully connected layer. In fully connected layer neuron are connected to two adjacent layer's neurons. Fully connected layer add up weights of features that come from earlier layer and show the probability of class[26].

2.4 Transfer Learning

Transfer learning refers to storing gained knowledge while training a model for a specific task and use that stored knowledge for another task which can be similar. Some approaches use doing category adaptation which doesn't need training for the new task, while some uses transfer learning for initialization which requires training for the new task.

2.5 Image Segmentation

Image segmentation is the process of assigning labels to each pixel in an image and partitioning it into multiple parts. Segmentation changes the representation of an image into something meaningful which can be easier to analyze. In this thesis, segmentation was used for labeling cow udder.

CHAPTER 3

3 LITERATURE REVIEW

Object recognition has always been a challenging task in computer vision, some of which are documented and can be traced back to the 1960s[27]. Initially, the constraint setting was included and main issues like light variation and clutter were ignored and only object under varying viewpoints was in focus. The first work on object categorization started in 1980 with a limited number of classes like digits and faces under a very controlled environment[28, 29]. Recently, the task of classification is focused on natural settings using a wider range of classes[30, 31]. This challenge has helped in the development of new advanced classifiers. A number of CNN based models are available currently which are performing exceptionally well in object recognition. Several pre-trained models are available for segmentation and classification tasks which can be modified using transfer learning. These models are trained on larger datasets like ImageNet and Pascal VOC for a specific task [32]. These trained CNNs models with weightless binary checkpoints are commonly used as weight initializers for training the same model for another task. It has also been tested that using these pre-trained models can give great improvement in the classification task[33]. Deeplab provides different modern techniques for better segmentation with pre-trained models which are discussed in Section 3.1. Transfer learning has been used in a lot of research works for classification. This thesis has some similarities with [34]. They used deep learning technique with transfer learning for artificially weaning of calves by deterring them from suckle, which requires to detect udder from a certain distance. The trained model is embedded into a device and mounted on collar of calves. The method they have used is bounding box detection, which although gives better accuracy when compared to segmentation models but the bounding box are mostly overconfident about area, which results in too small or too large mask. This makes the bounding box not suitable for the task that requires accurate segmentation with different shapes.

A transfer learning technique being used for trees classification from images acquired from satellite [35]. Initially, images are segmented and pre-trained model VGG-

16 was used to classify the subject. Result when compared with Random Forest shows that the VGG-16 network outperforms both Random Forest and Gradient Boosting with an accuracy of 92.13% [35]. D. Rathi et al. [36]describe a method to classify fish species. Using image pre-processing, noise is removed from the dataset and then DCNN is used for the classification task. Wang et al.[37] has used a pre-trained CNN model for action recognition task with a small training dataset and has achieved significant accuracy. YOLO model has been used in [38] for the detection of flame in the video segment and compared with other methods and accuracy of 76% is obtained. Some inspiration for this thesis comes from a machine learning approach explored by Warner, D., et al. for detection and monitoring of lameness in dairy herds, in which both decision tree based and machine learning approach are used, the machine learning approach has achieved better result comparatively[39]. Yang, A., et al. has used image analysis technique based on fully convolutional neural network for recognition of sow drinking, feeding, movement, activity and inactivity for facilitating farmer for improving livestock management[40].

3.1 Deeplab V3

The architecture of DeepLab has different versions: version v1[41],v2[42],v3+[43], each of them on their release achieved state-of-art performance. DeepLab v1 and v2 architecture are integrated with two main components: a Deep Convolution Neural Network (DCNN) for coarse pixel-wise prediction and for edge refinement fully connected Conditional Random Field (CRF) is used. For multiple scales, version 2 uses artrous spatial pyramid pooling ASPP module for effective segmentation. The CRF is excluded in version 3 and an improved ASPP module is added. The current v3+ version is an improvement over version v3 as a decoder module is added at the end of the network for refine localization particularly along the boundary of the object. For this research thesis, we have used the current version v3+ of Deeplab architecture. Three main problems in the application of DCNNs in semantic segmentation are addressed in DeepLab architecture: signal resolution reduction, spatial invariance and objects at multiple scales[44].

The problem of signal resolution is resolved by applying *atrous convolution* also called dilated convolution. Zeros are added between active filter taps for upsampling the

filters, this helps in enlarging filters field of view and densely mapping computing feature without increasing computational cost or losing receptive. The problem of spatial invariance occurs due to the usage of DCNNs for semantic segmentation task. Correct prediction of the image-level label with spatial accuracy is not important in image-level classification but in sematic segmentation spatial accuracy is of great importance. This problem is addressed in DeepLab v3+ by using a decoder module at the very end of the network to restore local spatial information. Decoder module help in the recovery of accurate object boundary without post-processing. CRF was used for this function in previous versions of DeepLab[45]. Chen et al. show that the decoder module gives better performance than CRF at object boundaries[42]. The third issue is addressed by applying artrous spatial pyramid pooling (ASPP) with varied artrous rates of different fields of view and combining the result into one feature map. With solving these problems of DCNN, DeepLab v3+ has achieved a state of art performance over PASCAL VOC 2012 dataset[43].

3.1.1 Artrous Convolution

Inspired by artrous algorithm for wavelet decomposition the artrous uses modified convolution operation[46]. In order to increase the receptive field of convolution filter without increasing model parameters, artrous convolution is used. Practically it adds zeros between active filters taps. Artrous rate parameter defines the number of zeros. Mathematically it can be expressed as follow.

$$Y(i) = \sum_k F(i + k)G(k) \qquad (3.1.1)$$

Y, F and G are two-dimensional value. While F represents the input signal, G is filter and Y is output value. An additional parameter: artrous rate r is introduced in artrous convolution as follows.

$$Y(i) = \sum_k F(i + rk)G(k) \qquad (3.1.2)$$

The factor r dilates the input signal F during convolution while G remains constant the make it easy to tune the artrous rate to calculate feature response at various scales. The concept can be seen in Figure 3.1.

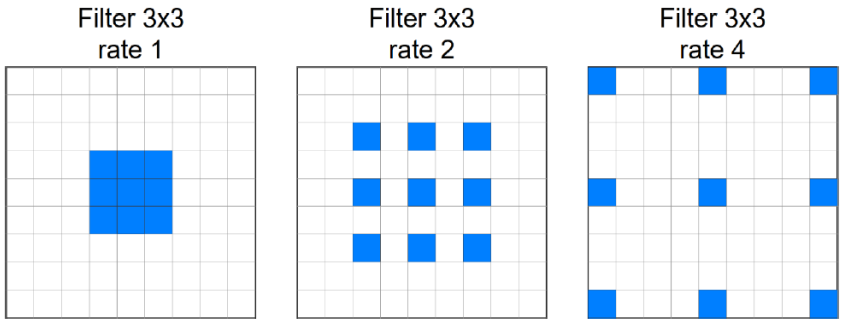

Figure 3.1 Artrous Convolution with 3x3 filter [41]

As shown above, the number of parameters are kept the same while the filter's field of view increases with artrous rate. For capturing contextual information at different scales we need to constantly down-sample the signal, if artrous convolution is not used. Artrous convolution can be used for computing feature map densely without having to increase the cost of computation or model complexity.

3.1.2 Artrous Spatial Pyramid Pooling (ASPP)

In ASPP there is a parallel convolution of feature map with various artrous rates and the output features map from various branches are combined to from final output. He et al. [47]was first to implement SPP in CNN and ASPP is inspired from it. The ASPP module is positioned at the end of the last convolutional layer of the encoder. Convolutional feature map sampling is done using 3x3filter and changing the artrous rate. Figure 3.2 shows the ASPP module.

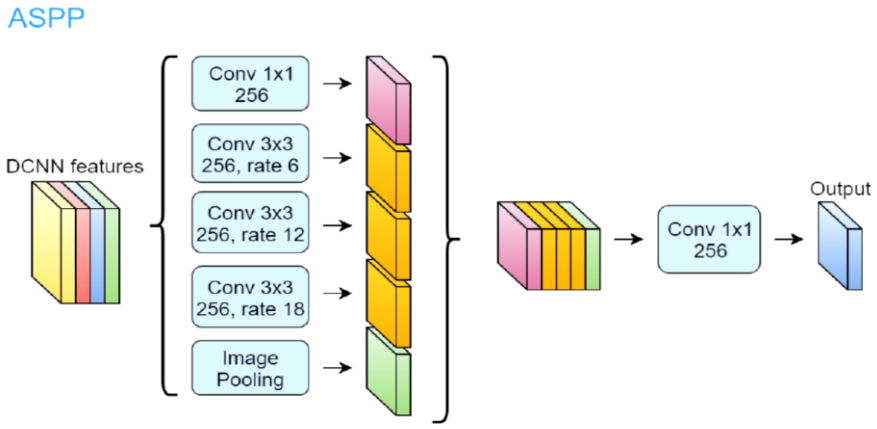

Figure 3.2 Schematic of DeepLab V3+ ASPP Module[41]

The output feature map of DCNN becomes an input for ASPP and its structure is depended on the network backbone. For capturing multiscale context the feature map is sampled with changing artrous rate. There are 256 filters in each convolution branch of ASPP thus giving an output of 256. Concatenated outputs from all branches are passed through 1x1 convolutions with 256 filters. The output of ASPP modules goes to the decoder module.

3.1.3 Decoder Module

For tasks where accurate spatial information is needed like human pose calculation, semantic segmentation and object detection, Encoder-Decoder networks are proved successful[43]. Encoder module is used to encode input to smaller resolution feature vector with rich semantic information but most of the spatial information is lost. Encoder output can be used for classification. For semantic segmentation decoder module is used for recovering lost spatial information. The decoder module combines low-level rich semantic segmentation information along with accurate spatial information from the previous layer of the network. Decoder Structure is shown in Figure 3.3.

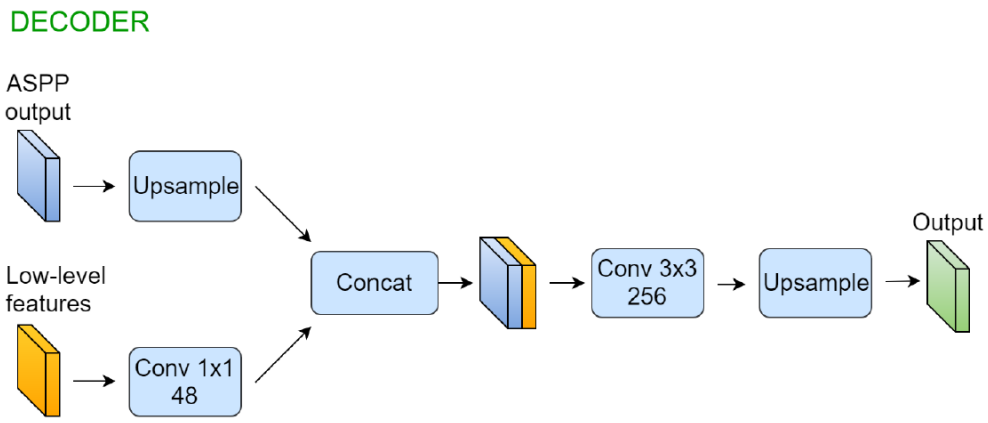

Figure 3.3 Decoder DeepLab V3+[41]

3.2 DeepLab Networks Backbones

Deep comprises of different network backbones. Backbone here means DCNN block which is the most important part responsible for feature learning and representing

objects to be segmented. Different models like Xception, MobileNet v2, ResNet are discussed in detail in the upcoming section. Xception has numbers of variants like 41, 65 and 71, ResNet has two variants 50 and 101; the number next to the model represents the depth of the network.

3.2.1 Xception

Xception has spatial correlation and cross channel in the CNN feature map which can be decoupled[48]. This is different from the common convolution method in which each filter learns both spatial correlation and cross-channel. The ideas came from the Inception architecture in which cross-channel and spatial correlation is partly decoupled[49]. In inception architecture, the input layer is convolved with various parallel filters. To reduce dimension convolution of 1x1 filter and each of the parallel branches is performed. Xception pushes this to an extreme as the name suggests “Extreme Inception” by complete decoupling of cross-channel and spatial correlation. Xception uses depth-wise separable convolution in which first spatial convolution is performed on every channel, followed by 1x1 convolution for capturing of cross-channel correlation. Inception has a better performance compared to inception in different classification tasks without the need to increase model parameters. Xception architecture contain 36 layers[48]. The modified Xception architecture have more layers, max-pooling has been replaced with batch normalization and ReLU activation after 3x3 convolution. Modified Xception model architecture can be seen in Figure 3.4. For this thesis, the Xception 65 model is used.

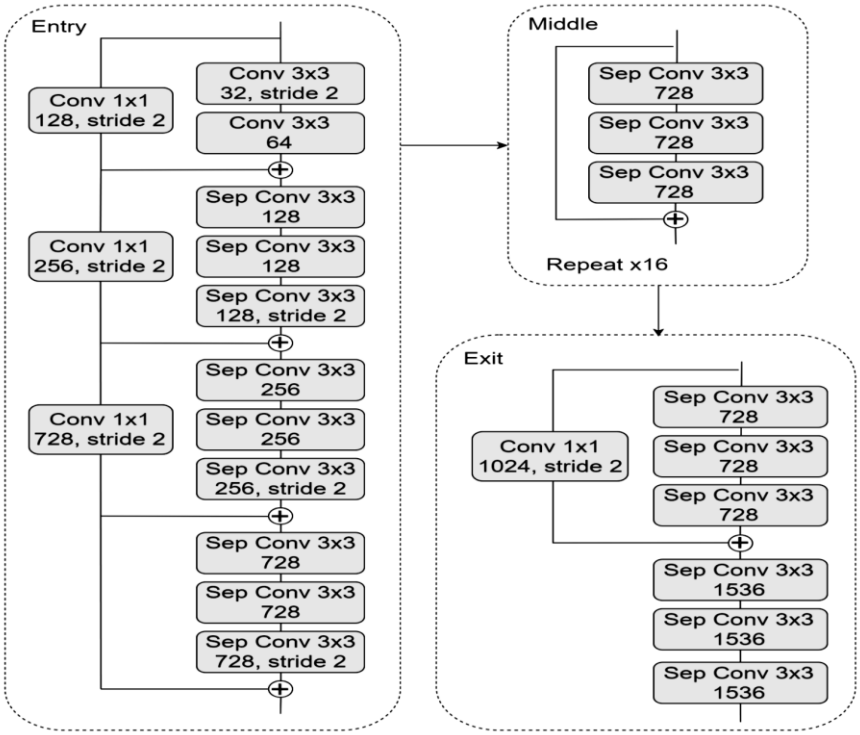

Figure 3.4 Xception 65 Architecture used in Deeplab V3[48]

3.2.2 MobileNet V2

Proposed by Sandler et al.[50] MobileNet v2 is deep neural network architecture with lightweight for mobile devices. Just like Xception, MobileNet v2 also uses depth-wise separable convolution to minimize the computation cost without effecting accuracy. Depth wise separable convolution with k sized kernel reduces the computational cost by a factor of k2 when compared to standard convolution. There is an extension of two components in original architecture: linear bottleneck and inverted residual. In ResNets residual connection connects input and output of block, while to reduce channels 1x1 bottleneck is used. However, in MobileNet v2 the intermediate layer is expanded by a factor known as expansion ration. The problem of traditional activation which discards information as negative values are set to zero is often tackled by the addition of more channels. In MobileNet v2 1x1 bottleneck is removed. Sandler et al.[50] Shows that linear activation improves the performance of the model. For fast interface and reduced accuracy Deeplab decoder and ASPP modules are dropped. Architecture can be seen in Figure 3.5

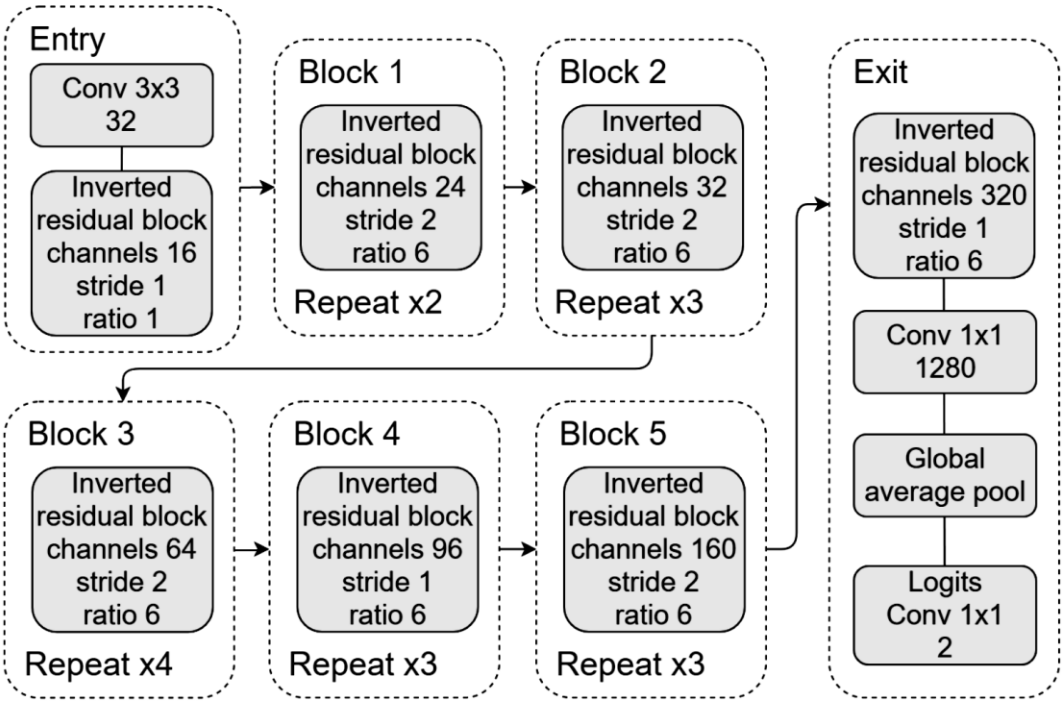

Figure 3.5 MobileNet v2 Architecture[50]

3.2.3 Inception v3

Inception v3 has 44 layers and learnable parameters of 21 million[51]. The inception module shown in Figure 3.6 is the main block for Inception, which works on the hypothesis that spatial correlation should be separately mapped from that of cross-channel[49]. These correlations are simultaneously correlated by a regular convolution kernel. Using 1x1 filter the inception module first finds cross-channel correlations, higher numbers of input features are mapped to the lower number of output feature map. Then the spatial correlations of every output feature map are found using 5x5 and 3x3 convolution kernels. The inception module uses a small 3x3 filter for spatial covering of smaller areas for accurate detail of the image, while the 5x5 filter is used for larger areas. Along with inception module Factorization of filters are used which uses two 3x3 cascaded filter replacing 5x5 to reduce computational power, as both of them have the same output result[51]. Using batch normalization data is whitened and the response for all the neural map is made in the same range with zero mean value. This allows the use of a high learning rate with faster speed by minimization of regularization requirements.

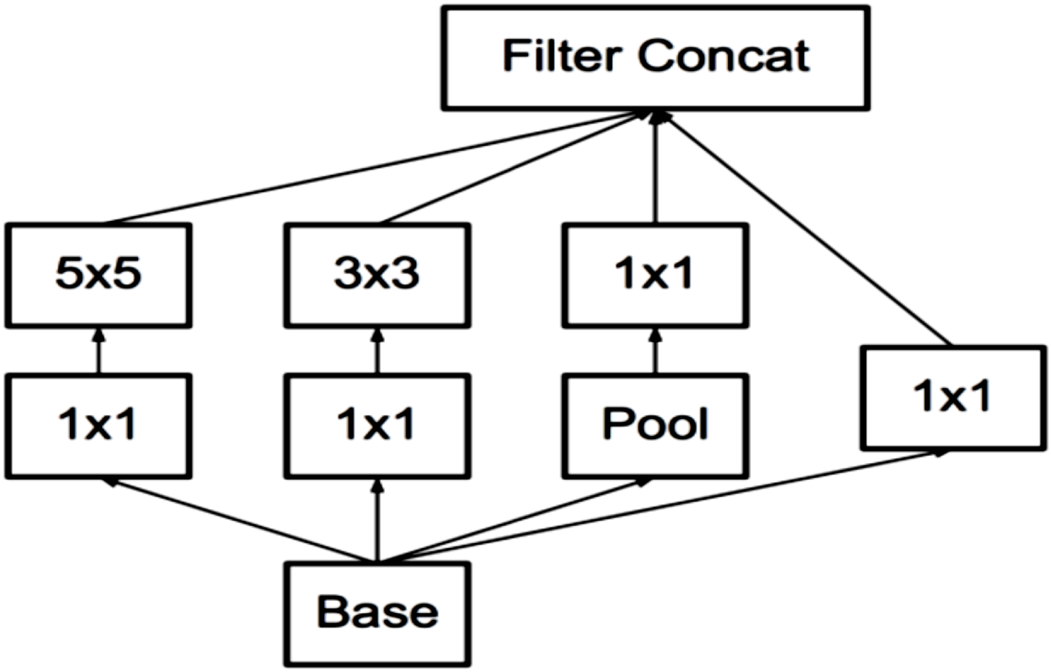

Figure 3.6 Inception Module[52]

3.2.4 ResNet

To solve the problem DNNs training ResNet was created. Though, it was acknowledged that depth was important in the performance of the model however earlier deep network had 16 to 30 layers. Simply adding more layers doesn't solve the problem as it degrades gradient to a small number when it backpropagation through the network. Different normalization schemes were introduced to address this problem. Along with degraded gradient performance also decreases when layers are increased. By effectively deactivating layers with the help of simple mapping by keeping the input same, denser model can perform better compared to shallow models. He et al.[53] Added direct connection around the convolution layer which made it easier to execute identity mapping. The residual network implemented in DeepLab v3+ has 50 and 101 layers, which can be seen in Figure 3.6.

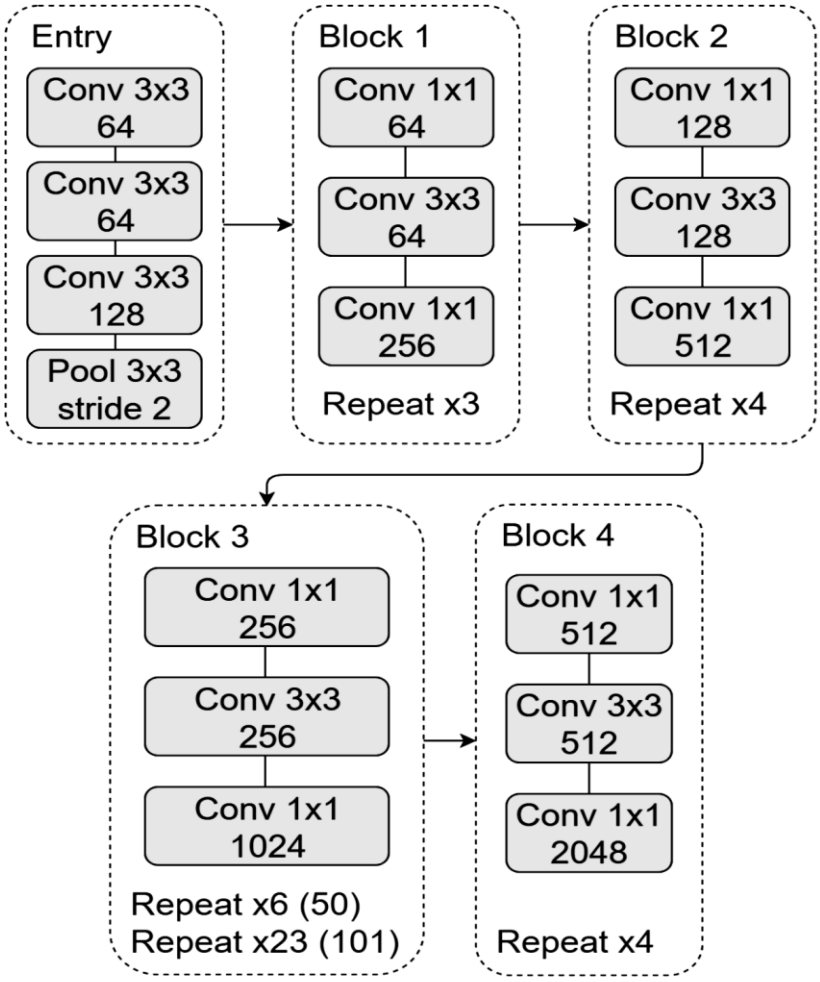

Figure 3.7 ResNet Architecture with 50 and 101 layers[53]

CHAPTER 4

4 IMPLEMENTATION

4.1 Preparing Dataset

Data is of key importance in machine learning. In order to get a proper dataset for this research work all public datasets were checked, no such dataset was publically available for this task. Further different research groups were also contacted in order to get the required data if available. Data used in this research work is a custom made dataset with raw images of cows in natural ambient conditions. Data acquisition was the most laborious part of this research work due to the unavailability of already acquired images, access to the dairy farm and lack of data acquisition system. After image acquisition, each sample of udder had to be manually labeled. The ground truth annotation process is presented below.

4.2 Annotation

Manual annotation of images is a tedious and time-consuming task. There is no auto annotation tool that is accurate enough for this critical task. LabelMe Annotation tool was used for the annotation task of this research work. Although larger datasets are meant for large size and greater accuracy, but due to time constraint, we had limited ourselves to a sufficiently large dataset that can produce a better-segmenting model. In our dataset preparation, 600 images were selected and subjected to annotation. Figure 4.1 shows dataset samples of ground truth overlaid annotations

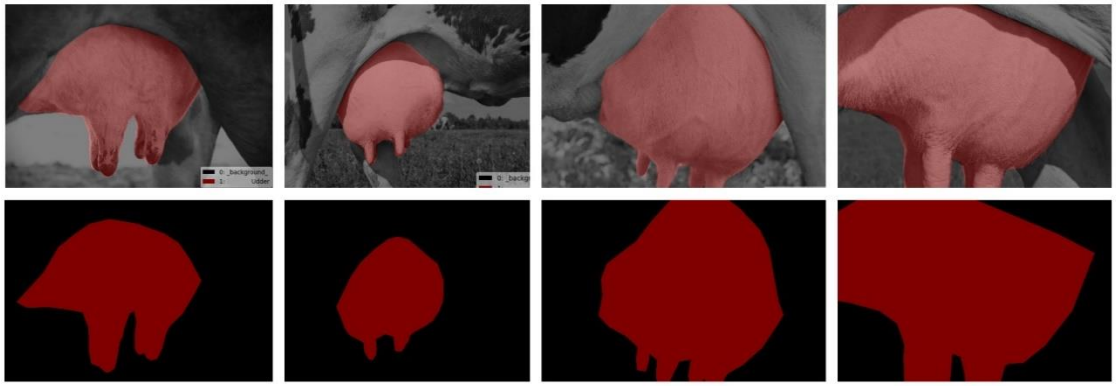

Figure 4.1 Ground truth images along with annotations.

Each image was labeled using polygon annotation by defining vertices. Based on visuals pixels belonging to udder were annotated. Each udder was labeled carefully so that no extra region is included to make it easy for the model to efficiently separated udder. Based on angle some udders are few pixels large while some are big enough and in some cases, the teats are not visibly separated in which case no gaps were left between the teats in polygon label.

4.3 Tools

Different tools both hardware and software used in this research work directly or indirectly, are discussed in this section. This section will also focus on the advantages of these tools and why were they chosen.

4.3.1 LabelMe

A graphical image annotation tool used in the annotation of data in this thesis, LabelMe is written in Python while for graphical interface Qt is used[54]. In order to annotate images in LabelMe, a single image or whole folder can be loaded. One dataset is loaded in it, a random image is displayed over screen. With the help of a mouse, the user can draw a polygon around the object, after closing the ends of a polygon a popup opens from where the user can select a label or create a new label. Once the annotation is complete the label is auto-saved to the same directory from where the file is loaded. The saved polygon label can also be deleted or changed later. Each class of label is displayed

with a different color which we can choose. The output label is saved in JSON format which can later be converted to the user's desired format such as PASCAL VOC, which was used in this thesis. The python script for the conversion is free available on the LabelMe GitHub directory. This software was download and used on Ubuntu 18.04. The software interface of LabelMe is shown in Figure 4.2.

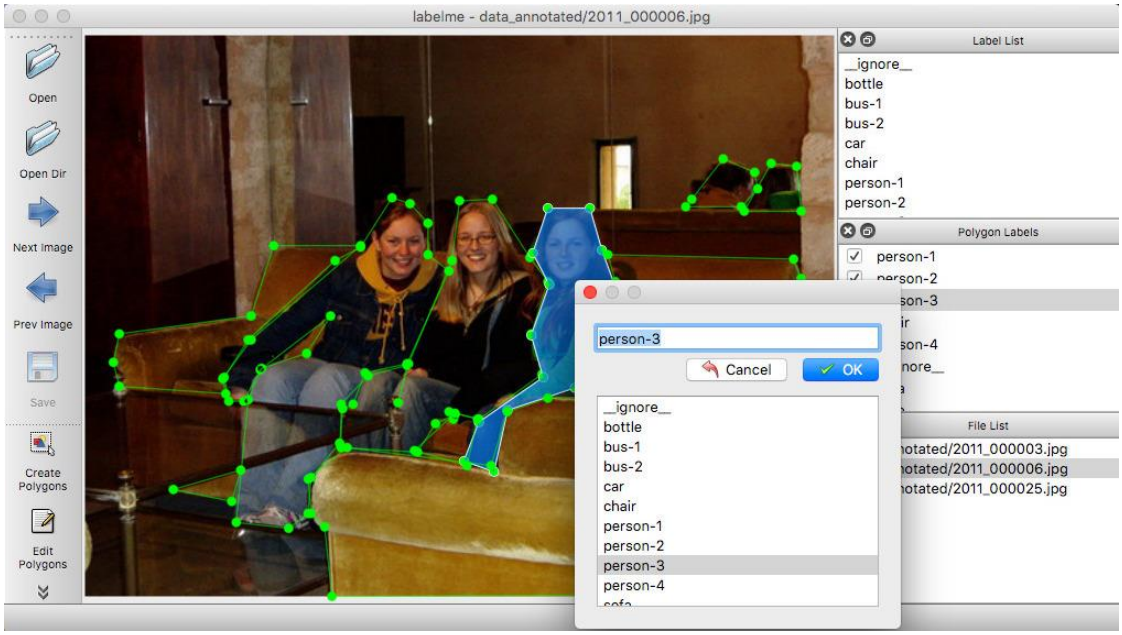

Figure 4.2 User Interface of LabelMe

4.3.2 TensorFlow

Tensor Flow is a framework for deep learning developed by Google for internal use. Released in 2015 as open-source software for developing and training neural models and since then a lot of users are contributing to it. Tensor Flow is written in python, whereas data expression is in the form of a multi-dimensional array known as a tensor. Each node represents arithmetic operation, the distinct calculation is visualized as a dataflow graph. Computation can be deployed over multiple CPUs or GPUs due to its flexible architecture, this significantly increases the processing speed. Apart from this TensorFlow offer a range of tools like graph visualization and tensor board. To interact with computational graphs TensorFlow uses sessions. The main purpose of the session is graph initialization for all variables as well as running the graph. Google being the highest investor in artificial intelligence has developed a new processing unit known as Tensor

processing unit or TPU, specially designed for machine learning tasks and it run flawlessly with TensorFlow. The TPU hardware is optimized for machine learning tasks. The second generation of TPU was released in 2017 with a better performance of two hundred teraflops. TensorFlow GPU version 1.14 was used in this thesis for the creation of a new model using transfer learning from existing models. Further detail about model training is in later section of this chapter.

4.3.3 Google Colaboratory

Google Colaboratory or simply known as Google Collab is a project of Google created to spread machine learning. Collab is a Jupyter notebook except that it doesn't require any setup and can be run on the cloud. It comes with support for major machine learning libraries with ease of use. It is entirely free to use with a session time of 12hrs. As discussed in the earlier section that Google had developed a new processing unit TPU for machine learning, it is free available on the Collab platform. Apart from this Collab also offer support of GPU for free with an aim to make it standard for machine learning. Figure 4.1 Show Google Colaboratory running in Google Chrome. The main features that Collab offers are as follows.

- Supports Python 2 & 3
- Support for upload/create/download of the notebook.
- Import from Github
- Data access from Google Drive
- Dataset can be imported from external sources like Kaggle
- Libraries like Keras, TensorFlow, PyTorch and OpenCV
- Free Cloud GPU service

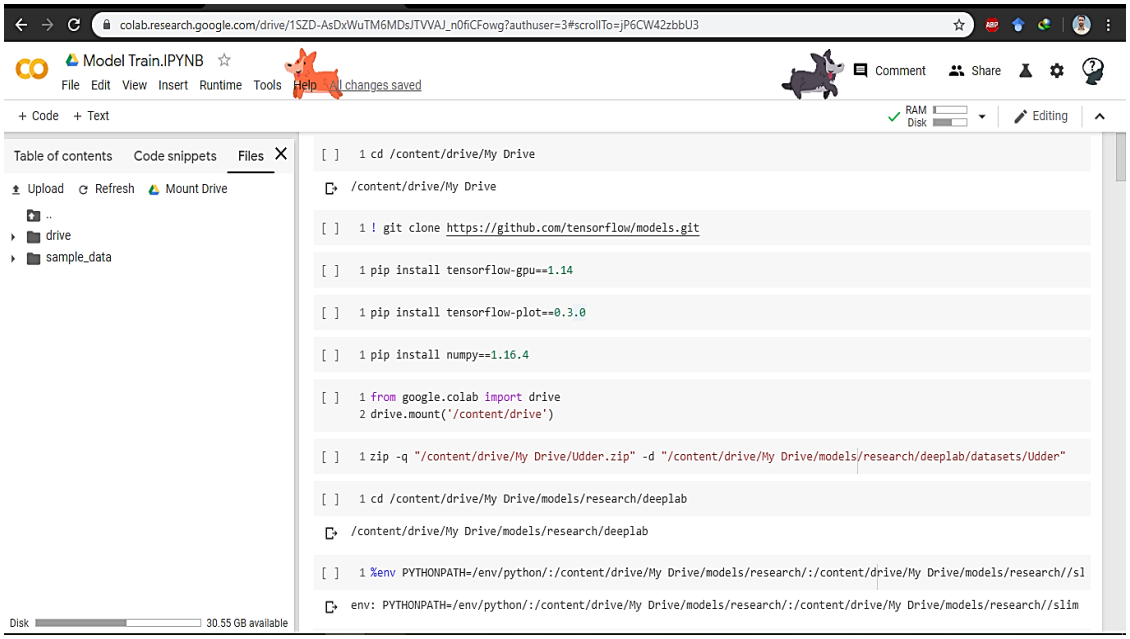

Figure 4.3 Google Colaboratory running in Google Chrome

The amount of available hardware resources varies over time on Google Collab, there is no way for self-selection of hardware. Google auto allot memory based on usage and computation the user is doing. The user if used more hardware resources recently are more likely to get lower resources temporarily, Moreover Collab also offers runtime over a local machine. Though the hardware resources vary on Google Collab, the available resources during the training of the model were as follows.

Table 4.1 Hardware Specification of Google Colaboratory

CPU Model	Intel (R) Xeon (R)
CPU Clock	2.30 GHz
Available Disk Space	68Gb
Available Memory	12GB
GPU Model	Tesla T4
GPU Memory	15GB

4.4 Model Building

Model in machine learning means training a machine learning algorithm on a dataset, which can then be applied on novel data to predict the output. There are certain steps to follow. First, we need to collect data, prepare the collected data, select a model, train the model, evaluate and then visualize the model. The first step which is data collection is of great importance in machine learning, as the success of the model depends mostly on data. For the purpose of this research, the required dataset wasn't available as discussed in the Dataset section and was manually collected. The dataset consists of two classes including the background with a total number of 600 images. The data preparation stage was the most time consuming, as for this research work segmentation task was done manually. After properly segmenting the data, the dataset needs to be converted into a proper format which will be compatible with the selected model. The output format from the annotation software LabelMe was JSON file which was not the required format for the selected model and needed to be converted into PASCAL VOC format. Through public available python script on LabelMe GitHub directory the JSON format annotation was converted to PASCAL VOC format which consist of four main directories: *JPEG Images* which contain the raw images files or the dataset, *SegmentationClass* contain *png* label for each of the image, the third directory *SegmentationClassRaw* contain raw labeled images and the *ImageSet* directory contain three txt files *train, val and trainval*, each of these files contain list of names of images names. The train has the largest portion of images, *train* contain images from which the model learns, *val* contain images that the model use to evaluate itself and infer model accuracy. The *val* set contain images that are not presented to the model during the training phase and unknown data for the model. Now the next step is model selection. Based on studying various models for the required task, the model that best suits this task was Xception 65. Then comes the fourth step or the training step in which the prepared data was fed into the model for learning iteratively, different convolution filters are applied for identifying classes. During the process of training neural network optimizes its parameter after every epoch to reduce the loss in output. The prediction of the model at the start of training is poor, however, it learns over every epoch and gets better, however, the training must be stopped after a certain epoch in order to avoid bias. In the evaluation step, the model predicts the output on unknown data, if the output probability value is high this means the model has good accuracy.

4.5 Implementation

The implementation of the project work particularly training the model, then evaluation and finally visualization of the model is discussed in this section. For this thesis method of transfer learning was used for model building.

4.5.1 Xception 65 Retraining and Evaluation

For retraining the Xception 65 which has been pre-trained on PASCAL VOC 2012 dataset. First, we needed to install TensorFlow and clone the TensorFlow *models* GitHub directory on the machine. However for this thesis, Google Collab was used which comes with TensorFlow 2.0 pre-installed, but due to some issues of Deeplab with TensorFlow 2.0, we have used TensorFlow GPU 1.14. Figure 4.4 Show installation of Tensorflow and other required libraries on Google Collab along with cloning TensorFlow *models* GitHub directory.

```
[ ]  1 ! git clone https://github.com/tensorflow/models.git

[ ]  1 pip install tensorflow-gpu==1.14

[ ]  1 pip install tensorflow-plot==0.3.0

[ ]  1 pip install numpy==1.16.4
```

Figure 4.4 Installation of required libraries for model training

As discussed in the earlier section the Google Collab session is allotted to a user for 12hrs of duration after which the virtual machine data is lost. To solve this problem and make the data persistent we mounted google drive onto Google Collab, the TensorFlow *models* directory was also cloned on Google drive so that the data and changes don't vanish after the closing of the session.

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

Listing 1: Command for mounting Google Drive in Google Collab.

The dataset conversion step was discussed in Section 4.4. However, in order to make it work with the required model, the dataset was first converted into a *TF Record*

format. The Conversion script was available in Deeplab directory, simply running the required script converted the dataset into *TF Record* format. The whole dataset in the zip file was first uploaded to Google Drive and the extracted to the required location within Google Collab. In the next step, the working directories were set upped and directory *models* was added to the python environment, along with this the pre-trained checkpoint were copied locally into google drive. Figure 4.5 shows all of the above steps.

Figure 4.5 Setting up working directories

Now the training steps actually begins here, after changing the current directory to *model/research/deeplab* , *train.py* python script located in the said directory was executed with changed parameters. Figure 4.6 shows the detail code for training.

Figure 4.6 Training parameters of the model

In Figure 4.6 ${TF_INIT_CKPT} is the path where we have downloaded our pre-trained checkpoint, which is PASCAL VOC training checkpoint, $TRAIN_LOGDIR is directory path where we want to write our training checkpoints and events while the ${Udder} is directory path in which our dataset resides. The atrous rate and output strides

were discussed in earlier section. As the dataset was small, shallow fine-tuning was used, which means tuning only the last few layers, for this purpose we have disabled the fine-tune batch normalization. The numbers of iteration were set to 5000. The learning rate for Adam optimizer was set to 0.001 with decay factor 0.1 for every 2000 steps. During each iteration, random 10 images are selected and sent to the last layer and results are compared with the correct label, and then compute the loss. The model then goes back to previous layers where weights are adjusted. This way the model improves its accuracy. After every few iterations, the output log was saved to the TRAIN_LOGDIR. Figure 4.7 shows the training log and loss during training.

```
[ ] I0120 17:47:47.537102 139664426276736 learning.py:507] global step 4820: loss = 0.2525 (1.314 sec/step)
    INFO:tensorflow:global step 4830: loss = 0.1823 (1.317 sec/step)
    I0120 17:48:02.431957 139664426276736 learning.py:507] global step 4830: loss = 0.1823 (1.317 sec/step)
    INFO:tensorflow:global step 4840: loss = 0.1969 (1.345 sec/step)
    I0120 17:48:17.162289 139664426276736 learning.py:507] global step 4840: loss = 0.1969 (1.345 sec/step)
    INFO:tensorflow:global step 4850: loss = 0.2027 (1.279 sec/step)
    I0120 17:48:33.368736 139664426276736 learning.py:507] global step 4850: loss = 0.2027 (1.279 sec/step)
    INFO:tensorflow:global step 4860: loss = 0.1788 (1.356 sec/step)
    I0120 17:48:46.249900 139664426276736 learning.py:507] global step 4860: loss = 0.1788 (1.356 sec/step)
    INFO:tensorflow:global step 4870: loss = 0.2106 (1.376 sec/step)
    I0120 17:48:59.386815 139664426276736 learning.py:507] global step 4870: loss = 0.2106 (1.376 sec/step)
    INFO:tensorflow:global step 4880: loss = 0.1903 (1.446 sec/step)
    I0120 17:49:14.684263 139664426276736 learning.py:507] global step 4880: loss = 0.1903 (1.446 sec/step)
    INFO:tensorflow:global step 4890: loss = 0.1637 (1.437 sec/step)
    I0120 17:49:29.053168 139664426276736 learning.py:507] global step 4890: loss = 0.1637 (1.437 sec/step)
    INFO:tensorflow:global step 4900: loss = 0.1853 (1.471 sec/step)
    I0120 17:49:44.418255 139664426276736 learning.py:507] global step 4900: loss = 0.1853 (1.471 sec/step)
    INFO:tensorflow:global step 4910: loss = 0.1915 (1.242 sec/step)
    I0120 17:49:59.036815 139664426276736 learning.py:507] global step 4910: loss = 0.1915 (1.242 sec/step)
    INFO:tensorflow:global step 4920: loss = 0.1930 (1.644 sec/step)
    I0120 17:50:13.127133 139664426276736 learning.py:507] global step 4920: loss = 0.1930 (1.644 sec/step)
    INFO:tensorflow:global step 4930: loss = 0.1796 (1.268 sec/step)
    I0120 17:50:30.017533 139664426276736 learning.py:507] global step 4930: loss = 0.1796 (1.268 sec/step)
    INFO:tensorflow:global step 4940: loss = 0.1923 (1.296 sec/step)
    I0120 17:50:42.820100 139664426276736 learning.py:507] global step 4940: loss = 0.1923 (1.296 sec/step)
    INFO:tensorflow:Saving checkpoint to path /content/drive/My Drive/models/research/deeplab/datasets/Udder/exp/train_on_trainval_set/train/model.ckpt
```

Figure 4.7 Training related log

Once the training was completed the model checkpoints were saved to the TRAIN_LOGDIR.

The next step now was to evaluate our trained model. The Evaluation python script was also available in the DeepLab directory as eval.py. Figure 4.8 shows the running evaluation script along with the setting parameters.

```
%cd /content/drive/My Drive/models/research/deeplab
!python eval.py \
    --logtostderr \
    --eval_split="val" \
    --model_variant="xception_65" \
    --atrous_rates=6 \
    --atrous_rates=12 \
    --atrous_rates=18 \
    --output_stride=16 \
    --decoder_output_stride=4 \
    --eval_crop_size="4050,4050" \
    --checkpoint_dir="${TRAIN_LOGDIR}"  \
    --eval_logdir="${EVAL_LOGDIR}" \
    --dataset_dir="/content/drive/My Drive/models/research/deeplab/datasets/Udder/tfrecord" \
    --max_number_of_evaluations=1
```

Figure 4.8 Evaluating the trained model

After training the model it was necessary to check the performance of the model over unseen data, in order to know how our model will perform over real-world data. Evaluation of model checks the accuracy of the model over unseen data or data which is not provided during training. Figure 4.9 shows the accuracy of the model in the form of mean intersection over union (miou).

```
eval/miou_1.0_class_0[0.978286862]
eval/miou_1.0_class_1[0.787486494]
eval/miou_1.0_overall[0.882886589]
```

Figure 4.9 Output accuracy of the model

The evaluation shows the accuracy of each class and overall. As we had two classes background and udder, mean intersection over union for both classes were calculated separately. Class_0 shows background miou_1.0 of 0.97 while that of the udder was 0.78 and overall 0.88. In the final step, the model was visualized to see the result of the evaluation set and prediction made by the model. The result from visualization is shown in Figure 4.10.

```
!python vis.py \
```

Listing 2: Command to run visualization script.

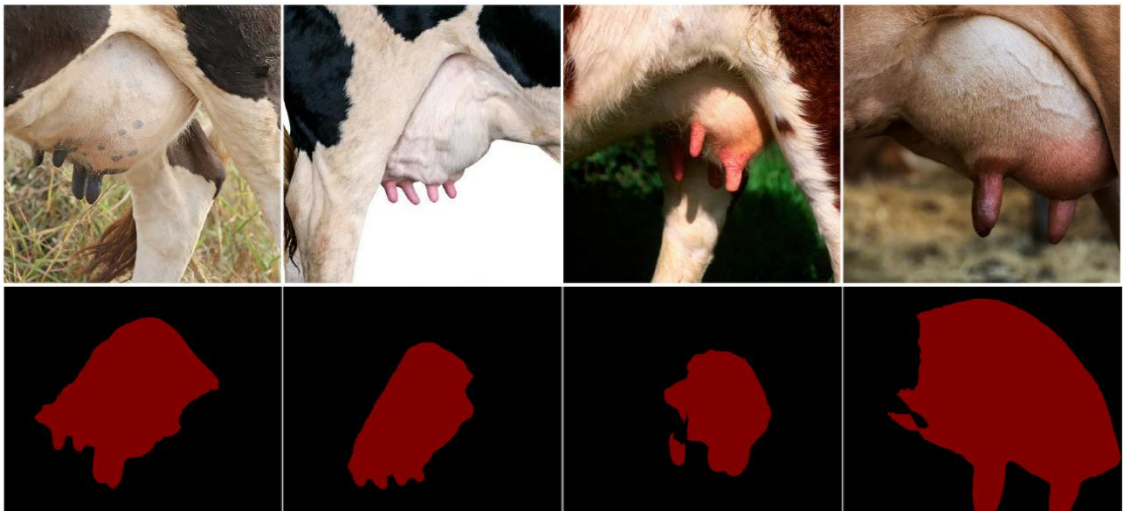

Figure 4.10 Visualization result of trained model over unseen data, images in upper row are from eval set while images in second row show prediction of model.

CHAPTER 5

5 RESULTS

This section discusses results in order to have an insight into how the model performed. The overall accuracy achieved by the model is given in Table 5.1.

Table 5.1 Output accuracy of the model

Class	Accuracy
Background	97%
Udder	78%
Overall	88%

Loss is one of the most important function in model performance. The loss actually interprets how well the model is performing. Loss is the summarization of error made for each iteration in training. The total training loss is shown in Figure 5.1 which shows how the loss decreased with the iterations starting very high to low (from 4.5 at first iteration to 0.2 at the end of the training). This number shows the model has converged well over the training data is expected to exhibit high accuracy. Reducing the loss is one of the main objectives in a learning model by changing the weights. The value of loss actually shows how good or bad the model is performing after each iteration. Generally, the loss reduces as the number of the iteration increases. The model is evaluated based on miou_1.0. MIoU refers to mean intersection over union which is the overlap area between the ground truth and predicted segmentation divided by area of union between them. Following is the equation for MIoU.

$$\text{MIoU} = \frac{(\text{Area of overlap})}{(\text{Area of union})}$$

The MIoU value ranges from 0-1 or 0-100%. 0 mean there is no overlap between the predicted and ground truth that is model is performing bad or is total garbage, while 1 signifies perfection or shows that model is perfectly segmenting the data. When the number of classes are two or more than two the MIoU is calculated by taking average of MIoU of each class.

Figure 5.1 Total Training loss function

Accuracy of the model can only be determined once the model has learned. Once training is complete, test data is fed into the model to calculate its accuracy, which is then compared to the target value in the form of a percentage. Figure 5.2 Show accuracy of class_1 while the accuracy of BACKGROUND is shown in Figure 5.3.

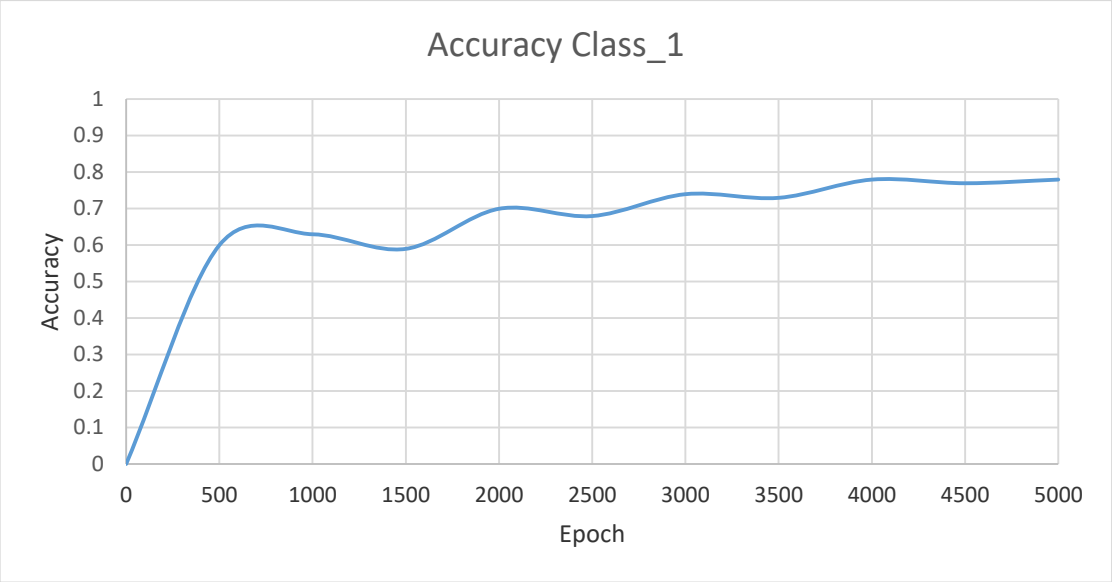

Figure 5.2 Accuracy of udder class

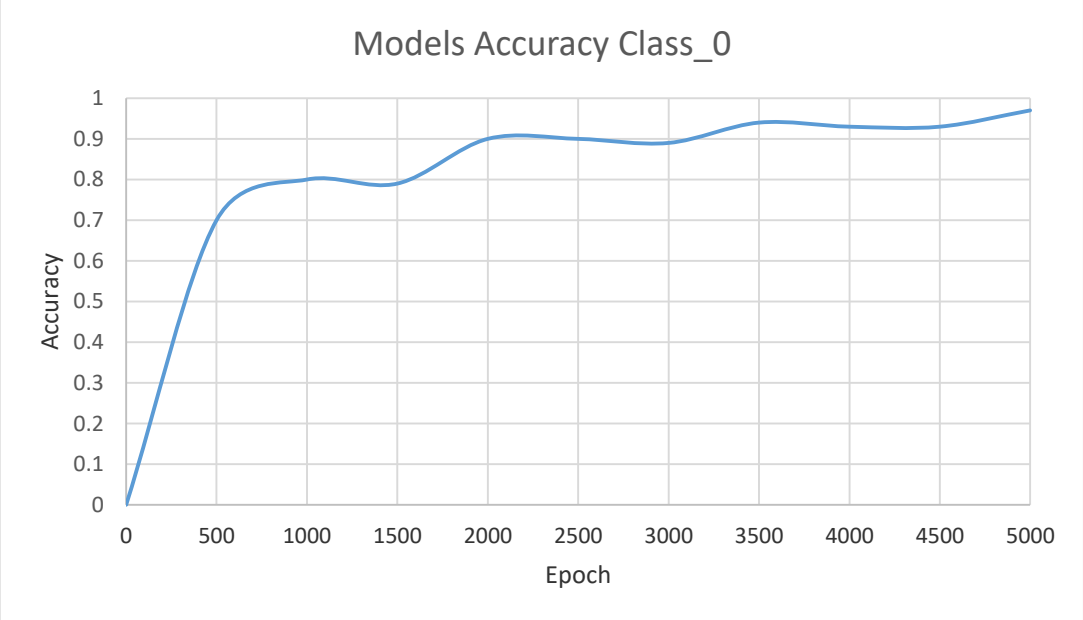

Figure 5.3 Accuracy of background Class

The overall accuracy of the model over both of the classes is shown in Figure 5.4.

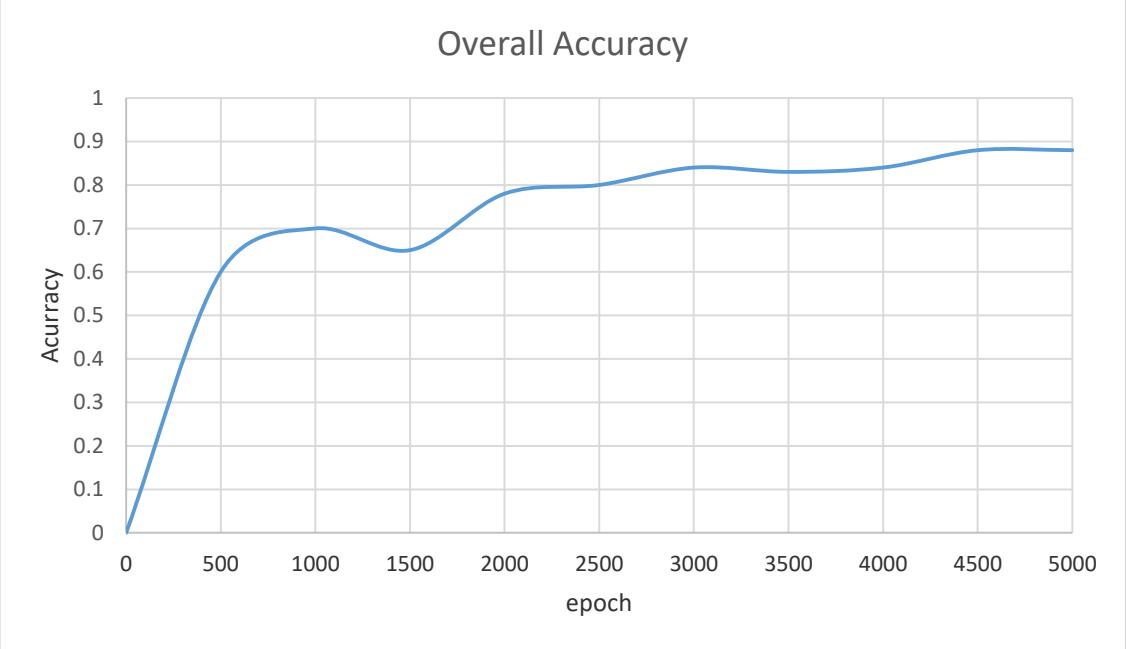

Figure 5.4 Accuracy of the model over 5000 iterations

Once the model procedure was completed and the desired result was obtained, the saved checkpoint created at the end of the training was used to export the frozen inference graph, which will be our network for udder detection.

CHAPTER 5

6 DISCUSSION

For this thesis several methods of auto image segmentation were studied, however, due to the complexity of the task it wasn't possible through simple image processing methods and a proper deep learning approach was needed. Before finalizing the framework, several architectures were studied and finally Deeplab v3+ system was used for the process of segmentation and detecting the udder. The Xception model of DeepLab reached an overall good miou_1.0 of 0.88 for the segmentation task by successfully segmenting udder in the image. The accuracy for the validation sample was encouraging given that the dataset was manually annotated. From the given result of the model, it can be concluded that it provides proof of automated udder detection using deep learning.

Data acquisition has been one of the most laborious and tedious parts of this research. All of the images were manually captured on different dairy farms in different operating conditions. The second most time-consuming part was the manual annotation of the data. Due to careful annotation and using appropriate learning model, such a small dataset has significantly contributed to generalize the results. The given model will further benefit if a larger dataset is used for classifying different classes of udders, individual teat, and other skin anomalies. In future an automated method for annotation of data can be used for speeding up the process of data generation. The system is dependent on DCNNs for the task of segmentation. As we have seen some major improvement in recent past and hopefully will continue to improve, which will directly be beneficial for segmentation tasks

6.1 Model Comparison

The pre-trained model's performance has been evaluated on accuracy and computational complexity in[55]. The performance of the identified models is shown in Figure 6.1.In order to have suitable model for the selected task, a model that have

optimum accuracy and computation complexity is selected.[48] Shows that Xception have better accuracy with slight higher computational complexity. Different models of CNN like VGGnet, requires a large amount of computational power for evaluation. While the Xception which is successor of inception model was made for functioning under computational limitations. The Xception design uses three times less parameters when compared to VGGnet. While other lighter models like MobileNet lack accuracy when compared to Xception. This justifies the use of Xception model, when data need to be processed at relatively low computational cost and better accuracy[55].

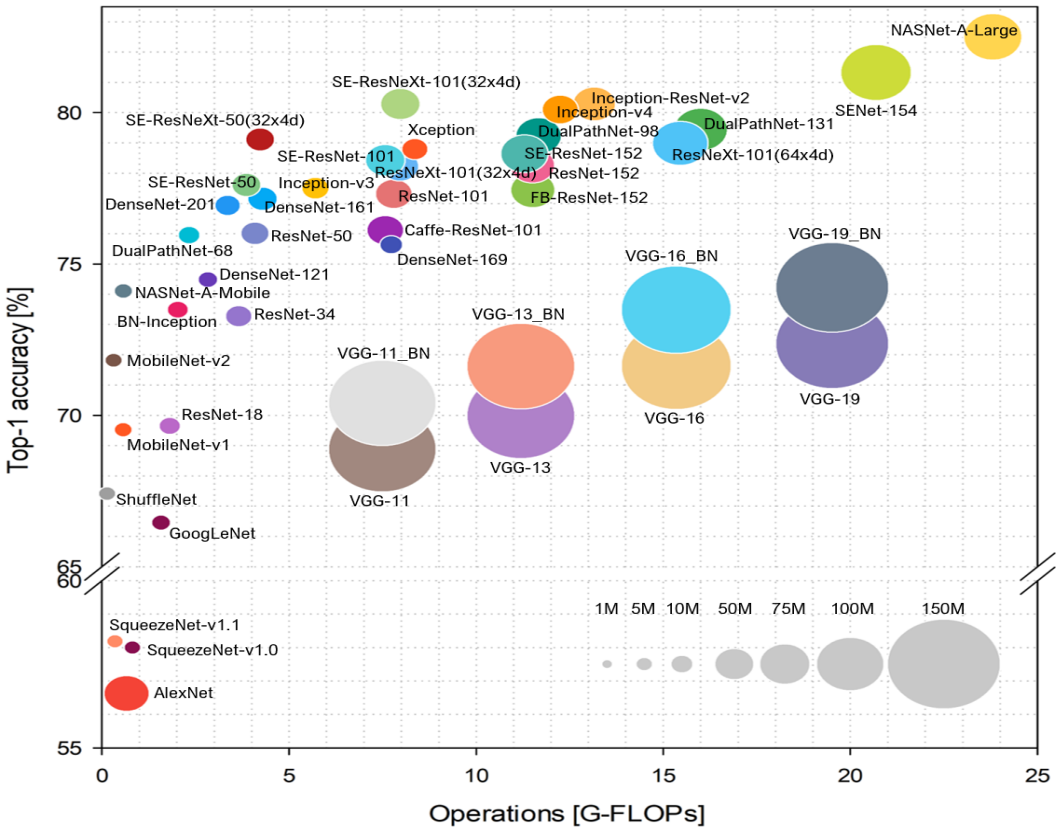

Figure 6.1 Accuracy vs computational power[55]

Varaich et.al [52] have made a comparison of Inception v3 and Xception on similar tasks with weight initialization, the result from which shows Xception is performing better in accuracy with similar model shown in figure.

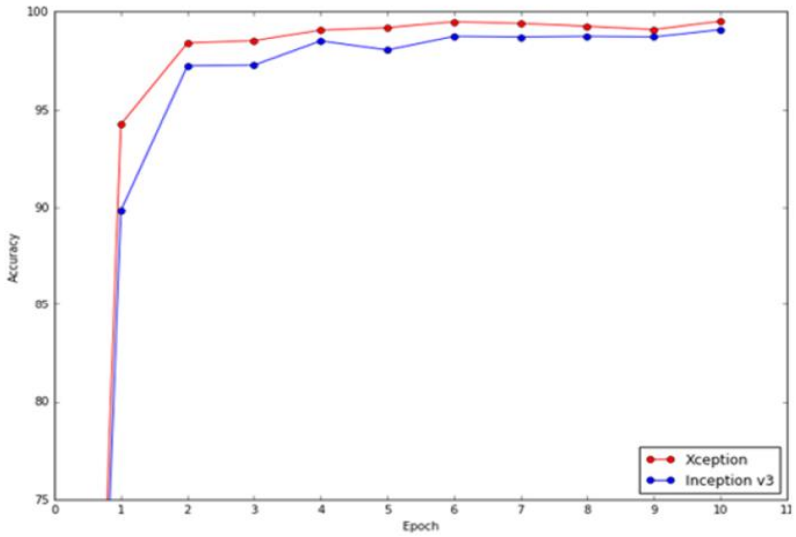

Figure 6.2 Accuracy comparison Xception and Inception[52]

Moreover in Kaggle leaderboard scores and raking also shows that Xception outperforms Inception v3 ResNet and VGG in term of accuracy which have almost the same computational complexities.

	Parameter count	Steps/second
Inception V3	23,626,728	31
Xception	22,855,952	28

	Top-1 accuracy	Top-5 accuracy
VGG-16	0.715	0.901
ResNet-152	0.770	0.933
Inception V3	0.782	0.941
Xception	**0.790**	**0.945**

Figure 6.3 Performance Comparison of models steps/sec and accuracy

6.2 Conclusion

The main focus of the research was to determine whether the problem of udder detection can be solved through deep learning or not. The model was successful in narrowing down the region of interest from the image i.e udder using deep learning technique. The outcomes of this research clearly demonstrate the feasibility of using a transfer learning model for accurate detection of cow udder in the dairy farm environment with an acceptable accuracy. With 88% accuracy the model detected udder successfully. There definitely is room for improvement, a larger dataset can lead to better training results. Without a doubt, the results obtained are almost close to human vision and can be used in an autonomous milking system for udder detection. Furthermore the dataset made can be used in other dairy farm related problems.

6.3 Future Work

Automated udder detection is one of the primary constraints in smart dairy farming. The proposed research brings up the solution for detecting cow udder in ambient environments without adding any specialized lighting or any wearable sensors. Accuracy of around 90% suffices most of the skin related researches yet it could be further improved for subpixel accuracy applications for robotic application. This can be achieved by adding more training data.

Fixed small-sized images can be used for speeding up the processing. Few other models can be implemented with adjustment of various parameters, like learning rate and number of layers. This can lead to picking better models easily but will also require more time. If the dataset is updated with more classes of date for a healthy udder, infected udder and non-milk-able teats then this can truly become a smart system for automatic milking in dairy farms.

References

1. Arel, I., D.C. Rose, and T.P. Karnowski, *Deep machine learning-a new frontier in artificial intelligence research.* IEEE computational intelligence magazine, 2010. **5**(4): p. 13-18.
2. Selvaraj, D. and R. Dhanasekaran, *A review on tissue segmentation and feature extraction of MRI brain images.* International Journal of Computer Science and Engineering Technology (IJCSET), 2013. **4**(10): p. 1313-1332.
3. Bengio, Y., A. Courville, and P. Vincent, *Representation learning: A review and new perspectives.* IEEE transactions on pattern analysis and machine intelligence, 2013. **35**(8): p. 1798-1828.
4. Mitchell, R., J. Michalski, and T. Carbonell, *An artificial intelligence approach*. 2013: Springer.
5. Cheng, B. and D.M. Titterington, *Neural networks: A review from a statistical perspective.* Statistical science, 1994: p. 2-30.
6. Gavrila, D.M., *The visual analysis of human movement: A survey.* Computer vision and image understanding, 1999. **73**(1): p. 82-98.
7. Notsuki, I. and K. Ueno, *System for managing milking-cows in stanchion stool*. 1977, Google Patents.
8. Akerman, D., *Verfahren und Vorrichtung zum Melken.* Dt. Offenlegungsschr., 1979. **28**(49): p. 227.
9. Andersson, L. and M. Nilsson, *Apparatus and method for recognizing and determining the position of a part of an animal*. 2001, Google Patents.
10. LeCun, Y., Y. Bengio, and G. Hinton, *Deep learning. nature 521*. 2015.
11. Goodfellow, I. and Y. Bengio, *Aaron Courville Deep Learning*. 2016, MIT press Cambridge, MA.
12. Kadam, D.B., *Neural network based brain tumor detection using MR images.* 2012.
13. Joshi, D.M., N. Rana, and V. Misra. *Classification of brain cancer using artificial neural network*. in *2010 2nd International Conference on Electronic Computer Technology*. 2010. IEEE.
14. Kon, M.A. and L. Plaskota, *Information complexity of neural networks.* Neural Networks, 2000. **13**(3): p. 365-375.
15. Khan, A., et al., *A review of machine learning algorithms for text-documents classification.* Journal of advances in information technology, 2010. **1**(1): p. 4-20.
16. Kotsiantis, S.B., I. Zaharakis, and P. Pintelas, *Supervised machine learning: A review of classification techniques.* Emerging artificial intelligence applications in computer engineering, 2007. **160**: p. 3-24.
17. LeCun, Y., et al. *Handwritten digit recognition with a back-propagation network*. in *Advances in neural information processing systems*. 1990.
18. Litjens, G., et al., *A survey on deep learning in medical image analysis.* Medical image analysis, 2017. **42**: p. 60-88.
19. Vedaldi, A. and K. Lenc. *Matconvnet: Convolutional neural networks for matlab*. in *Proceedings of the 23rd ACM international conference on Multimedia*. 2015. ACM.
20. LeCun, Y., Y. Bengio, and G. Hinton, *Deep learning.* nature, 2015. **521**(7553): p. 436.

21. Li, Y., S. Hara, and K. Shimura. *A machine learning approach for locating boundaries of liver tumors in ct images*. in *18th International Conference on Pattern Recognition (ICPR'06)*. 2006. IEEE.
22. Hinton, G.E., S. Osindero, and Y.-W. Teh, *A fast learning algorithm for deep belief nets.* Neural computation, 2006. **18**(7): p. 1527-1554.
23. Bengio, Y., *Learning deep architectures for AI.* Foundations and trends® in Machine Learning, 2009. **2**(1): p. 1-127.
24. Nagalkar, V. and S. Asole, *Brain tumor detection using digital image processing based on soft computing.* Journal of signal and image processing, 2012. **3**(3): p. 102-105.
25. Lawrence, S., et al., *Face recognition: A convolutional neural-network approach.* IEEE transactions on neural networks, 1997. **8**(1): p. 98-113.
26. LeCun, Y., *Yoshua Bengio, and Geoffrey Hinton.* Deep learning. Nature, 2015. **521**(7553): p. 436-444.
27. Duin, R.P.W., *On the choice of smoothing parameters for Parzen estimators of probability density functions.* IEEE Transactions on Computers, 1976(11): p. 1175-1179.
28. LeCun, Y., et al., *Gradient-based learning applied to document recognition.* Proceedings of the IEEE, 1998. **86**(11): p. 2278-2324.
29. Le Cun, Y., et al., *Handwritten digit recognition: applications of neural net chips and automatic learning*, in *Neurocomputing*. 1990, Springer. p. 303-318.
30. Ponce, J., et al., *Dataset issues in object recognition*, in *Toward category-level object recognition*. 2006, Springer. p. 29-48.
31. Thomas, A., et al. *Towards multi-view object class detection*. in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. 2006. IEEE.
32. Deng, J., et al. *Imagenet: A large-scale hierarchical image database*. in *2009 IEEE conference on computer vision and pattern recognition*. 2009. Ieee.
33. Hu, F., et al., *Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery.* Remote Sensing, 2015. **7**(11): p. 14680-14707.
34. Katamreddy, S., et al. *Visual Udder Detection with Deep Neural Networks*. in *2018 12th International Conference on Sensing Technology (ICST)*. 2018. IEEE.
35. Xinyue, Z., et al. *Inversion of Heavy Metal Content in a Copper Mining Area Based on Extreme Learning Machine Optimized by Particle Swarm Algorithm*. in *2018 10th IAPR Workshop on Pattern Recognition in Remote Sensing (PRRS)*. 2018. IEEE.
36. Rathi, D., S. Jain, and S. Indu. *Underwater fish species classification using convolutional neural network and deep learning*. in *2017 Ninth International Conference on Advances in Pattern Recognition (ICAPR)*. 2017. IEEE.
37. Sargano, A.B., et al. *Human action recognition using transfer learning with deep representations*. in *2017 International joint conference on neural networks (IJCNN)*. 2017. IEEE.
38. Shen, D., et al. *Flame detection using deep learning*. in *2018 4th International Conference on Control, Automation and Robotics (ICCAR)*. 2018. IEEE.
39. Warner, D., et al., *A machine learning based decision aid for lameness in dairy herds using farm-based records.* Computers and Electronics in Agriculture, 2020. **169**: p. 105193.

40. Yang, A., et al., *An automatic recognition framework for sow daily behaviours based on motion and image analyses.* Biosystems Engineering, 2020. **192**: p. 56-71.
41. Chen, L.-C., et al., *Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs.* IEEE transactions on pattern analysis and machine intelligence, 2017. **40**(4): p. 834-848.
42. Chen, L.-C., et al., *Rethinking atrous convolution for semantic image segmentation.* arXiv preprint arXiv:1706.05587, 2017.
43. Chen, L.-C., et al. *Encoder-decoder with atrous separable convolution for semantic image segmentation*. in *Proceedings of the European conference on computer vision (ECCV)*. 2018.
44. Chen, L.-C., et al., *Semantic image segmentation with deep convolutional nets and fully connected crfs.* arXiv preprint arXiv:1412.7062, 2014.
45. Krähenbühl, P. and V. Koltun. *Efficient inference in fully connected crfs with gaussian edge potentials*. in *Advances in neural information processing systems*. 2011.
46. Yu, F. and V. Koltun, *Multi-scale context aggregation by dilated convolutions.* arXiv preprint arXiv:1511.07122, 2015.
47. He, K., et al., *Spatial pyramid pooling in deep convolutional networks for visual recognition.* IEEE transactions on pattern analysis and machine intelligence, 2015. **37**(9): p. 1904-1916.
48. Chollet, F. *Xception: Deep learning with depthwise separable convolutions*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.
49. Szegedy, C., et al. *Going deeper with convolutions*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.
50. Sandler, M., et al. *Mobilenetv2: Inverted residuals and linear bottlenecks*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.
51. Szegedy, C., et al. *Rethinking the inception architecture for computer vision*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
52. Varaich, Z.A. and S. Khalid. *Recognizing Actions of Distracted Drivers using Inception v3 and Xception Convolutional Neural Networks*. in *2019 2nd International Conference on Advancements in Computational Sciences (ICACS)*. 2019. IEEE.
53. He, K., et al. *Deep residual learning for image recognition*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
54. Wada, K., *labelme: Image Polygonal Annotation with Python*. 2016.
55. Bianco, S., et al., *Benchmark analysis of representative deep neural network architectures.* IEEE Access, 2018. **6**: p. 64270-64277.

Appendices

Project Code

```
#Mounting Google Drive into Google Collab

from google.colab import drive
drive.mount('/content/drive')
#Installing Required Libraries
pip install tensorflow-gpu==1.14
pip install tensorflow-plot==0.3.0
pip install numpy==1.16.4

#Unzip Dataset to the required path
!unzip -q "/content/drive/My Drive/Udder.zip" -d "/content/drive/My
Drive/models/research/deeplab/datasets/Udder"

#Add Deeplab to Python Enivroment Path
%cd /content/drive/My Drive/models/research/deeplab
%env PYTHONPATH=/env/python/:/content/drive/My
Drive/models/research/:/content/drive/My Drive/models/research//slim
!export PYTHONPATH=$PYTHONPATH:`pwd`:`pwd`/slim

#Working Directories Setup
%env Udder=/content/drive/My Drive/models/research/deeplab/datasets/Udder/tfrecord
%env CKPT_PATH=/content/drive/My
Drive/models/research/deeplab/datasets/Udder/exp/train_on_trainval_set/train/model.ckpt-
20000
%env EXPORT_PATH=/content/drive/My
Drive/models/research/deeplab/datasets/Udder/exp/train_on_trainval_set/export/frozen_infer
ence_graph.pb

# Set up the working directories.
%env INIT_FOLDER=/content/drive/My
Drive/models/research/deeplab/datasets/Udder/exp/train_on_trainval_set/init_models
%env TRAIN_LOGDIR=/content/drive/My
Drive/models/research/deeplab/datasets/Udder/exp/train_on_trainval_set/train
%env VIS_LOGDIR=/content/drive/My
Drive/models/research/deeplab/datasets/Udder/exp/train_on_trainval_set/vis
%env EVAL_LOGDIR=/content/drive/My
Drive/models/research/deeplab/datasets/Udder/exp/train_on_trainval_set/eval
%env EXPORT_DIR=/content/drive/My
Drive/models/research/deeplab/datasets/Udder/exp/train_on_trainval_set/export
%env TF_INIT_ROOT=http://download.tensorflow.org/models

# Copy locally the trained checkpoint as the initial checkpoint.
%cd '/content/drive/My
Drive/models/research/deeplab/datasets/Udder/exp/train_on_trainval_set/init_models'
%env TF_INIT_CKPT=deeplabv3_pascal_train_aug_2018_01_04.tar.gz
!wget -nd -c "${TF_INIT_ROOT}/${TF_INIT_CKPT}"
!tar -xf "/content/drive/My
Drive/models/research/deeplab/datasets/Udder/exp/train_on_trainval_set/init_models/deeplab
v3_pascal_train_aug_2018_01_04.tar.gz"
```

```
#Start Tensorboard
%load_ext tensorboard
import tensorflow as tf
import datetime, os

%tensorboard --logdir "/content/drive/My
Drive/models/research/deeplab/datasets/Udder/exp/train_on_trainval_set/"

#Start Training model
%cd '/content/drive/My Drive/models/research/deeplab'
!python train.py \
  --logtostderr \
  --train_split="train" \
  --model_variant="xception_65" \
  --atrous_rates=6 \
  --atrous_rates=12 \
  --atrous_rates=18 \
  --output_stride=16 \
  --decoder_output_stride=4 \
  --train_crop_size=513,513 \
  --train_batch_size=4 \
  --training_number_of_steps=5000 \
  --fine_tune_batch_norm=false \
  -tf_initial_checkpoint="${TF_INIT_CKPT}" \
  --train_logdir="${TRAIN_LOGDIR}" \
  --dataset_dir="${Udder}"

#Model Evaluation
%cd /content/drive/My Drive/models/research/deeplab
!python eval.py \
    --logtostderr \
    --eval_split="val" \
    --model_variant="xception_65" \
    --atrous_rates=6 \
    --atrous_rates=12 \
    --atrous_rates=18 \
    --output_stride=16 \
    --decoder_output_stride=4 \
    --eval_crop_size="4050,4050" \
    --checkpoint_dir="${TRAIN_LOGDIR}"  \
    --eval_logdir="${EVAL_LOGDIR}" \
    --dataset_dir="/content/drive/My Drive/models/research/deeplab/datasets/Udder/tfrecord" \
    --max_number_of_evaluations=1

#Model Visualization
!python vis.py \
--logtostderr \
--vis_split="val" \
--model_variant="xception_65" \
--atrous_rates=6 \
```

```
--atrous_rates=12 \
--atrous_rates=18 \
--output_stride=16 \
--decoder_output_stride=4 \
--vis_crop_size="4050,4050" \
--checkpoint_dir="${TRAIN_LOGDIR}" \
--vis_logdir="${VIS_LOGDIR}" \
--dataset_dir="/content/drive/My Drive/models/research/deeplab/datasets/Udder/tfrecord" \
--max_number_of_iterations=1

#Hardware Detail
!lscpu |grep 'Model name'
!lscpu | grep "MHz"
!df -h / | awk '{print $4}'
!cat /proc/meminfo | grep 'MemAvailable'
!nvidia-smi -L
!nvidia-smi
```