

**OBJECT BASED SEGMENTATION AND ANALYSIS
USING DEEP LEARNING ALGORITHM
FOR CATS AND DOGS IMAGES
BY
SHREEVISHAL A/L N.RAJANDHIRAN**

A REPORT
SUBMITTED TO
Universiti Tunku Abdul Rahman
in partial fulfillment of the requirements
for the degree of
BACHELOR OF COMPUTER SCIENCE (HONOURS)
Faculty of Information and Communication Technology
(Kampar Campus)

JANUARY 2023

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: OBJECT BASED SEGMENTATION AND ANALYSIS USING DEEP LEARNING ALGORITHM FOR CATS AND DOGS IMAGES

Academic Session: JAN 2023

I SHREEVISHAL A/L N.RAJANDHIRAN
(CAPITAL LETTER)

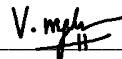
declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.



(Author's signature)

Verified by,



(Supervisor's signature)

Address:

L4, Jalan Cenderuh 4,
Taman Bamboo,
51200,
Kuala Lumpur

Mogana Vadiveloo
Supervisor's name

Date: 28th of April 2023

Date: 28/04/2023

Universiti Tunku Abdul Rahman			
Form Title : Sample of Submission Sheet for FYP/Dissertation/Thesis			
Form Number: FM-IAD-004	Rev No.: 0	Effective Date: 21 JUNE 2011	Page No.: 1 of 1

FACULTY/INSTITUTE* OF INFORMATION AND COMMUNICATION TECHNOLOGY
UNIVERSITI TUNKU ABDUL RAHMAN

Date: 28th of April 2023

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that Shreevishal A/l N.Rajandhiran (ID No: 19ACB04290) has completed this final year project/ dissertation/ thesis* entitled “Object Based Segmentation and Analysis Using Deep Learning Algorithm for Cats and Dogs Images” under the supervision of Dr. Mogana Vadiveloo (Supervisor) from the Department of Computer Science , Faculty/Institute* of Faculty of Information and Communication Technology

I understand that University will upload softcopy of my final year project / dissertation/ thesis* in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,



(Shreevishal A/L N.Rajandhiran)

DECLARATION OF ORIGINALITY

I declare that this report entitled “OBJECT BASED SEGMENTATION AND ANALYSIS USING DEEP LEARNING ALGORITHM FOR CATS AND DOGS IMAGES” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature :



Name : Shreevishal A/L N.Rajandhiran

Date : 28th of April 2023

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Mogana Vadiveloo for giving me this opportunity to partake in the execution of this FYP title. It is a new field of research for me, and I have had an enjoyable experience learning new algorithms and techniques in realising image segmentation on natural images especially within the domain of images for cats and dogs. I am truly grateful for the skillset that I have gained in executing this final year project and am looking forward to applying these skills elsewhere.

ABSTRACT

Image segmentation is the process of segmenting an image into the region of interest. These regions of interest (ROI) can be objects that are found in an image. Natural images consist of objects such as dogs, cats, and etc that can be segmented as region of interest. There are existing software such as Image Annotation Lab, ImageJ and Interactive Segmentation Tool that are used to delineate the ROI in natural images however, these software have common limitation. User intervention is constantly required to perform the segmentation. In addition, the algorithms used for the segmentation are conventional rather than modern deep learning techniques which is inevitably more efficient. Thus, this proposed work uses deep learning techniques to segment the region of interest in natural images, therefore reducing human intervention in performing the segmentation. The deep learning algorithm implemented is the Unet architecture and the Segnet architecture respectively. The dataset representing the narrative of natural images is the Oxford IIIT Pet Dataset which contains 7500 images of dogs and cats of different breeds with the respective ground truth images. The evaluation measurement to determine the performance of both implemented Unet architecture and Segnet architecture is the validation loss and the accuracy. The Unet model built from scratch has observed an accuracy of 89.22% , while the pretrained Unet model has observed an accuracy of 89.68% and the Segnet model built from scratch observed an accuracy of 84.87% in delineating the cats and dogs as ROIs in the natural images dataset.

TABLE OF CONTENTS

TITLE PAGE	i
REPORT STATUS DECLARATION FORM	ii
FYP THESIS SUBMISSION FORM	iii
DECLARATION OF ORIGINALITY	iv
ACKNOWLEDGEMENTS	v
ABSTRACT	vi
TABLE OF CONTENTS	vii
LIST OF FIGURES	x
LIST OF TABLES	xiii
LIST OF ABBREVIATIONS	xiv
CHAPTER 1 INTRODUCTION	15
1.1 Problem Statement and Motivation	16
1.2 Objectives	16
1.3 Project Scope and Direction	17
1.4 Contributions	17
1.5 Report Organization	17
CHAPTER 2 LITERATURE REVIEW	18
2.1 Objectives of Literature Review	18
2.2 Image Annotation Lab	18
2.3 Interactive Segmentation Tool	22
2.4 ImageJ	27
2.5 Understanding Common Algorithms for Image Segmentation	33
2.6 Critical Remark and Limitations of Tools	36
2.7 Proposed Solutions	38
CHAPTER 3 SYSTEM METHODOLOGY/APPROACH (FOR DEVELOPMENT-BASED PROJECT)	39
3.1 Methodology	39

3.2 Tools used	40
3.3 User Requirement	41
3.4 System Performance Definition	42
3.5 Dataset	42
3.6 Model Architecture	43
3.7 Timeline	45
3.8 Chapter Summary	48
 CHAPTER 4 SYSTEM DESIGN	 49
4.1 Model Creation Flowchart	49
4.2 Overall Flowchart of Proposed System	50
4.3 Activity Diagram	51
4.4 Verification Plan (Deep Learning Model)	53
4.5 Verification Plan (Web Application)	56
 CHAPTER 5 SYSTEM IMPLEMENTATION (FOR DEVELOPMENT-BASED PROJECT)	 59
5.1 Start	59
5.2 Data Pre-processing	59
5.3 Data Loading	60
5.4 Data Splitting	60
5.5 Data Augmentation	60
5.6 Hyperparameter set up	61
5.7 Unet Model from scratch	62
5.8 Pretrained Unet Model Implementation	63
5.9 Segnet Model from scratch	64
5.10 Model Training and Validation	65
5.11 Deploying built models as a web application using Flask framework	67
5.12 Cats and Dogs Image Segmentation produced from the web application	68
5.13 Implementation Issues and Challenges	71

CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION	73
6.1 Testing with Experimental Models and Hyperparameters	73
6.2 Testing with Testing Set for each model	75
6.3 Testing the existence of False Positives	77
CHAPTER 7 CONCLUSION AND RECOMMENDATION	82
7.1 Conclusion	82
7.2 Recommendation	82
REFERENCES	83
APPENDIX	85
WEEKLY LOG	130
POSTER	136
PLAGIARISM CHECK RESULT	137
FYP2 CHECKLIST	143

LIST OF FIGURES

Figure Number	Title	Page
Figure 1.1	Differences in image segmentation	15
Figure 2.1	Image Annotation Lab User Interface	19
Figure 2.2	Segmentation Result on a Mushroom (Image)	20
Figure 2.3	User Interface of Interactive Segmentation Tool	22
Figure 2.4	Input image of an eagle sitting on a branch (foreground and background labelled markers)	24
Figure 2.5	Segmentation Result of Eagle as Region of Interest	24
Figure 2.6	Input image of a painting on a background (foreground and background labelled markers)	25
Figure 2.7	Result of Segmentation for painting as Region of Interest	25
Figure 2.8	ImageJ User Interface	27
Figure 2.9	Selecting a boundary from the entire input image	29
Figure 2.10	ImageJ identifying the cells via thresholding	29
Figure 2.11	Results of each respective cell in the selected boundary	30
Figure 2.12	Results exported to Excel	31
Figure 2.13	(a) Traditional Computer Vision workflow vs. (b) Deep Learning workflow	34
Figure 2.14	Experimental Analysis of Various Segmentation Methods	35
Figure 2.15	Result Summary for Algorithm Efficiencies	36
Figure 3.1	Input Image with it's True Mask	43
Figure 3.2	Visualisation of the Unet architecture	43
Figure 3.3	Segnet architecture	44
Figure 3.4	Gantt Chart of FYP1 Project Timeline	46
Figure 3.5	Activity done during FYP1	46
Figure 3.6	Gantt Chart of FYP2 Project Timeline	47
Figure 3.7	Activity done during FYP2	47
Figure 4.1	Model Creation Flowchart	49
Figure 4.2	Overall flowchart of the proposed system	50
Figure 4.3	Activity Diagram of the proposed system	51

Figure 5.1	Importing the necessary libraries	59
Figure 5.2	Function to normalize true mask	59
Figure 5.3	Loading the dataset from tensorflow	60
Figure 5.4	Data splitting according to pre-determined ratio	60
Figure 5.5	Initialization of data augmentation	61
Figure 5.6	Hyperparameter set up for model training	61
Figure 5.7	Creating the convolutional block (downsample and upsample blocks)	62
Figure 5.8	Finalizing the dimensions of the blocks	63
Figure 5.9	Implementation of the pretrained encoder (MobileNetV2)	63
Figure 5.10	Importing the convolutional layers that exists within Keras libraries	64
Figure 5.11	Defining the Segnet model using the convolutional layers from Keras	64
Figure 5.12	Training and Validation Loss (Unet model from scratch)	65
Figure 5.13	Training and Validation Loss (pretrained Unet)	66
Figure 5.14	Training and Validation Loss (Segnet model from scratch)	66
Figure 5.15	Default app route using Flask	67
Figure 5.16	Loading in the model weight	68
Figure 5.17	Function implementation to segment user image	68
Figure 5.18	Homepage of the web application hosted on localhost IP address	68
Figure 5.19	User chooses and uploads an image of a cat for segmentation	69
Figure 5.20	Display the input image for user verification	70
Figure 5.21	Segnet model output for the input image and its accuracy and loss values	71
Figure 5.22	Unet model output for the input image and its accuracy and loss values	72
Figure 6.1	Segmentation Result of Natural Oxford IIT Pet Dataset (Unet from scratch)	76
Figure 6.2	Segmentation Result of Natural Oxford IIT Pet Dataset (pretrained Unet)	76

Figure 6.3	Segmentation Result of Natural Oxford IIT Pet Dataset (Segnet from scratch)	77
Figure 6.4	Distorted Segmentation Result (Unet from scratch)	77
Figure 6.5	Distorted Segmentation Result (pretrained Unet)	77
Figure 6.6	Distorted Segmentation Result (Segnet from scratch)	77
Figure 6.7	Rabbit Input Image	78
Figure 6.8	Segmented Rabbit Image from Segnet	78
Figure 6.9	Segmented Rabbit Image from Unet	79
Figure 6.10	Deer Input Image	79
Figure 6.11	Segmented Deer Image from Segnet	80
Figure 6.12	Segmented Deer Image from Unet	80

LIST OF TABLES

Table Number	Title	Page
Table 2.1	Comparison of ImageJ, Interactive Segmentation Tool and Image Annotation Lab	37
Table 3.1	Laptop Specifications	40
Table 4.1	Verification Plan (Epoch Value)	53
Table 4.2	Verification Plan (Optimizers)	54
Table 4.3	Verification Plan (Loss Function)	55
Table 4.4	Verification Plan (Input Image)	56
Table 4.5	Verification Plan (Segmented Image Production)	57
Table 4.6	Verification Plan (Google Colab Connectivity)	58
Table 6.1	Summary Experimental Models (Model)	73
Table 6.2	Summary Experimental Models (Epoch Value and Model Type)	73
Table 6.3	Summary Experimental Models (Optimizers)	74
Table 6.4	Summary Experimental Models (Loss Function)	75

LIST OF ABBREVIATIONS

<i>ROI</i>	Region of Interest
<i>AI</i>	Artificial Intelligence
<i>GUI</i>	Graphical User Interface
<i>CV</i>	Computer Vision
<i>DL</i>	Deep Learning
<i>SCE</i>	Sparse Categorical Crossentropy

Chapter 1

Introduction

Image segmentation can be perceived as a process that breaks down a digital image into subgroups known as image segments. In easy words, segmentation is basically assigning labels to each individual pixel and all the pixels belonging to the same category have a common label assigned to them [1]. These image segments are also known as regions of interest (ROI). ROIs can be determined based on the classification of their pixels with a certain property such as colour, spatial contextual and texture. This object-based segmentation to obtain ROIs is done to encourage analysis of the ROI and the classification of the ROI aligned with the human perceptual system. There is no rule or way as to how these visual objects can be segmented as there are many algorithms and tools available to users that intend to segment objects based on their own requirements and needs. Some of the known tools would be the Interactive Segmentation Tool, ImageJ and Image Annotation Lab. Each of these tools are also subjected to the usage of image segmentation algorithms such as region-based, clustering-based, edge-based and thresholding [1]. Each of these methods take different approaches in segmenting objects in natural images and it all depends on the requirements of the users. It is also worth noticing that these methods may have an accurate segmentation result for one input natural image but not necessarily for other input images. Below is Figure 1.1 that shows the differences in types of natural image segmentation.

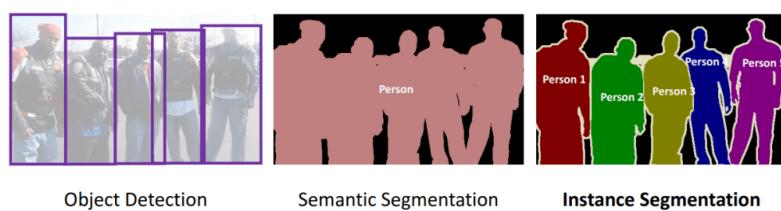


Figure 1.1 Differences in natural image segmentation

1.1 Problem Statement and Motivation

As mentioned earlier, users are given the privilege to use various types of tools to execute natural image segmentation and the ones mentioned previously were ImageJ, Image Annotation Lab and Interactive Segmentation Tool [1,2,3]. There is no doubt that this software can indeed execute the natural image segmentation process adequately however they use semi-automatic image segmentation methods when segmenting the ROIs in a given input image. The methods are perceived as semi-automatic and not entirely automatic because it still requires user input such as manually drawing the boundary and annotating the marker on objects that need to be segmented as ROIs. This can bring about inconveniences when users are trying to segment complex natural images and there is no doubt that it can be time consuming as well. The aim of the work is to propose an application that can automatically segment region of interests from given natural image of dogs and cats using deep learning techniques. Human interaction using software requires human interaction to a certain extent in order for the image segmentation to occur and the goal is to reduce this human interaction whilst prioritizing accuracy in segmenting the region of interest in natural images particularly in the domain of images with cats and dogs.

1.2 Objectives

The main objective of this project is to develop an application that segments visual object(s) of cats and dogs into region of interests (ROI) in natural images. The main objective is achieved by first achieving the sub-objective as stated below:

1. To implement a model using deep learning algorithm to segment the object regions in images of cats and dogs.
2. To automate the process of the object region delineation in (1).
3. To deploy the proposed model as a web application by integrating the flask framework to show the segmented objects regions results for further analysis.

1.3 Project Scope and Direction

The scopes of the proposed system are as follows:

1. The project will be focusing on segmenting cats and dogs as region of interests (ROI) in natural images.
2. The rules derived during the region identification of segmenting the object is based on the expert's opinion on the datasets or a common understanding established on the dataset.
3. The dataset used in the project will be natural images in either PNG or JPG formats.

1.4 Contributions

The main impact and contribution for this project is to develop an application that can perform an accurate segmentation of the region of interest from the cats and dogs images. This is achieved via the sub-contributions as stated below:

1. An application that is able to segment cats and dogs as the region of interest in natural images using deep-learning algorithm.
2. A web application that automatically performs object-based image segmentation with minimal user input

1.5 Report Organization

The details of this research are shown in the following chapters. Chapter 1 described the overall intent in pursuing this project. While, in Chapter 2, some related software that is designed for natural image segmentation tasks are reviewed alongside research performed to compare the efficiencies of deep learning image segmentation algorithms in comparison to conventional image segmentation techniques. Moreover, Chapter 3, discusses the system methodology of this project while Chapter 4 describes the system design and illustrates some diagrams for the execution of this project. Chapter 5 and Chapter 6 cover the implementation and testing aspects of the project and Chapter 7 then concludes the overall project.

Chapter 2

Literature Review

2.1 Objective of Literature Review

The main objective of this literature review is to understand the functionalities of existing tools used in image segmentation and to compare their differences as well as get a deeper understanding towards research done on the differences in efficiencies of deep learning algorithms in comparison to traditional image segmentation methods. The tools/software which are being reviewed allows for users to segment objects based on a certain desired input image. The advantages and disadvantages of each tool will be reviewed as well. Besides that, the research papers done on comparing deep learning algorithms and traditional segmentation methods (used by tools/software) will give a better insight as to why deep learning has been popular as of lately and give us a clearer picture on the limitation of existing object-based image segmentation software for segmenting desired regions in natural images. The review begins with comparing the existing tools before looking at the comparison of existing algorithms (deep learning vs traditional methods).

2.2 Image Annotation Lab

Image Annotation Lab is a renowned software tool invented by 4SmartMachines, a company that prioritizes in machine vision, artificial intelligence, and quality control in manufacturing [2]. The company came up with this software to allow users around the world to segment desired objects from images and since the software is free to use, it has the attraction of a lot of users, but the downside is that it only runs on the Windows operating system which limits the overall user count since other operating system (MacOS, Linux) users will not be able to enjoy the privileges of the software.

2.2.1 User Interface

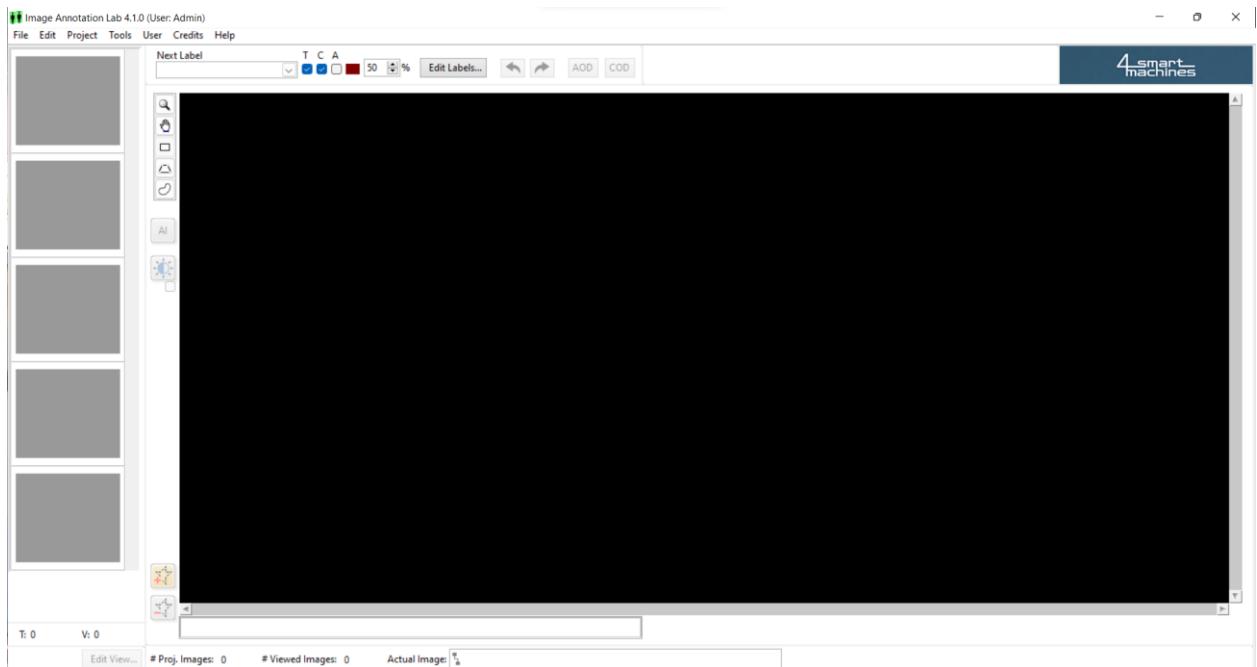


Figure 2.1 Image Annotation Lab User Interface

Image Annotation Lab has a straightforward interface that most users should be able to work out and understand when using the tool as shown in Figure 2.1. The left portion of the user interface is the portion that allows for users to view images that is readable by the software and there are a few buttons alongside the left portion that gives users the privilege to segment their desired objects from the input images inserted by them previously. The main canvas to view the segmented object takes up most of the interface and there is a label control section above the canvas that allows users to decide which layer to use when annotating input images. The input is subjected to different layers and each label interacts with the layer it is associated to.

2.2.2 Features and Functionalities

The usage of the tool begins with requiring the user to provide input images (png, jpg). Users can provide the directory to the file that stores all the input images and Image Annotation Lab will load all the input images directly into the interface providing the user with the convenience of inserting more than one input at a time rather than having to select them one by one. Once the input is loaded, users can then immediately start segmenting objects from the inputs by drawing out the contour of the objects

manually and the software then applies a unique colored layer which fills the contour drawn previously and uses this layer to perform the object-based segmentation.

Another experimental feature that Image Annotation Lab has is that the segmentation process can be executed using artificial intelligence (AI) techniques which helps the user when drawing the object's contour and it is widely known as Interactive Segmentation using AI techniques. This is achieved using positive clicks (left click) and negative clicks (shift + left click) to region of interest and the regions to be avoided, respectively. This method saves the user's time as they can draw contour for multiple objects within a matter of mouse clicks. A preview of how the contour is drawn around the desired object can be observed in the Figure 2.2 below.



Figure 2.2 Segmentation Result on a Mushroom (Image)

2.2.3 Advantages and Disadvantages

2.2.3.1 Advantages

The most obvious advantage of this software is that it allows object-based segmentation for images also not forgetting to provide users with convenience because it has a simple concept and it is easy to operate. Speaking of users' convenience, the software is able to load all readable inputs in a particular folder at one go rather than expecting users to provide inputs one at a time. Furthermore, the AI techniques feature within the software allows for users to segment their objects easily and if the image is too complex, users can still segment it manually to ensure maximum accuracy. The software then generates a closed boundary region automatically during segmentation which should help with the segmentation results.

2.2.3.2 Disadvantages

The main notable disadvantage is that the algorithms applied for the AI techniques within the tool is unknown. Since the software is designed to be user-friendly and simple, it is limited in terms of the features it has to offer when doing object-based segmentation. There is also a time-consuming element involved as most input images normally require a lot of clicks to segment the region of interest and user would find themselves having to manually do this most of the time. The AI techniques implemented within the software also tends to have some difficulty when trying to segment complex images which then resorts in users having to manually segment the image yet again. It is also a known fact that there a lot of input images that are subjected to noise and pre-processing is required on these noisy images before segmentation of the region of interest takes place however this software does not have the feature of pre-processing images. Therefore, users are expected to have pre-processed images before performing the segmentation.

2.3 Interactive Segmentation Tool

The Interactive Segmentation Tool is a freely available software that allows users to segment an object from input images based on the definition of the market on the object segmented. The software was released back in 2009 and it was created using the renown Java language which is why it is easily supported in most operating systems (Windows, MacOS, Linux). As long as the operating system can run the Java interpreter alongside its libraries, there should not be any issues in running this software tool. The Interactive Segmentation Tool was developed by Kevin McGuiness and Noel O'Connor from the University College Dublin as they were a part of the Center for Digital Video Processing [3] and the tool had the main objective of facilitating the development of interactive segmentation algorithms.

2.3.1 User Interface

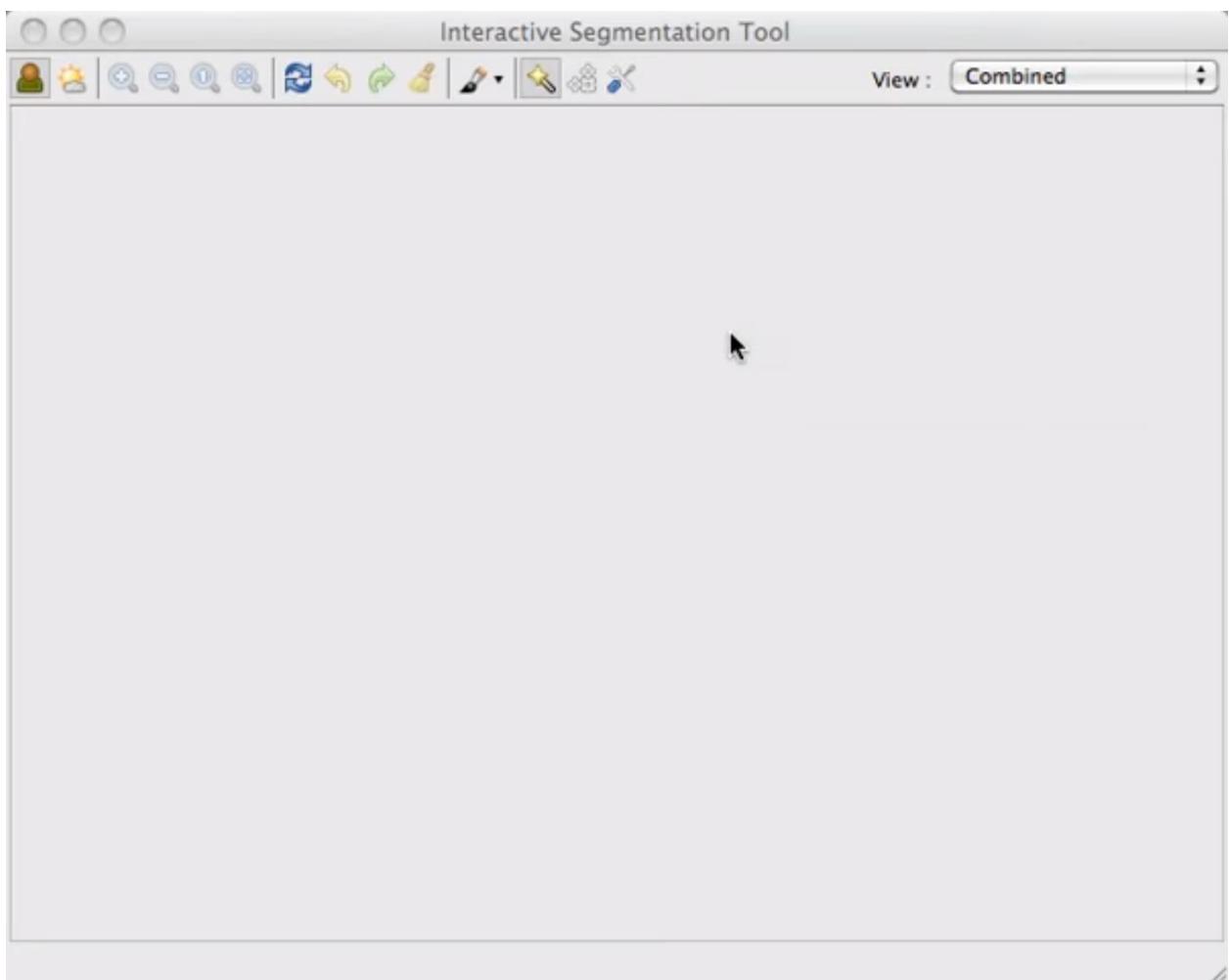


Figure 2.3 User Interface of Interactive Segmentation Tool

The user interface for this interactive segmentation tool prioritises in user experience meaning that it ensures that it is simple and easy to use as shown in Figure 2.4 above. The interactive segmentation tool has an easily understandable user interface because all it has is a single toolbar at the very top and the icons on the toolbar each represent a certain function. The images on the icons are self-explanatory in terms of its usage but user still has the option to hover their mouse over the icons to get a brief description of the functionalities of each icon. Users can then decide what the system will be displaying based on the drop-down bar available at the toolbar section.

2.3.2 Feature and Functionalities

This tool takes a straight-forward method when it comes to segmenting objects. Users are just required to mark the input based on two regions which are the foreground and background regions as shown in Figure 2.4 and Figure 2.6. The region of interest from the input image is segmented based on these regions as illustrated in Figure 2.5 and Figure 2.7. It is worth noticing that the interactive segmentation tool is subsided to the usage of four types of conventional interactive segmentation algorithms. The algorithms are Simple Interactive Object Extraction, Interactive Segmentation using Binary Trees, Seeded Region Growing and Interactive Graph Cuts [3, 10]. Interactive Segmentation using Binary Tree transforms a hierarchical region segmentation into an object-background segmentation by splitting and merging regions in the tree representation based on user interactions. The Seeded Region Growing algorithm characterizes the colour value of pixels that are connected and associated in a relevant region while the Simple Interactive Object Extraction algorithm is used when users intend on classifying the pixels in images as either an object or a background by identifying their colour models and then classify the pixel in the images based on the colour model. Then, the Interactive Graph Cuts algorithm works by capturing the hard constraints marked by user and soft constraints which is the pixel relationship with respect to the spatial and range domain in the images.

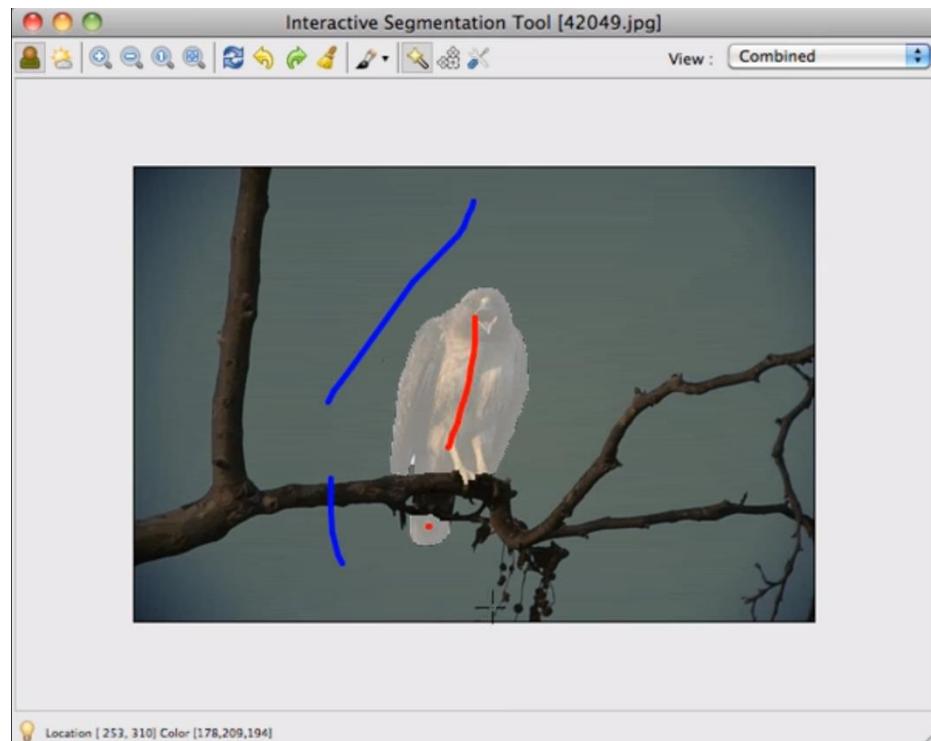


Figure 2.4 Input image of an eagle sitting on a branch (foreground and background labelled markers)

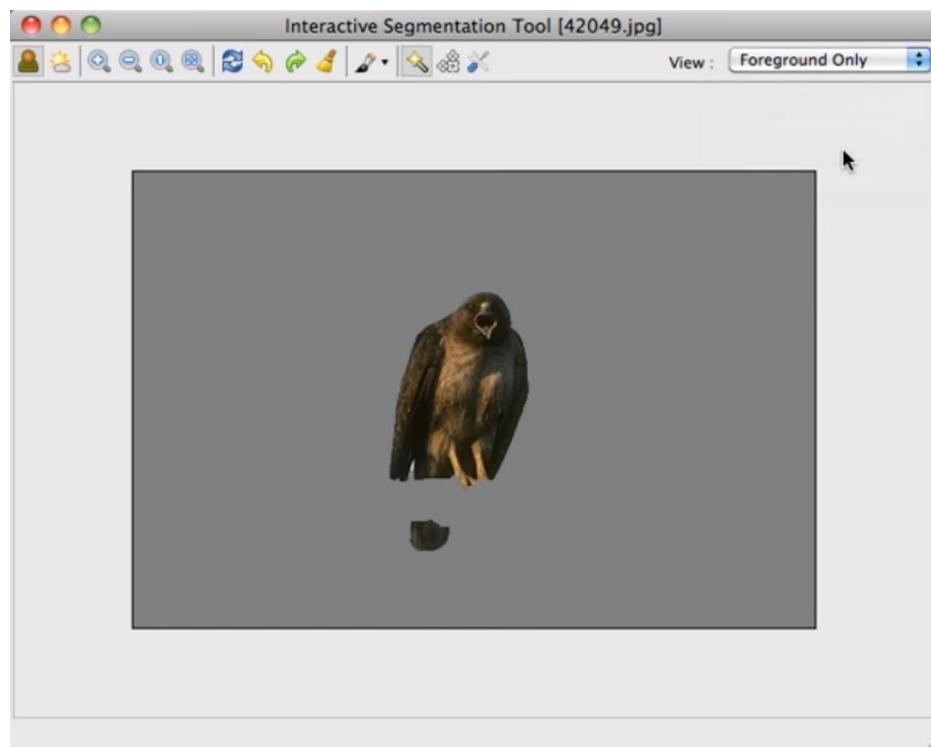


Figure 2.5 Segmentation Result of Eagle as Region of Interest

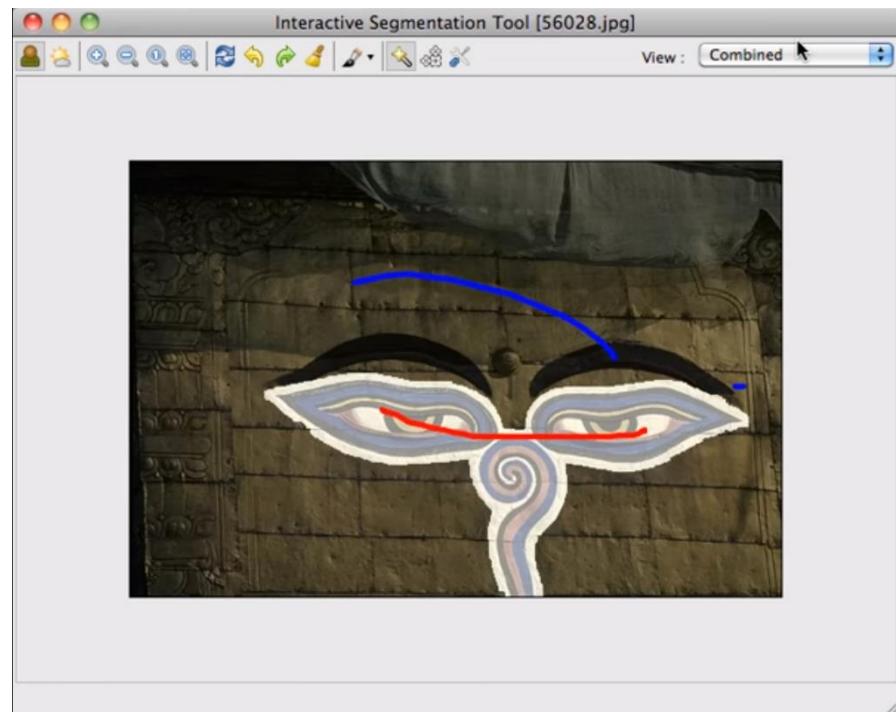


Figure 2.6 Input image of a painting on a background (foreground and background labelled markers)

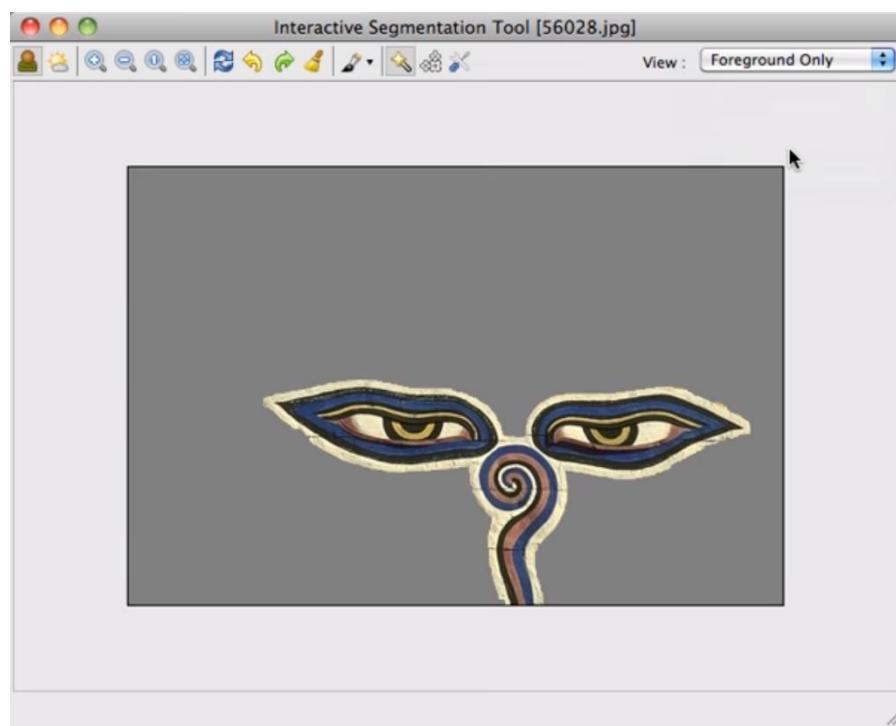


Figure 2.7 Result of Segmentation for painting as Region of Interest

2.2.3 Advantages and Disadvantages

2.2.3.1 Advantages

The software is advantageous because it has a simple approach in its usage and it is open source meaning that more experienced users can get a better understanding of the functionality of the tool. The tool tries to keep the user interaction minimal as users are only required to determine the background and foreground of the input image. The software uses a colour space (CIELUV) when implementing the Seeded Region Growing algorithm instead of the conventional grayscale which all in all results in a better performance [10].

2.2.3.2 Disadvantages

As expected, there is bound to be errors when segmenting object regions as the region of interest can be inaccurate at times. The software has a difficult time recognizing the region of interest when trying to segment noisy images. Since this a tool, it does require user intervention and input at some point in time which is inputting the marker onto the image when segmenting the region of interest which without a doubt can be very time consuming. The software too has difficulties in pre-processing the background noise in images therefore the processing speed and the final result of the segmented object is affected.

2.4 ImageJ

ImageJ is by far one the most well-known software tools available and it is mainly used in processing and analysing scientific images [4]. This tool has benefitted the medical industry in so many ways such as detecting cancerous cells and tumor in the brain. First designed in 1997 by Wayne Rasband, it has come a long way in its development with other developers contributing to the development which is why we have other variants of ImageJ such as ImageJ2 and Fiji. ImageJ can be run on any platform that supports the Java programming language therefore we can expect ImageJ to be able to run on most operating systems.

2.4.1 User Interface

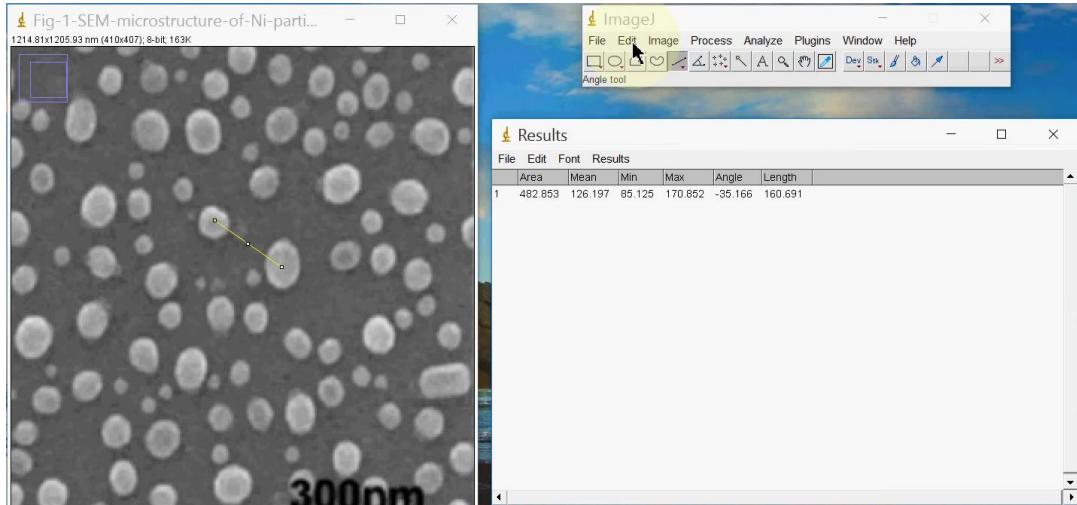


Figure 2.8 ImageJ User Interface

ImageJ has an initial user interface which just seems like a toolbar and not the usual application screen that you would get with other tools as shown in Figure 2.8 above. Upon launching ImageJ, users are greeted with just a horizontal toolbar and the input image only loads as a separate window for processing once the user has decided to input their desired image. All the image processing requirements can be accessed from this toolbar. It is evident that ImageJ does prioritise in user convenience as the interface does not take up the entire screen in the beginning.

2.4.2 Features and Functionalities

ImageJ indeed does offer a wide range of parameters for users to choose from when segmenting the region of interests which is why ImageJ can be perceived as versatile since users are subjected to these parameters and give them the freedom to choose the parameters that best fit their requirements and desired output when segmenting their desired objects. ImageJ also provides their users with various kinds of image pre-processing function which guides users to generate marker images that are accurate. Some of the notable functions would be adding a Gaussian Filter, smooth masking, sharpening mask and removing outliers [4]. Furthermore, users are also given the privilege to choose the image segmentation algorithm of their choice when segmenting the region of interest in an image. Some of these algorithms are Watershed algorithm, Sliding Paraboloid algorithm and the Rolling Ball algorithm [4]. Each of these algorithms serve a different purpose and it depends on the requirements of the user. As mentioned earlier, there are many ways to implement and use the functionalities of ImageJ, the figures below describe one way in using the features of ImageJ. User can select a specific boundary in the given entire input image and using thresholding in that specific boundary, ImageJ is able to identify all the unique cells in that given boundary. Then, users are given the option to export the final results to excel which is very convenient when users intend on sharing this result to other users on different platforms. Figure 2.9 and Figure 2.10 below shows how users can select the boundary within the image and identify the region of interest via the image segmentation algorithm of their choice and in this case, thresholding was used. Figure 2.11 and Figure 2.12 then shows how the results are stored and how it can be exported to other platforms via Excel.

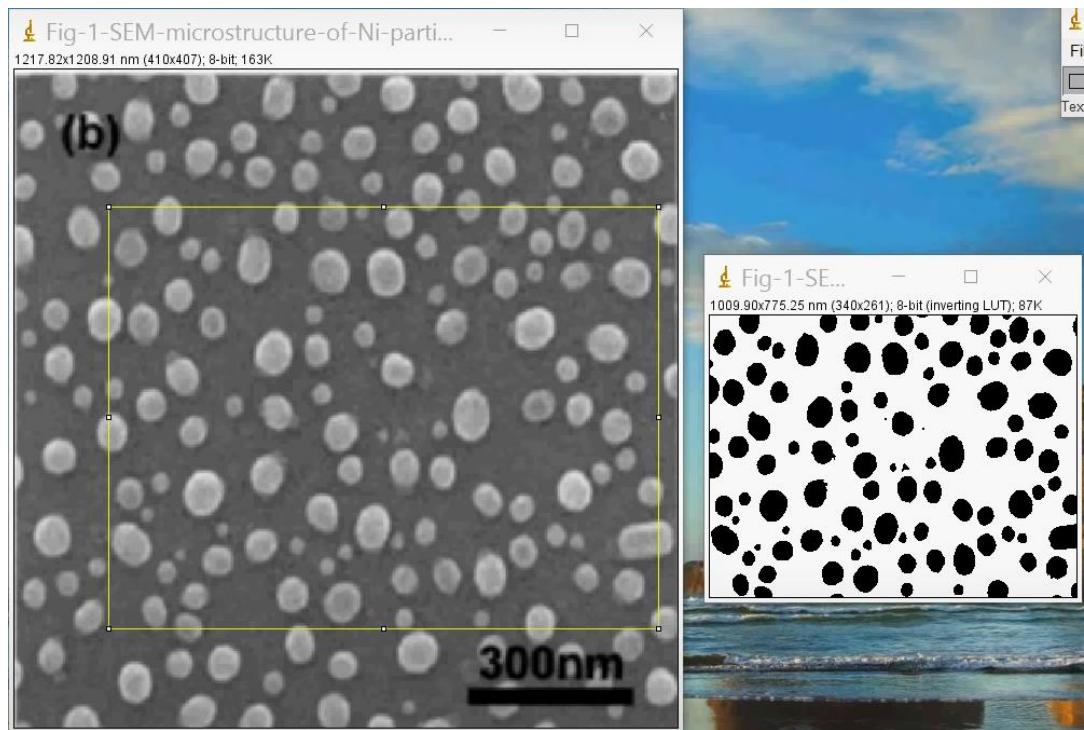


Figure 2.9 Selecting a boundary from the entire input image

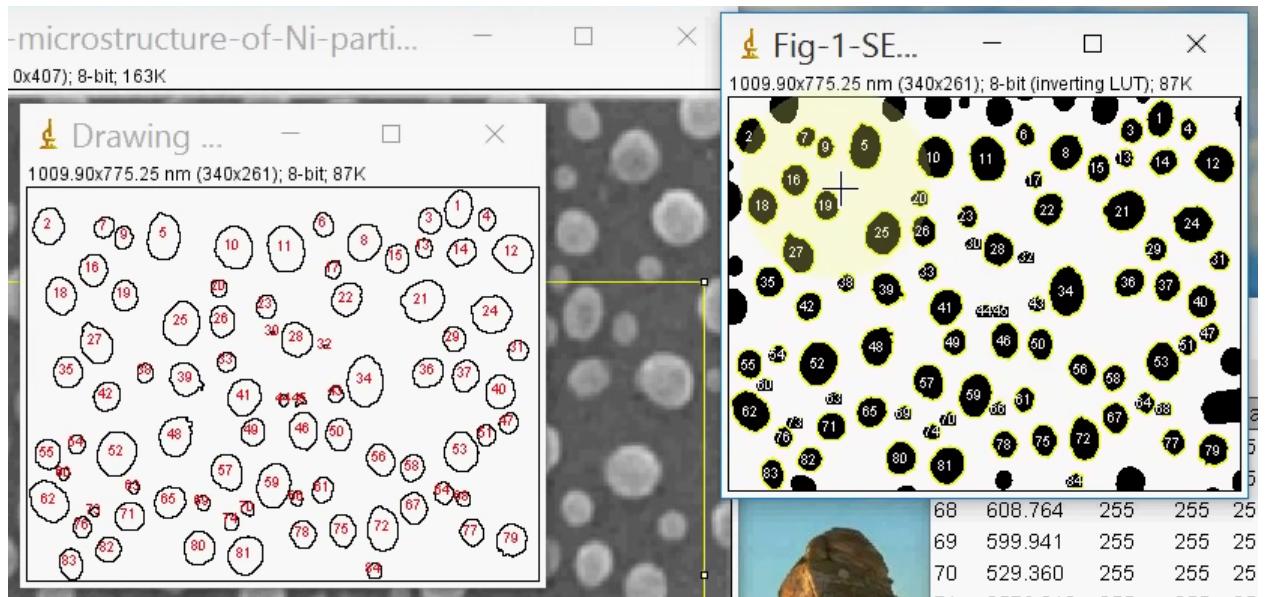


Figure 2.10 ImageJ identifying the cells via thresholding

	Label	Area	Mean	Min	Max
44		4040.780	255	255	255
45		2329.183	255	255	255
46		2602.686	255	255	255
47		3361.435	255	255	255
48		1896.873	255	255	255
49		4464.268	255	255	255
50		1596.902	255	255	255
51		4605.431	255	255	255
52		1217.528	255	255	255
53		2946.770	255	255	255
54		2496.814	255	255	255
55		2576.218	255	255	255
56		4111.362	255	255	255
57		2505.637	255	255	255
58		1146.946	255	255	255
59		1835.114	255	255	255
60		2090.971	255	255	255
61		2929.125	255	255	255
62		3167.337	255	255	255

Figure 2.11 Results of each respective cell in the selected boundary

	A	B	C	D	E	F
1		Label	Area	Mean	Min	Max
2	1		3043.819	255	255	255
3	2		3308.499	255	255	255
4	3		1799.824	255	255	255
5	4		1146.946	255	255	255
6	5		4570.14	255	255	255
7	6		1402.804	255	255	255
8	7		1243.996	255	255	255
9	8		3696.696	255	255	255
10	9		1235.173	255	255	255
11	10		4790.707	255	255	255
12	11		5214.195	255	255	255
13	12		4473.091	255	255	255
14	13		2232.134	255	255	255
15	14		2038.035	255	255	255
16	15		2699.735	255	255	255
17	16		3379.08	255	255	255
18	17		2329.183	255	255	255
19	18		5187.727	255	255	255
20	19		2990.883	255	255	255
21	20		1393.981	255	255	255
22	21		4464.268	255	255	255
23	22		4755.416	255	255	255
24	23		2355.651	255	255	255
25	24		3740.81	255	255	255
26	25		3211.45	255	255	255
27	26		1693.952	255	255	255
28	27		1235.173	255	255	255
29	28		1094.01	255	255	255
30	29		5496.52	255	255	255
31	30		2752.671	255	255	255

Figure 2.12 Results exported to Excel

2.4.3 Advantages and Disadvantages

2.4.3.1 Advantages

ImageJ is a software that is not only open-source but is compatible with devices that support Java plugins and can run applications using Java language. The notable advantage of ImageJ in comparison to the other tools is that it has a various kind of features that allows for users to get the output of their desire. Speaking of features, some of the useful image pre-processing features are such as finding object edges, sharpening image quality, enhancing image contrast, smoothing edges of selected objects and obtaining binary images from the original images. These features are implemented within a click of a button and the image that is being segmented is displayed on a new window which helps prevent interference of the segmented image and the marker image. The marker image and segmented image are displayed in their own respective windows. Each image might have more than segmented object and all these objects can be viewed via the “Analyze Particles” feature.

2.4.3.2 Disadvantages

ImageJ is unable to segment the region of objects that have a colour that is similar to the colour of the background region and this is because the object-based segmentation is performed using an algorithm known as the colour thresholding algorithm which cannot detect ambiguous boundaries between background region and the object region. As usual, ImageJ is still a tool that requires user input when making the region of interest to perform image segmentation and there is no doubt that this will inevitably be time consuming and it is more obvious for a complex image that has various region of interests. Since ImageJ is a well-known tool, there are a lot of available plug-ins and users are expected to install these plug-ins to be able to use their features and sometimes it can be challenging for users to determine which plug-in to use. Users might experience exportation issues at times as the segmented object cannot be stored separately on their local drive therefore users will have to export the entire project for other users to be able to view the segmented object which can be tedious at times.

2.5 Understanding Common Algorithms for Image Segmentation

Since a brief understanding on image segmentation has been established, it is time to dive in a little deeper as to what some of the common algorithms implemented in image segmentation and what is the approach that stands out amongst the existing algorithms. It is known that deep learning techniques are currently coming up against traditional Computer Vision (CV) [5]. First, there is a need to understand the usage of deep learning algorithms before proceeding with understanding some of the well-known traditional CV techniques. Deep learning (DL) is well known in the field of digital image processing recently because it helps users solve difficult problems (image colorization, classification and segmentation). One of the well-known methods within DL is Convolutional Neural Networks (CNNs) and it is used mainly because it improves prediction performance. Problems that were once assumed as unsolvable are now being solved with high accuracy and image segmentation is one good example of this [5].

It is better to first understand main difference between traditional CV methods and DL algorithms. Rapid progressions in DL and improvements in device capabilities (computing power, memory capacity, power consumption, image sensor resolution) have improved the performance and lowered the cost to produce vision-based applications. DL enables engineers to achieve greater accuracy in tasks such as image classification and segmentation. This greater accuracy is achieved because DL algorithms are trained rather than programmed, applications using such an approach normally require less analysis from experts and fine-tuning is not necessary all the time. This brings to one of the main differences between traditional CV methods and DL algorithms which is their workflow. Workflow here refers to the feature extraction and the classifier used in producing the output. Based on Figure 2.13, it can be observed that the differences between these two workflows.

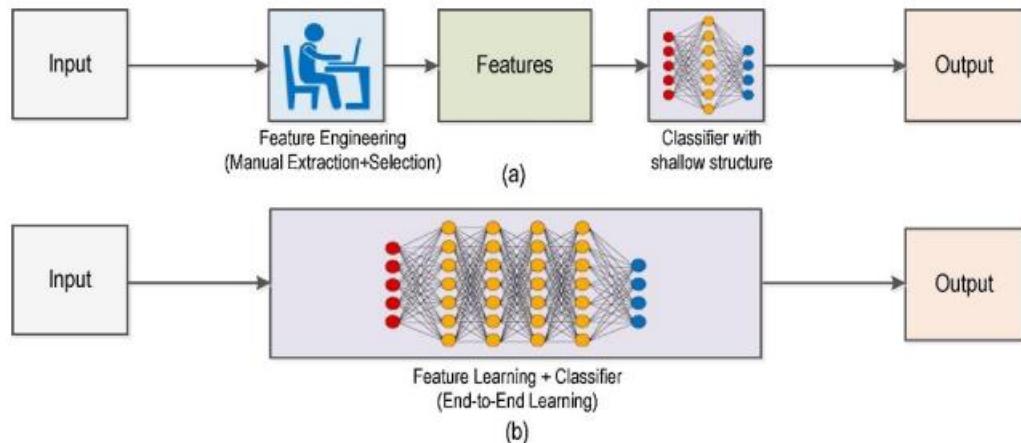


Figure 2.13 (a) Traditional Computer Vision workflow vs. (b) Deep Learning workflow

In traditional CV methods, users are expected to manually extract certain features (boundary selection for example) and select which regions should be the region of interest prior to segmenting the image. The classifier has a shallow structure as well. However, it is observable that DL uses end-to-end learning where the machine is given a dataset of images which have been annotated with what classes of objects are present in each image which ‘trains’ the DL model and to take it even further, CNN algorithms can discover the underlying patterns in the classes of the images and try to find for similar patterns in new input images as well [5].

A better understanding can be obtained on the efficiency of some existing algorithms based on a comparison on segmentation methods that has been researched on recently and the research concerns itself with segmenting dental panoramic images. The algorithms used in this research are Global Thresholding (thresholding segmentation), Fuzzy C-Means (clustering-based segmentation), Watershed (mathematical morphology approach), Canny Edge Detection (edge-based segmentation) and U-Net Architecture (convolutional neural network) [6]. It is safe to say that the U-Net Architecture is the algorithm that uses deep learning techniques in segmenting the image(s) region of interest in comparison to the other algorithms stated. For this research, these algorithms are trying to segment an X-ray image of a person’s teeth and the final result is compared to the given ground truth so that the accuracy of these algorithms can be compared as shown in Figure 2.14.

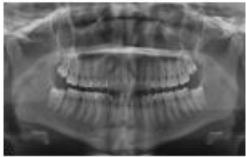
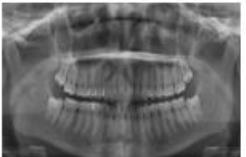
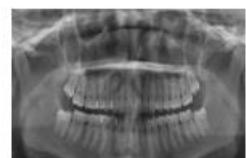
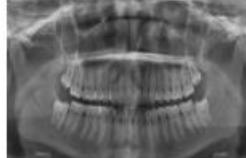
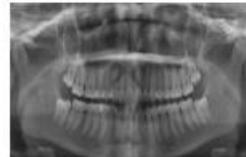
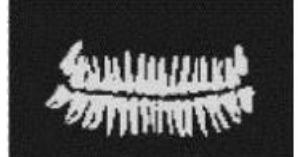
Name of the Segmentation	X-ray Image	Ground Truth	Segmented Image
Global Thresholding			
Watershed Segmentation			
Canny edge detection			
Fuzzy C-means Clustering			
U-Net Architecture			

Figure 2.14 Experimental Analysis of Various Segmentation Methods

It is evident that the U-Net Architecture had the best performance out of the other algorithms that leaned more towards traditional CV methods. U-Net Architecture was the only segmentation method that entirely applied deep learning techniques in segmenting the region of interest. It can be observed that the accuracy of these algorithms from some statistical analysis. Based on Figure 2.16, it also evident that the U-Net Architecture gave us the best results [6].

Segmentation Method	Accuracy	Specificity	Precision	Recall	F1-Score	Dice Score
Global thresholding	0.79	0.81	0.52	0.69	0.56	0.74
Fuzzy C-means	0.82	0.91	0.61	0.45	0.49	0.81
Watershed	0.77	0.75	0.48	0.82	0.58	0.75
Canny Edge detection	0.79	0.94	0.45	0.11	0.17	0.54
UNet	0.97	0.95	0.93	0.94	0.93	0.94

Figure 2.15 Result Summary for Algorithm Efficiencies

Based on this research, it is safe to conclude that deep learning techniques are advantageous in comparison to the conventional image segmentation algorithms. However, it does not necessarily mean that it is true for all cases. This is because despite deep learning undoubtedly being more accurate, though sometimes it can be perceived as an overkill and it is unnecessary at times [5].

2.6 Critical Remark and Limitations of Tools

This portion of the literature review will focus on comparing the advantages and disadvantages of the tools reviewed earlier. A better understanding can be obtained on not only the usage of existing algorithms but also give us a better overall perspective of the usage of these tools as the algorithms and tools do complement each other at the end of the day. Overall, both Image Annotation Lab and the Interactive Segmentation Tool uses interactive segmentation approach whereas ImageJ allows for users to choose their own algorithms when performing the interactive segmentation to obtain the region of interest and because of this, ImageJ can be said to have the greatest number of features since users are given options to begin with [4].

Table 2.1 Comparison of ImageJ, Interactive Segmentation Tool and Image Annotation Lab

	ImageJ	Interactive Segmentation Tool	Image Annotation Lab
Advantages	<ul style="list-style-type: none"> - Subjected to a lot of features - Subjected to the usage of various algorithms when segmenting region of interests 	<ul style="list-style-type: none"> - Uses semi-automatic segmentation (more user input) - Segmented results displayed on a different plane (viewing access) - User friendly 	<ul style="list-style-type: none"> - Uses manual and semi-automatic segmentation (partial user input or entire user input) - Segments regions of interest from videos - User friendly
Disadvantages	<ul style="list-style-type: none"> - Ambiguous boundaries between object region and background region is undetected - User effort required when marking region of interest and installing additional plugins when using certain features - Users cannot store segmented objects separately 	<ul style="list-style-type: none"> - Noisy images issue (Unable to segment region of interest) - No pre-processing methods for noisy images - Human interaction is required when marking region of interest 	<ul style="list-style-type: none"> - Software is not open-source (cannot understand features entirely) - Human interaction required when marking region of interest - Image pre-processing for noisy images is unavailable

2.7 Proposed Solutions

This project aims to propose solutions to limitations brought about by the reviewed software tools. The obvious limitation that all these tools have is that they require human interaction to a certain extent in order to be able to accurately segment the region of interest and even so, there is chance that the segmentation may fail due to some limitation within the algorithm used by the tool. The existence of the limitation of such algorithms alongside the limitation of the tools undoubtedly causes user inconvenience. Therefore, the project proposes to develop an application that is able to segment region of interests from a given input without the requirement of human interaction. The proposed application is to be implemented using deep learning techniques so that the region of interest can be segmented accurately. In addition, since the deep learning techniques have end-to-end learning, therefore the users will not have to fine-tune the parameters all the time.

Chapter 3

System Methodology

3.1 Methodology

Since there is an intention on implementing deep learning techniques for natural image segmentation, there should be an approach that is similar to the data science workflow. The processes of the project can generally be broken down into different phases such as analysis, development, deployment and testing. The development phase is the phase to look out for as this phase contains important activities such as data pre-processing, model training, data training and performing the image segmentation against a test to observe the deep learning model's efficiency. A prototype of the application is also developed during this phase. Based on the existing methodologies in project development, the scrum methodology seems to be a good fit for this project simply because this application will definitely be developed based on an iterative and incremental process [7]. During each sprint cycle, a function of the application is developed (data pre-processing function, implementing a deep learning technique, ensuring hyperparameters of algorithm are fine-tuned, implementing proper plug-ins, build base functions for applications to host the deep learning algorithm, etc.) and this function is then subject for analysis and review so that improvements can be made accordingly. The duration of each cycle is yet to be determined as a better understanding of the time required for each cycle can only be determined once the development of the project begins. However, the scrum methodology allows for the student to ensure that each function developed in a sprint cycle aligns with the project objectives and the project supervisor can provide their input during the deliverables presentation for each sprint cycle. Using this methodology, student can ensure that the deep learning model and application development is being done with a proper workflow. The scrum methodology allows for the project to be broken down into the main functional objective (features that intend to achieve the main project objectives) and the non-main functional objectives (features that improve user experience but does not serve great importance). This way, students can have a better planning as to which portions of the project should be given the most attention to begin with. All in all, an application to segment the region of interest using deep learning techniques can certainly be

developed by following the generic data science workflow as well as implementing the scrum methodology during the development of the project.

3.2 Tools used

Hardware

Laptop

Table 3.1 Laptop specifications

Description	Specifications
Model	Illegear Onyx V Ryzen
Processor	AMD Ryzen 7 4800H (2.90 GHz)
Operating System	Windows 11
Graphic	NVIDIA GeForce RTX 2060 6GB GDDR6
Memory	16GB DDR4 RAM
Storage	512GB SATA HDD

Software

-Google Colab

The web-based integrated development environment (IDE) known as Google Colab is indispensable for streamlining work and monitoring the segmentation algorithm development process. Code and the results it produces can both be viewed simultaneously in Jupyter Notebook in the Google Colab. For the purposes of training, validating, and testing the model, it is convenient for error checking and visualising.

-Python Language

The majority of the system for this project is constructed using the Python programming language. This is due to the numerous helpful built-in libraries that Python offers for deep learning-based machine learning and image processing. Building the model for training can be greatly simplified and easily visualised throughout the coding thanks to the language's ease of reading.

-Tensorflow/Keras

Python is used to create the Application Programming Interface (API) known as Keras. Because Python is a simple language to read, Keras is a high-level API for creating

neural networks that is available as an open-source library. Running on top of Tensorflow, Keras is quick to conduct experiments while constructing deep neural networks.

-Flask

Flask is used to deploy the deep learning model and put up the suggested application for general use. The web application can be created using Flask, a micro web framework developed in Python, without the need for any special tools or libraries. As a result, it is compact and simple to integrate with the paradigm suggested in this study. With Flask, the proposed deep learning model receives segmentation queries from the user and responds to those requests by sending the segmented image result to the user.

-Visual Studio Code

Visual Studio Code is an integrated development environment (IDE) that is used in computer programming and it has an extensive usage for application development supporting a huge range of languages. This IDE is used when developing the web application using Flask and provides a user interface for public usage.

3.3 User Requirement

- The user should be able to load the Oxford Pet Dataset (version 3)
- The user should be able to view the image as it is being processed for segmentation
- The user should be able to view the segmented image, true mask and the original image
- The user should be able to save the segmented pet (dogs and cats) region image
- The user should be able to input an image of their choice into the web application
- The user should be able to display the input image
- The user should be able to display the Segnet segmented image
- The user should be able to display the Unet segmented image
- The user should be able to view the implementation on google colab from the web application

3.4 System Performance Definition

For the natural image (pets) segmentation, the accuracy of the segmented tumor region is evaluated to indicate the performance of the proposed system. The deep learning image segmentation algorithm that is used in the implementation of image segmentation aims to provide a higher accuracy and low loss values for the natural image segmentation in the validation set and also ensuring that the model does not overfit. The training and validation loss using sparse categorical-crossentropy (SCE) is used for the deep learning model evaluation. A loss function is used to calculate the quantity that the model needs to seek in order to minimize during training and SCE is used as a loss function for multi-class classification model where the label for the output is assigned to an integer value (0, 1, 2, 3, 4.....) [12]. Other than the loss function, there is also the accuracy metrics which is used to determine the accuracy of the segmented natural image. The usage of the loss function, accuracy metrics, epoch value and the number of filters set in the convolution block all play a role in the final accuracy of the segmented image.

3.5 Dataset

In this work, the dataset is from the Visual Geometry Group (VGG) of the University of Oxford and it is called the Oxford-IIIT Pet Dataset [11]. The dataset consists of a 37 category pet dataset and each category roughly containing 200 images for each class. These images have a huge variation in scale, lighting and pose. Each image have an associated ground truth annotation of head ROI, breed and pixel level trimap segmentation. There's a total of 7349 images for training, testing and validation and it is 800MB in size [11].

Figure 3.1 below shows an example of the image alongside its true mask.



Figure 3.1 Input Image with it's True Mask

3.6 Model Architecture

3.6.1 Unet based model architecture

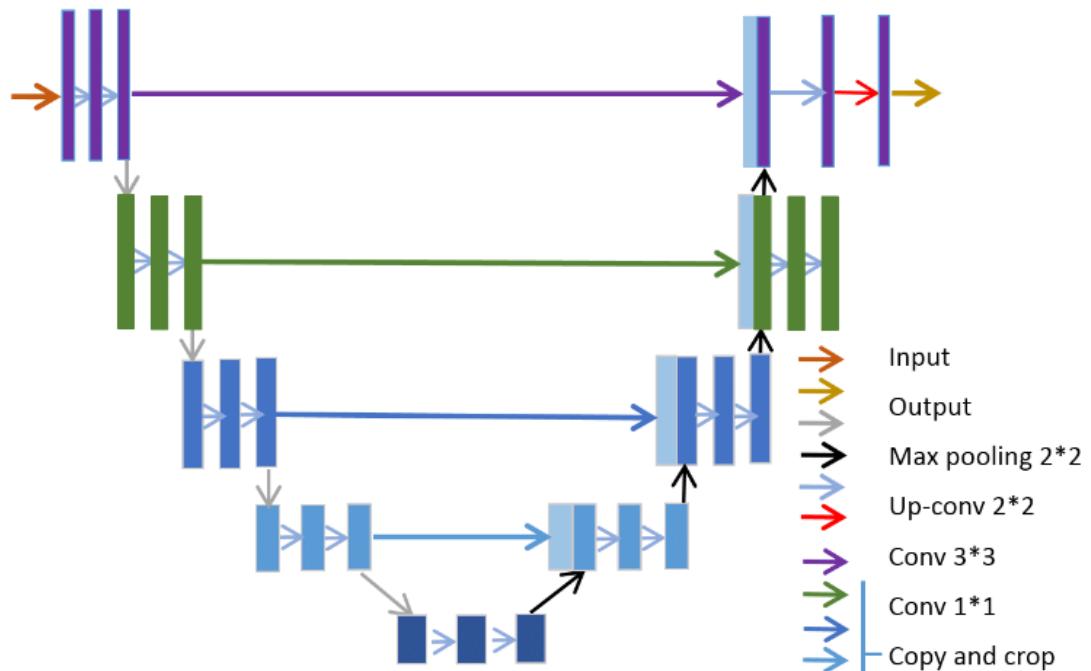


Figure 3.2 Visualisation of the Unet architecture

Figure 3.2 above shows the visualisation of the Unet architecture. Unet architecture, a novel CNN-based network architecture, was introduced by Ronneberger in 2015 [15]. From the feature extraction (encoding) phase to the decoding phase, it crops and concatenates patterns the model learned (predictions). Because of the 'U' shape the architecture adopts during implementation, this network is known as Unet [13].

Bachelor of Computer Science (Honours)

Faculty of Information and Communication Technology (Kampar Campus), UTAR

encoding phase of the Unet resembles any typical convolutional neural network (CNN). The image is downsampled using a 2×2 max pooling after the convolution blocks are constructed by stacking two 3×3 convolutions (green arrow). Reducing the dimension of the target data is known as downsampling [13]. The number of channels doubles from the starting number of channels at each downsampling step [13]. When the model reaches the bottleneck region, which is the lowest point on the 'U' shape, the downsampling process is then terminated. The data enter the decoding phase after passing through the bottleneck region (right side of the architecture which moves from bottom to up). To upsample the data in this phase, two 2×2 up-convolutions are performed (purple arrow), followed by two 3×3 convolutions (green arrow). Upsampling refers to increasing the target data's dimension [13]. The number of channels is halved every time an upsampling step is taken. In order to provide information from the encoding phase, the decoding phase also performs concatenation (grey arrows) of feature maps from the encoding phase after each 2×2 up convolution. This is crucial because without it, information would be lost after each convolution due to the loss of border pixels, which connects the encoding and decoding phases. [13]

3.6.2 Segnet model architecture

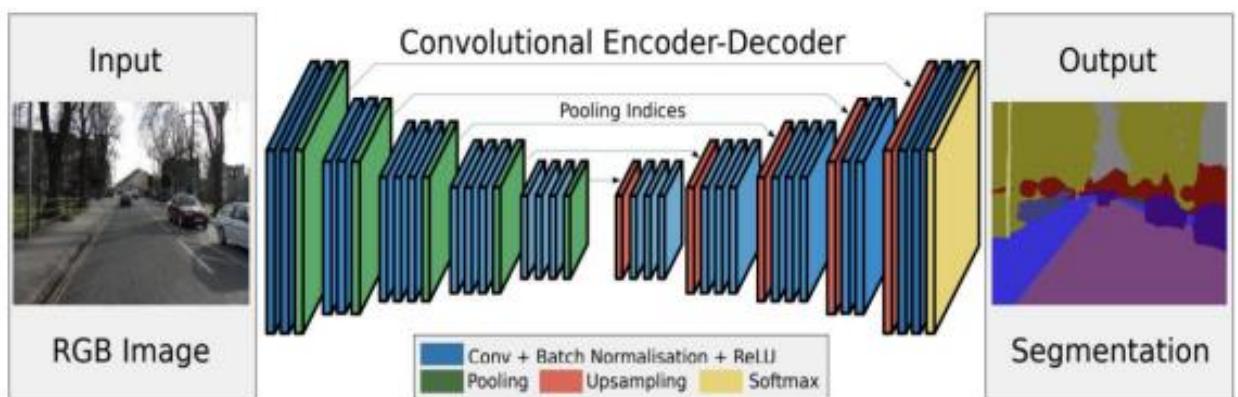


Figure 3.3 Segnet Architecture

Segnet is a type of convolutional neural network (CNN) model that was designed for semantic segmentation of images and this model consists of an encoder and decoder network [16]. The encoder network of the Segnet model extracts the features from the input image using convolutional layers and downsampling operations which then results in a lower-dimensional representation of the original input image. The network

of the decoder then uses a series of upsampling operations and deconvolutional layers to produce a high-resolution segmentation map. These descriptions mentioned are very similar features to what the Unet model offers but there are features to the Segnet model that is unique. One key feature that is unique to the Segnet model is the usage of the max-pooling indices which are recorded during the stage of encoding and later it is used in the decoding stage to perform upsampling efficiently. This will help preserve the spatial information in the map of the segmentation and reduces artifacts that could probably arise from using traditional upsampling methods. Furthermore, the model uses connections that skip to transfer information from the encoder to the decoder which then allows for the fusion of high-level and low-level features and improving the quality of the segmentation results.

3.7 Timeline

The timeline will be broken down into the works done for both FYP1 and FYP2. For FYP1, it began by reviewing previous works and understanding the current project objectives. Once the objectives were understood, data collection was performed, and some pre-processing activities was executed on the data collected. A work plan was then developed to ensure the model development (pretrained Unet and Unet from scratch) could be executed within the timeframe. The model development then took place which is the portion that took up the most time. Once the model was completed, the model was then trained and evaluated for its accuracy. Finally, FYP1 concluded with the report writing and submission. For FYP2, it began with the Segnet model research, implementation and training was performed. Once all the models have a solid foundation, then the hyperparameter tuning and refinements were performed on all models. The models were then deployed to the web app to begin designing the web application before concluding with the report writing and submission. Figures 3.4, 3.5, 3.6 and 3.7 below shows the timeline and the activities performed for both FYP1 and FYP2 respectively.

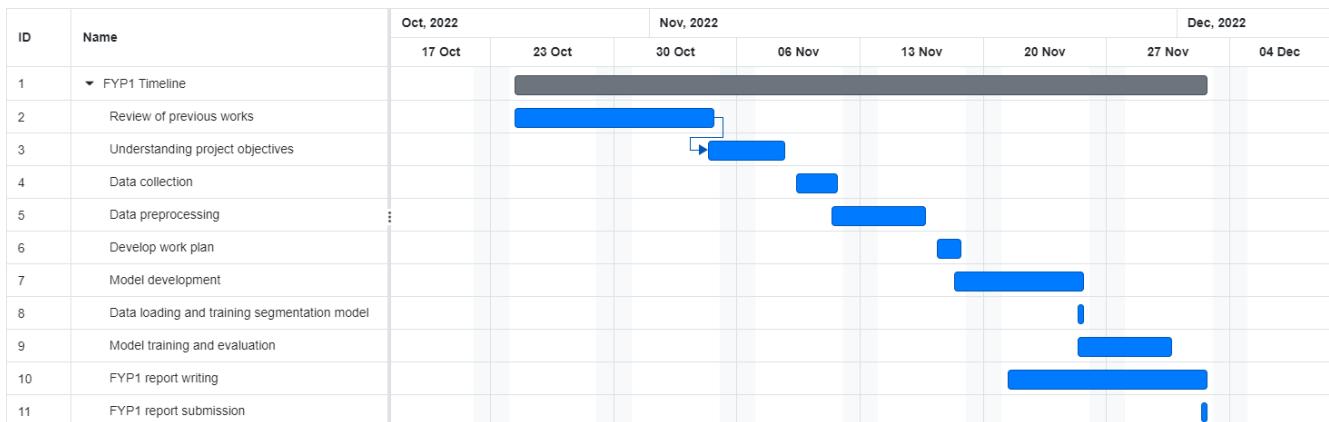


Figure 3.4 Gantt Chart of FYP1 Project Timeline

ID	Name	Start Date	End Date	Duration
1	▼ FYP1 Timeline	Oct 24, 2022	Dec 02, 2022	30 days
2	Review of previous works	Oct 24, 2022	Nov 04, 2022	10 days
3	Understanding project objectives	Nov 04, 2022	Nov 08, 2022	3 days
4	Data collection	Nov 09, 2022	Nov 11, 2022	3 days
5	Data preprocessing	Nov 11, 2022	Nov 16, 2022	4 days
6	Develop work plan	Nov 17, 2022	Nov 18, 2022	2 days
7	Model development	Nov 18, 2022	Nov 25, 2022	6 days
8	Data loading and training segmentation model	Nov 25, 2022	Nov 25, 2022	1 day
9	Model training and evaluation	Nov 25, 2022	Nov 30, 2022	4 days
10	FYP1 report writing	Nov 21, 2022	Dec 02, 2022	10 days
11	FYP1 report submission	Dec 02, 2022	Dec 02, 2022	1 day

Figure 3.5 Activity done during FYP 1

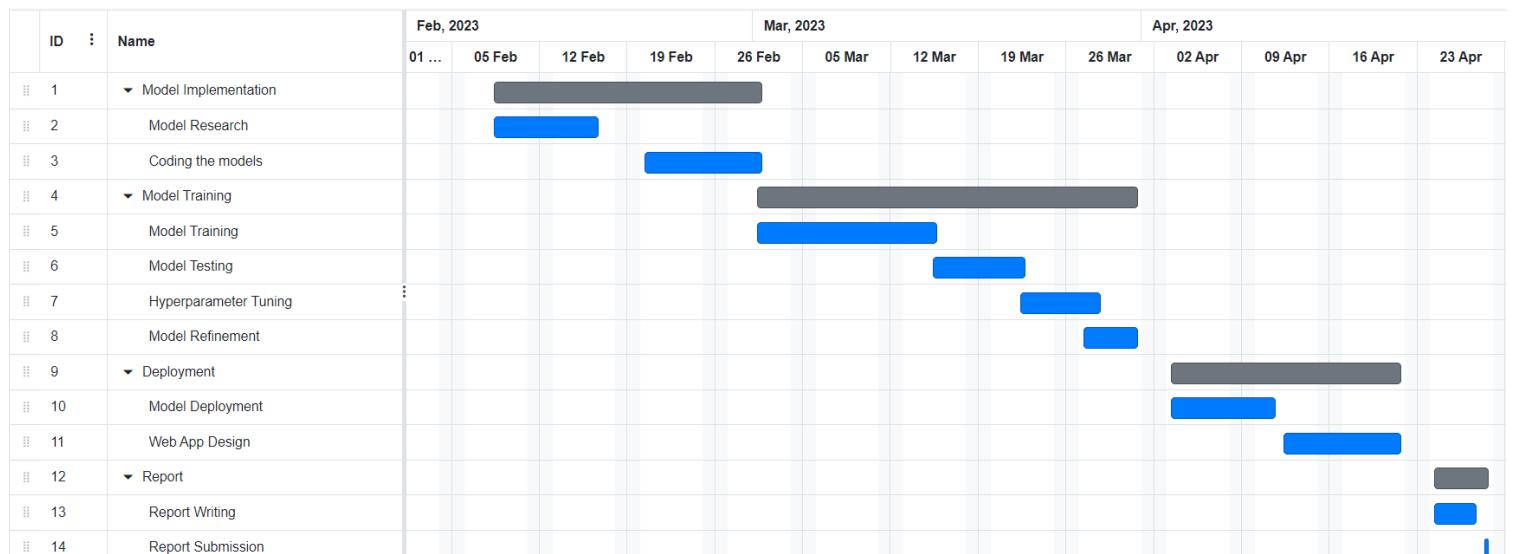


Figure 3.6 Gantt Chart of FYP2 Project

ID	Name
1	Model Implementation
2	Model Research
3	Coding the models
4	Model Training
5	Model Training
6	Model Testing
7	Hyperparameter Tuning
8	Model Refinement
9	Deployment
10	Model Deployment
11	Web App Design
12	Report
13	Report Writing
14	Report Submission

Figure 3.7 Activity done during FYP2

3.8 Chapter Summary

The methods used to create the web application and the suggested deep learning model that is integrated into the web application were both described in Chapter 4. This chapter also described the assessment metrics, datasets, resources, and model architecture utilised to construct the suggested deep learning model and the web application, as well as the difficulties encountered throughout the project's implementation.

Chapter 4

System Design

4.1 Model Creation Flow Chart

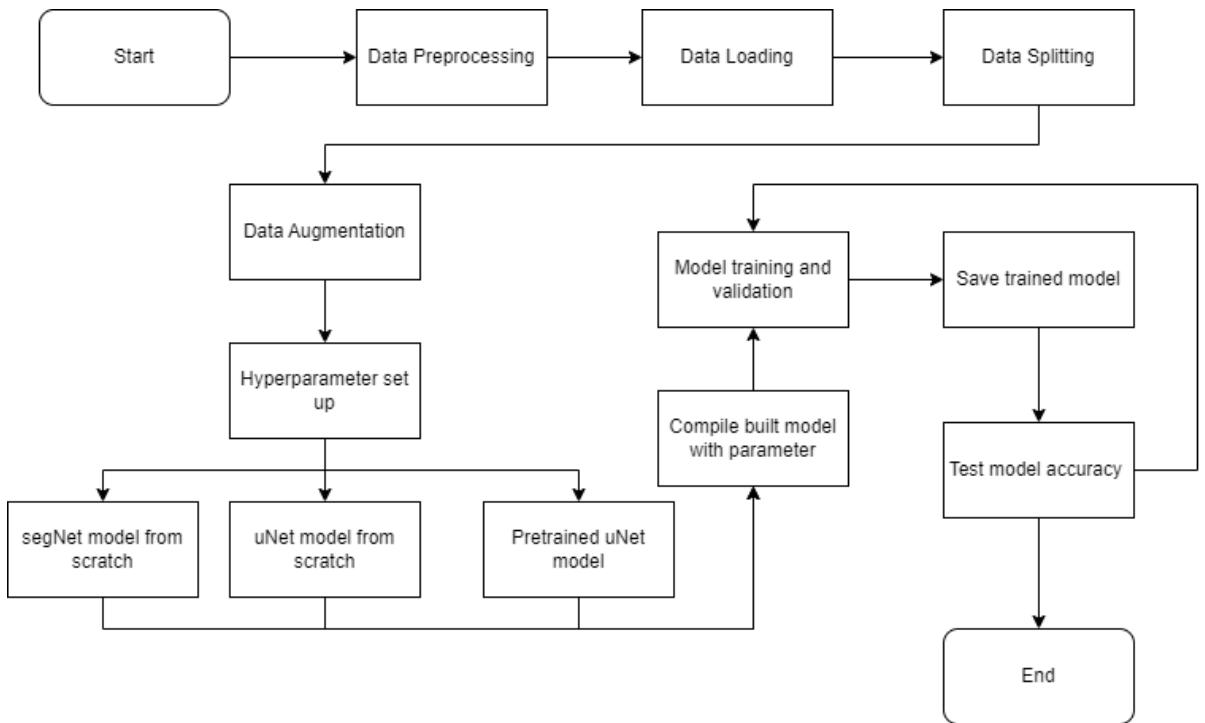


Figure 4.1 Model Creation Flow Chart

Figure 4.1 above shows the overview of all creation steps in implementing the proposed system. The approach begins with the data preprocessing where the training dataset used are all set to a specific size for training and then it is loaded accordingly for model input. The data splitting process is to split the data into training, testing and validation batches respectively. Once the data has been split, data augmentation is performed on the images to introduce a sense of variation such as horizontal, vertical and blurry augmentations. The initial hyperparameters (optimizers, loss function and metrics) are all then chosen based on best practices for each of the models which are the Segnet model, Unet model and pretrained Unet model respectively. These models are then compiled with the hyperparameters chosen before starting off with the model training and validation. Once the model has been trained with its appropriate hyperparameter settings and epoch value, the model is then saved into a h5 file which can be used in other platforms such as Visual Studio Code and implementing it within the Flask

framework. The model that is saved then is then tested for its training and validation statistics which are the loss and accuracy respectively. The accuracy and loss results will ensure the acceptability of the model for usage within the web application later.

4.2 Overall Flowchart of Proposed System

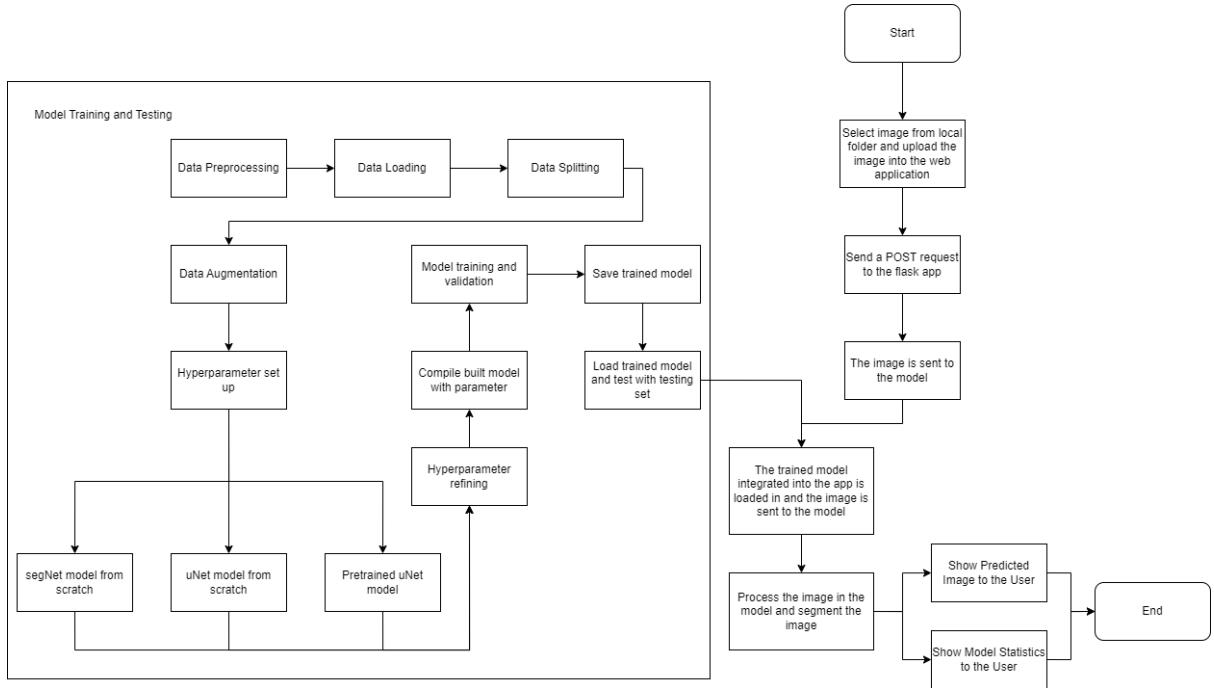


Figure 4.2 Overall flowchart of the proposed system

The model training and testing portion of the flowchart explained in Figure 4.2. This flowchart includes the external implementation of the web application and how it interacts with the creation of the model. The model training, testing and creation is done on Google Colab and a final h5 file of the model is stored for usage later within the Visual Studio Code environment alongside the Flask framework implementation. The web application allows for user to select any image of their choice and upload that image into the web application. The web application then sends a POST request to the Flask framework to allow the model to work on the input image from the user. The model then predicts the image and returns the image back to the user with the help of the web application's functionalities supported by flask.

4.3 Activity Diagram

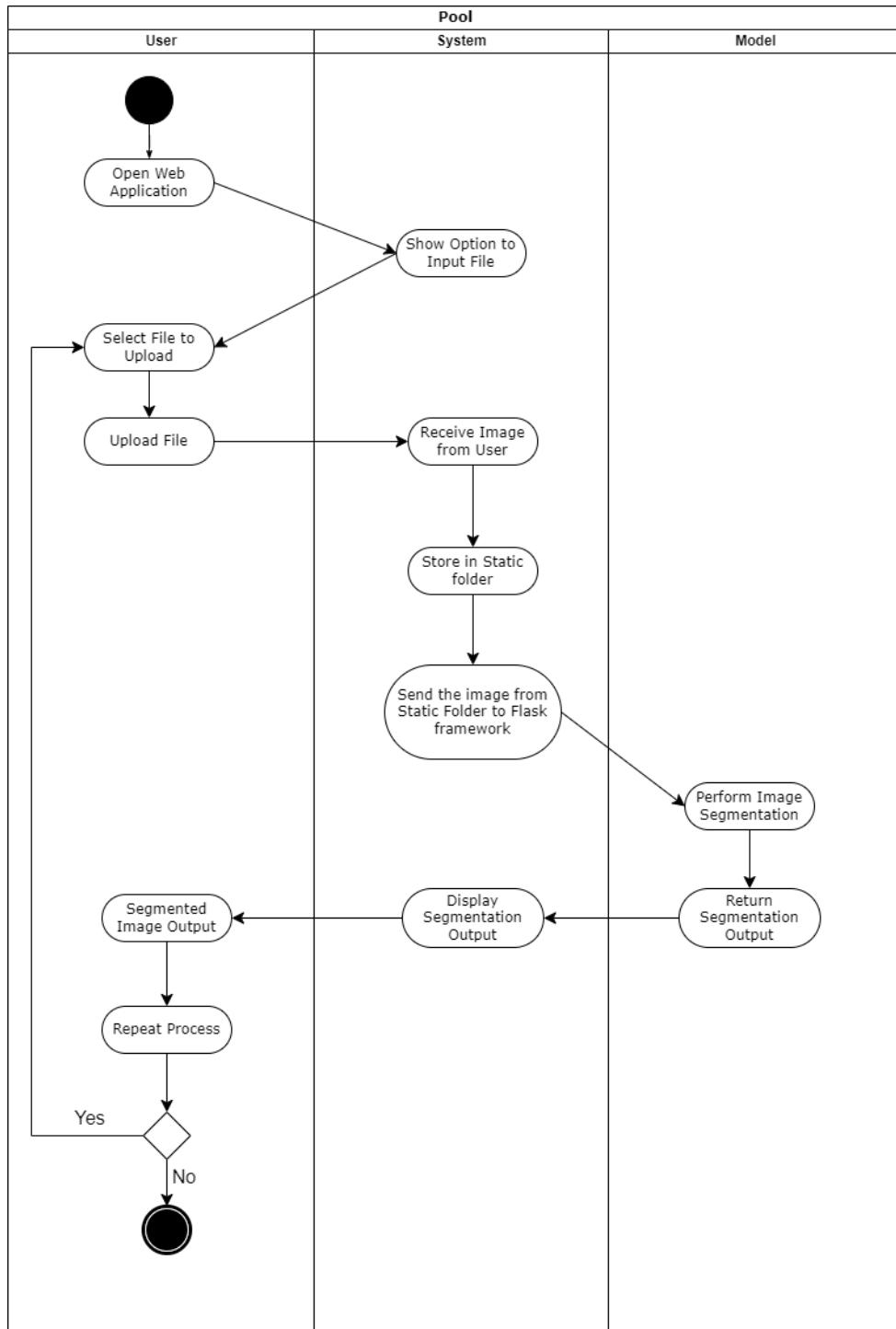


Figure 4.3 Activity Diagram of Proposed System

The activity diagram in Figure 4.3 above shows the activity diagram that depicts how the user, system and the deep learning models interact with one another to ensure a user experience that is smooth while using the web application. The user will first open the Bachelor of Computer Science (Honours) Faculty of Information and Communication Technology (Kampar Campus), UTAR

web application hosted under an Internet Protocol (IP) address that is publicly available and this IP is connected to port 5000. Once the user has connected to the website, the system then shows the user a user interface that allows the user to input an image of a cat or a dog of their choice and then upload this image. The web application then stores the image in its static folder. The system then sends this image from the static folder to either the Segnet or Unet model to perform segmentation. Once the model has performed the segmentation, it then returns the output to the system so that the system can display the output to the user. The user can then view the segmented image output and also have the option to be redirected to the Colab implementation for that particular model (Unet or Segnet). The user also has the option to return back to the main page and input another image of their choice and the execution process of producing the segmented image begins again.

4.4 Verification Plan (Deep Learning Model)

Table 4.1 Verification Plan (Epoch Value)

Procedure Number	P1
Method	Analysis
Applicable Requirements	Analyze the suitable epoch values for training the model while ensuring there is no degradation in system performance.
Purpose/Scope	To improve the accuracy of the trained model while avoiding the overfitting issue
Item Under Test	Number of natural images of dogs and cats in the dataset
Precautions	The epoch values should not be too low or too high to avoid underfitting and overfitting issues
Limitations	If there are very little images in the dataset, epoch values would not have much effect on the accuracy of the model
Equipment/Facilities	Laptop
Data Recording	None
Acceptance Criteria	Analyze the suitable epoch values
Procedure	<ol style="list-style-type: none"> 1. Load in the natural images dataset 2. Set the epoch values for the training of the model accordingly 3. Analyze the impact on the performance of the model
Troubleshooting	Iterate this procedure
Post-Test Activities	None

Table 4.2 Verification Plan (Optimizers)

Procedure Number	P2
Method	Analysis
Applicable Requirements	Analyze the suitable optimizers for training the model while ensuring there is no degradation in system performance.
Purpose/Scope	To improve the accuracy of the trained model while avoiding the overfitting issue
Item Under Test	Number of natural images of dogs and cats in the dataset
Precautions	The optimizers chosen should not mediate the segmented image by a huge margin but still produce a visually accurate segmentation
Limitations	If there are very little images in the dataset, optimizers would not have much effect on the accuracy of the model
Equipment/Facilities	Laptop
Data Recording	None
Acceptance Criteria	Analyze the suitable optimizer
Procedure	<ol style="list-style-type: none"> 1. Load in the natural images dataset 2. Set the optimizer for the training of the model accordingly 3. Analyze the impact on the performance of the model
Troubleshooting	Iterate this procedure
Post-Test Activities	None

Table 4.3 Verification Plan (Loss function)

Procedure Number	P3
Method	Analysis
Applicable Requirements	Analyze the suitable loss function for training the model while ensuring there is no degradation in system performance.
Purpose/Scope	To improve the accuracy of the trained model while avoiding the overfitting issue
Item Under Test	Number of natural images of dogs and cats in the dataset
Precautions	The loss function should not mediate the segmented image by a huge margin but still produce a visually accurate segmentation.
Limitations	If there are very little images in the dataset, the loss function would not have much effect on the accuracy of the model
Equipment/Facilities	Laptop
Data Recording	None
Acceptance Criteria	Analyze the suitable epoch values
Procedure	<ol style="list-style-type: none"> 1. Load in the natural images dataset 2. Set the loss function for the training of the model accordingly 3. Analyze the impact on the performance of the model
Troubleshooting	Iterate this procedure
Post-Test Activities	None

4.5 Verification Plan (Web Application)

Table 4.4 Verification Plan (Input Image)

Procedure Number	P4
Method	Testing
Applicable Requirements	Detect that the user has input an image and display the image back to the user for user verification.
Purpose/Scope	To improve quality of system
Item Under Test	Image of cat/dog
Precautions	None
Limitations	None
Equipment/Facilities	Laptop
Data Recording	None
Acceptance Criteria	The system should store the user input image in an appropriate local folder and display that image again to the user.
Procedure	<ol style="list-style-type: none"> 1. Input the cat/dog image into the web application 2. Submit the cat/dog image and allow user to display the image 3. Display the recent input image to the user for verification
Troubleshooting	Iterate this procedure
Post-Test Activities	None

Table 4.5 Verification Plan (Segmented Image Production)

Procedure Number	P5
Method	Testing
Applicable Requirements	Detect that the system can output the segmented image of the user's choice
Purpose/Scope	To improve quality of system
Item Under Test	Segmented Image from Segnet/Unet model
Precautions	None
Limitations	None
Equipment/Facilities	Laptop
Data Recording	None
Acceptance Criteria	The system should display either the Segnet segmented image or the Unet segmented image based on the user's choice.
Procedure	<ol style="list-style-type: none"> 1. Input the cat/dog image into the web application 2. Submit the cat/dog image and allow user to display the image 3. Display the recent input image to the user for verification 4. Allow for user to choose between Segnet or Unet model to display the segmented image result 5. Display the segmented image result based on user's choice
Troubleshooting	Iterate this procedure
Post-Test Activities	None

Table 4.6 Verification Plan (Google Colab Connectivity)

Procedure Number	P6
Method	Testing
Applicable Requirements	Detect that the system can redirect the user to the Google Colab implementation for the model of their choice
Purpose/Scope	To improve quality of system
Item Under Test	Colab redirection capability
Precautions	None
Limitations	None
Equipment/Facilities	Laptop
Data Recording	None
Acceptance Criteria	The system should redirect the user to the Google Colab implementation of the model based on the users input
Procedure	<ol style="list-style-type: none"> 1. Input the cat/dog image into the web application 2. Submit the cat/dog image and allow user to display the image 3. Display the recent input image to the user for verification 4. Allow for user to choose between Segnet or Unet model to display the segmented image result 5. Display the segmented image result based on user's choice 6. Redirect the user to the Google Colab implementation if user decides to do so
Troubleshooting	Iterate this procedure
Post-Test Activities	None

Chapter 5

System Implementation

5.1 Start

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import tensorflow_datasets as tfds
import matplotlib.pyplot as plt
import numpy as np
```

Figure 5.1 Importing the Necessary Libraries

Figure 5.1 above shows the method in implementing the necessary libraries. Before the model creation, all the necessary libraries will have to be first imported. Majority of the libraries are from tensorflow for model building and these packages will be used for hyperparameters, data loading, splitting, importing existing datasets from tensorflow and segmentation visualizations for the natural images.

5.2 Data Preprocessing

The formatting of the data overall had no issues as the dataset used is well established but there was some precaution taken with the respective true mask image for each input image. The true mask was in the form of a trimap meaning that there were only three pixel values to represent the “pet”, “background” and “border” of the true mask. The pixel values were 0,1 and 2 respectively therefore it looked black because it only contained pixel values ranging from 0-2 on a scale of 0-255. Therefore, some normalization on the true mask trimap image was required in order to be able to view the true mask properly. Figure 5.2 below shows the function implemented to normalize the trimaps true mask image.

```
def normalize(input_image, input_mask):
    input_image = tf.cast(input_image, tf.float32) / 255.0
    input_mask -= 1
    return input_image, input_mask
```

Figure 5.2 Function to normalize true mask

5.3 Data Loading

```
dataset, info = tfds.load('oxford_iit_pet:3.*.*', with_info=True)

Downloading and preparing dataset 773.52 MiB (download: 773.52 MiB, generated: 774.69 MiB, total: 1.51 GiB)
DL Completed...: 100% [██████████] 2/2 [00:46<00:00, 22.59s/ url]
DL Size....: 100% [██████████] 773/773 [00:46<00:00, 20.37 MiB/s]
Extraction completed...: 100% [██████████] 2/2 [00:46<00:00, 25.77s/ file]
```

Figure 5.3 Loading the dataset from tensorflow

Figure 5.3 above shows the process of importing the dataset. The dataset used for this project (Oxford_IIT_Pet) is already existing within tensorflow datasets therefore it can be loaded directly from tensorflow's function [11]. Once downloaded, it can be observed that the dataset is around 800MB in size.

5.4 Data Splitting

Figure 5.4 below explains the methods implemented to split the data according to the pre-determined ration. Once all the images have been loaded and stored into the dataset variable, it can be split into training, validation and testing set respectively. The splitting is done similarly for both the pretrained model and the model from scratch. The dataset already contains the recommended amount of training and test splits, therefore the usage of the existing splits can take place.

```
TRAIN_LENGTH = info.splits['train'].num_examples
BATCH_SIZE = 64
BUFFER_SIZE = 1000
STEPS_PER_EPOCH = TRAIN_LENGTH // BATCH_SIZE

train_images = dataset['train'].map(load_image, num_parallel_calls=tf.data.AUTOTUNE)
test_images = dataset['test'].map(load_image, num_parallel_calls=tf.data.AUTOTUNE)
```

Figure 5.4 Data Splitting according to the pre-determined ratio

5.5 Data Augmentation

Data augmentation is an essential step in any machine learning approach because it addresses the problem of model overfitting by introducing challenging training samples and artificially inflating the number of training samples the model can use to train [14].

```

class Augment(tf.keras.layers.Layer):
    def __init__(self, seed=42):
        super().__init__()
        # both use the same seed, so they'll make the same random changes.
        self.augment_inputs = tf.keras.layers.RandomFlip(mode="horizontal", seed=seed)
        self.augment_labels = tf.keras.layers.RandomFlip(mode="horizontal", seed=seed)

    def call(self, inputs, labels):
        inputs = self.augment_inputs(inputs)
        labels = self.augment_labels(labels)
        return inputs, labels

```

Figure 5.5 Initialization of data augmentation

The augmentation technique applied currently is a horizontal flip which causes the image to flip horizontally (left side to the right side and vice versa). This augmentation is fairly straight forward but it still brings in an element of realism with the dataset before the model is trained. Figure 5.5 above shows the function created to augment the images in the dataset.

5.6 Hyperparameter set up

```

unet_model.compile(optimizer=tf.keras.optimizers.Adam(),
                    loss="sparse_categorical_crossentropy",
                    metrics="accuracy")

```

Figure 5.6 Hyperparameter set up for model training

Figure 5.6 above shows the final step in setting up the model with its hyperparameters. This step involves setting up the hyperparameters required to train the model. As previously mentioned, hyperparameter fine tuning is essential to ensuring that the model gives the best possible output from training. Adam has been selected as the optimizer for this model. Adam uses stochastic gradient descent to gradually improve the weights in the model to lower the loss attained after each epoch. The sparse categorical crossentropy loss function, which is primarily used to determine how much single sample prediction deviates from its actual labels for multiclass data such as this proposed model where the model labels each pixel as either a background, foreground and boundary class, was chosen to work in tandem with the optimizer.

5.7 Unet Model from Scratch

```
def double_conv_block(x, n_filters):
    # Conv2D then ReLU activation
    x = layers.Conv2D(n_filters, 3, padding = "same", activation = "relu", kernel_initializer = "he_normal")(x)
    # Conv2D then ReLU activation
    x = layers.Conv2D(n_filters, 3, padding = "same", activation = "relu", kernel_initializer = "he_normal")(x)

    return x

def downsample_block(x, n_filters):
    f = double_conv_block(x, n_filters)
    p = layers.MaxPool2D(2)(f)
    p = layers.Dropout(0.3)(p)

    return f, p

def upsample_block(x, conv_features, n_filters):
    # upsample
    x = layers.Conv2DTranspose(n_filters, 3, 2, padding="same")(x)
    # concatenate
    x = layers.concatenate([x, conv_features])
    # dropout
    x = layers.Dropout(0.3)(x)
    # Conv2D twice with ReLU activation
    x = double_conv_block(x, n_filters)

    return x
```

Figure 5.7 Creating the convolutional block (downsample and upsample blocks)

Figure 5.7 above shows the functions created to perform upsampling and downsampling. The model for the segmentation of the natural image of cats and dogs is built from scratch using the implementation of the Unet architecture. It utilizes the conventional number of output channels after each convolution which is 64, 128, 256 and 512. Since there is a large number of pictures in the dataset (7000+), the proposed model can use the recommended amount of output channels. The model will not overfit and will generalize decently for the validation and testing set. The natural images have all been set to the input size (128x128) before they are down sampled after each pooling layer, reducing it by half to obtain a wider view of the model until the fourth convolution of the encoding phase. Then, the up convolutions are used to bring back the original spatial dimension of the images through the decoding phase increasing the size of the natural image back to (128x128). Figure 5.8 and Figure 5.9 below shows the implementation of the Unet model from scratch.

```

# inputs
inputs = layers.Input(shape=(128,128,3))

# encoder: contracting path - downsample
# 1 - downsample
f1, p1 = downsample_block(inputs, 64)
# 2 - downsample
f2, p2 = downsample_block(p1, 128)
# 3 - downsample
f3, p3 = downsample_block(p2, 256)
# 4 - downsample
f4, p4 = downsample_block(p3, 512)

# 5 - bottleneck
bottleneck = double_conv_block(p4, 1024)

# decoder: expanding path - upsample
# 6 - upsample
u6 = upsample_block(bottleneck, f4, 512)
# 7 - upsample
u7 = upsample_block(u6, f3, 256)
# 8 - upsample
u8 = upsample_block(u7, f2, 128)
# 9 - upsample
u9 = upsample_block(u8, f1, 64)

# outputs
outputs = layers.Conv2D(3, 1, padding="same", activation = "softmax")(u9)

# unet model with Keras Functional API
unet_model = tf.keras.Model(inputs, outputs, name="U-Net")

```

Figure 5.8 Finalizing the dimensions of the blocks

5.8 Pretrained Unet Model Implementation

The method in building the model is similar to the Unet model that was built from scratch but the only difference is that the encoder implemented is a pretrained MobileNetV2 model. The encoder consists of outputs that are specific from intermediate layers in the model. There are other pretrained models which can be implemented but due to certain constraints the MobileNetV2 model has been implemented for now. Figure 5.9 below shows the implementation of the pretrained encoder (MobileNetV2).

```

base_model = tf.keras.applications.MobileNetV2(input_shape=[128, 128, 3], include_top=False)

# Use the activations of these layers
layer_names = [
    'block_1_expand_relu',    # 64x64
    'block_3_expand_relu',    # 32x32
    'block_6_expand_relu',    # 16x16
    'block_13_expand_relu',   # 8x8
    'block_16_project',      # 4x4
]
base_model_outputs = [base_model.get_layer(name).output for name in layer_names]

# Create the feature extraction model
down_stack = tf.keras.Model(inputs=base_model.input, outputs=base_model_outputs)

down_stack.trainable = False

```

Figure 5.9 Implementation of the pretrained encoder (MobileNetV2)

5.9 Segnet Model from Scratch

```
from keras.models import Model
from keras.layers import Input, Conv2D, MaxPooling2D, UpSampling2D
```

Figure 5.10 Importing the convolutional layers that exists within Keras libraries

```
inputs = layers.Input(shape=(128,128,3))

# Encoder
conv1 = Conv2D(64, (3, 3), activation='relu', padding='same')(inputs)
pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
conv2 = Conv2D(128, (3, 3), activation='relu', padding='same')(pool1)
pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
conv3 = Conv2D(256, (3, 3), activation='relu', padding='same')(pool2)
pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)
conv4 = Conv2D(512, (3, 3), activation='relu', padding='same')(pool3)

# Decoder
conv5 = Conv2D(512, (3, 3), activation='relu', padding='same')(conv4)
up6 = UpSampling2D(size=(2, 2))(conv5)
conv6 = Conv2D(256, (3, 3), activation='relu', padding='same')(up6)
up7 = UpSampling2D(size=(2, 2))(conv6)
conv7 = Conv2D(128, (3, 3), activation='relu', padding='same')(up7)
up8 = UpSampling2D(size=(2, 2))(conv7)
conv8 = Conv2D(64, (3, 3), activation='relu', padding='same')(up8)

outputs = layers.Conv2D(3, 1, padding="same", activation='softmax')(conv8)
```

Figure 5.11 Defining the Segnet model using the convolutional layers from Keras

The model for the segmentation of the natural image of cats and dogs is built from scratch using the implementation of the Segnet architecture. It utilizes the conventional number of output channels after each convolution which is 64, 128, 256 and 512. Since there is a large number of pictures in the dataset (7000+), the proposed model can use the recommended amount of output channels. The model will not overfit and will generalize decently for the validation and testing set. The natural images have all been set to the input size (128x128) before they are down sampled after each pooling layer, reducing it by half to obtain a wider view of the model until the fourth convolution of the encoding phase. Then, the up convolutions are used to bring back the original spatial dimension of the images through the decoding phase increasing the size of the natural image back to 128x128. The Segnet approach is unique in comparison to the Unet approach because the usage of the max-pooling indices which are recorded during the stage of encoding and later it is used in the decoding stage to perform upsampling efficiently. This will help preserve the spatial information in the map of the

segmentation and reduces artifacts that could probably arise from using traditional upsampling methods. Figure 5.10 above shows how the convolutional layers being imported and Figure 5.11 above shows how the Segnet model is defined from the imported convolutional layers.

5.10 Model Training and Validation

The results obtained (using 20 epochs) from the Unet model from scratch, pretrained Unet model and Segnet model are illustrated in the Figure 5.12, Figure 5.13, and Figure 5.14 below:

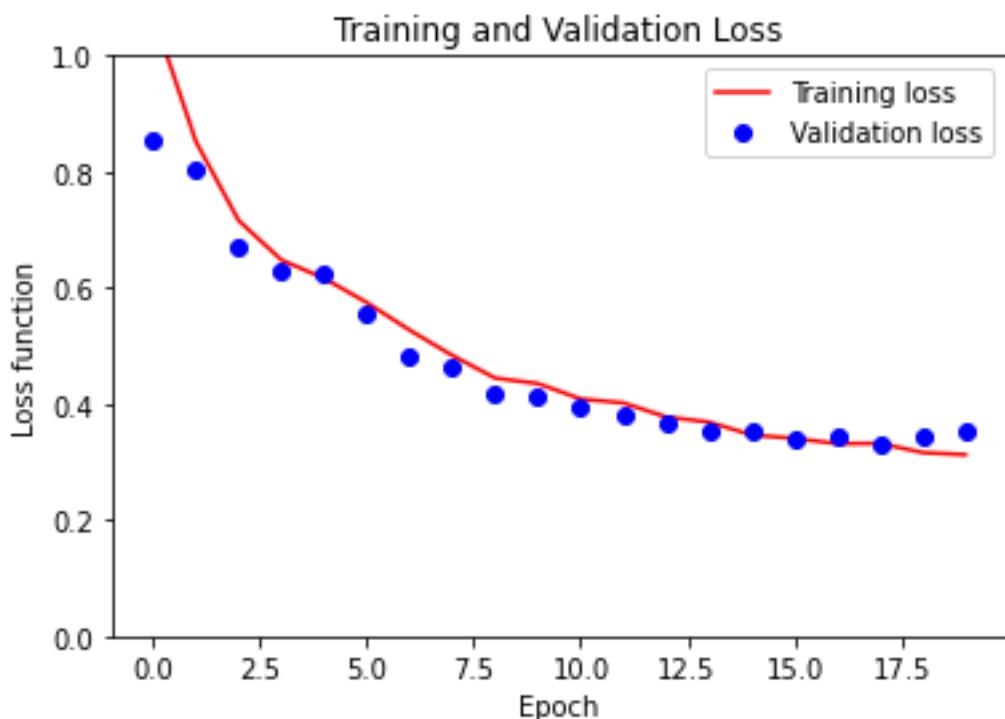


Figure 5.12 Training and Validation loss (Unet from Scratch)

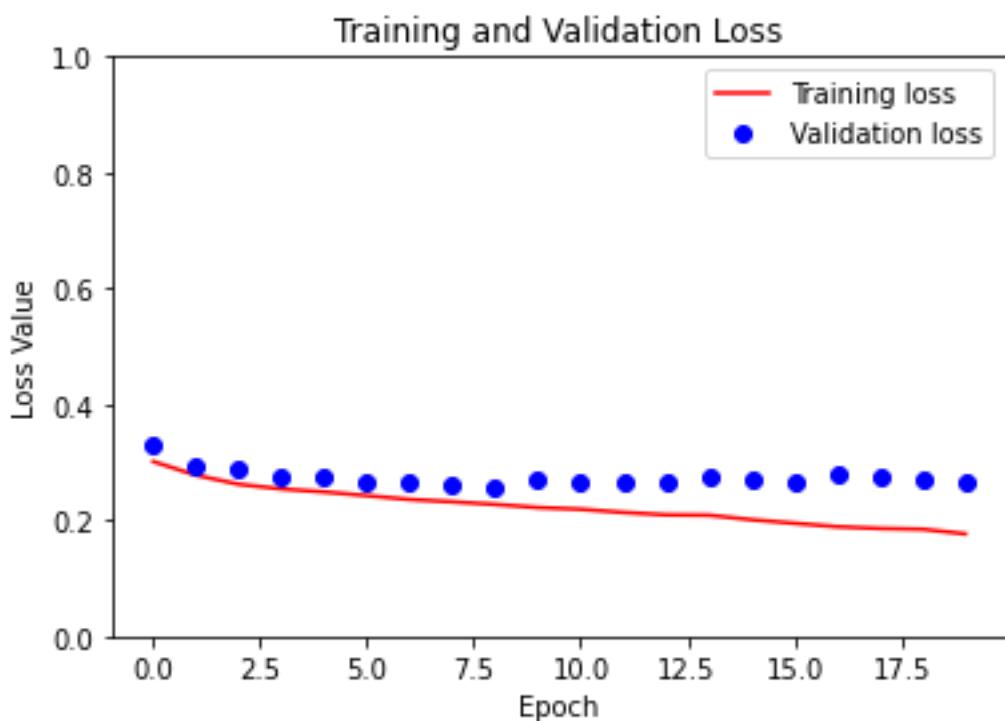


Figure 5.13 Training and Validation loss (Pretrained Unet model)

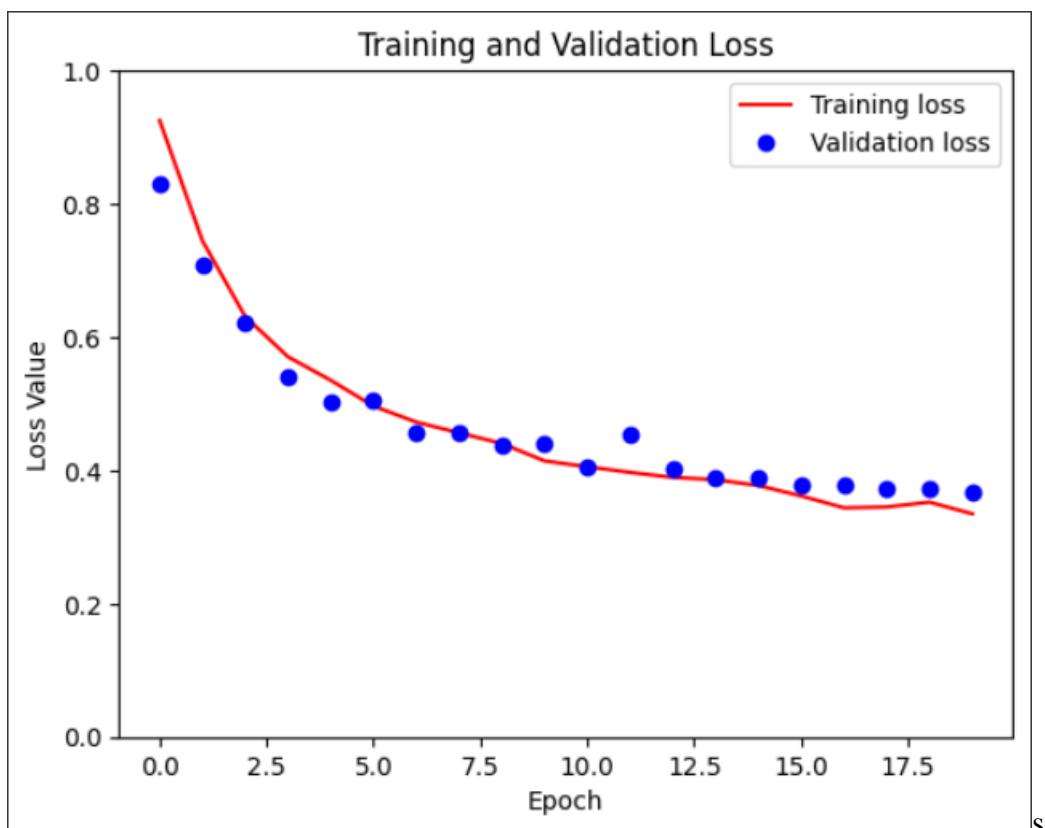


Figure 5.14 Training and Validation loss (Segnet model)

There are differences in results for all three models implemented as it can be observed from Figure 5.12, Figure 5.13 and Figure 5.14 above. For the Unet model from scratch, balance is achieved around epoch 18 with a validation loss of around 0.29 and a few more epochs are required to produce a model that is reliant. Whereas for the pretrained model, balance is achieved around epoch 5 and a few more epochs indicate that there is a huge improvement in validation loss however there might be some overfitting elements. The pretrained model has been exposed to images of similar types previously which is why it does not require as much training in comparison to the Unet model from scratch. Finally, the Segnet model achieves balance around epoch 16 with a validation loss of 0.3 indicating there could be some room for additional epoch values.

5.11 Deploying built models as a web application using Flask framework

After deciding on using of the Unet and Segnet model for the web application implementation, both the models are then transferred over to Visual Studio Code where the web application is built. Before applying the usage of the models into Visual Studio Code from Google Colab, it is important to ensure that the homepage is set up where the web application would route to on default and this would be with the help of `app.route("/")` in Flask which allows the app homepage to receive ‘POST’ request from users when they decide to input an image of their choice. Figure 5.15 below shows the implementation of this method.

```

142 @app.route('/')
143 def index():
144     | | return render_template('index_upload_and_display_image.html')
145
146 @app.route('/', methods=("POST", "GET"))
147 def uploadFile():
148     if request.method == 'POST':
149         # Upload file flask
150         uploaded_img = request.files['uploaded-file']
151         # Extracting uploaded data file name
152         img_filename = secure_filename(uploaded_img.filename)
153         # Upload file to database (defined uploaded folder in static path)
154         uploaded_img.save(os.path.join(app.config['UPLOAD_FOLDER'], img_filename))
155         # Storing uploaded file path in flask session
156         session['uploaded_img_file_path'] = os.path.join(app.config['UPLOAD_FOLDER'], img_filename)
157
158     return render_template('index_upload_and_display_image_page2.html')
```

Figure 5.15 Default app route using Flask

Images with the .png format is accepted and both the weights of Segnet and Unet is then loaded so that they can perform their segmentations respectively. Figure 5.16 and Figure 5.17 below

show the weights of the model being loaded into and the function as to how the models perform their predictions.

```
#Loading the trained weights from Google Colab into the model defined in Flask (unet)
unet_model.load_weights("C:\\Users\\Vishal\\Flask Project\\Unet_Testing.h5")
```

Figure 5.16 Loading in the model weight

```
167     #function to return Unet Predicted Image
168     @app.route('/predicted_image')
169     def showPredictedImage():
170         image = session.get('uploaded_img_file_path', None)
171         image1 = preprocess_image(image)
172         image2 = unet_model.predict(image1)
173         image3 = create_mask(image2)
174         image_filename = "predicted_image.jpeg"
175         plt.savefig(os.path.join(app.config['UPLOAD_FOLDER'],image_filename))
176
177     return render_template('predicted_image.html')
```

Figure 5.17 Function implementation to segment user image

5.12 Cats and Dogs Image Segmentation produced from the web application

Figure 5.18 and Figure 5.19 below show the homepage of the web application and how the user chooses to upload an image file of their choice into the web application. The User Interface (UI) was built using Flask and the UI is presented in a way that requires minimal intervention from the user to perform the image segmentation.

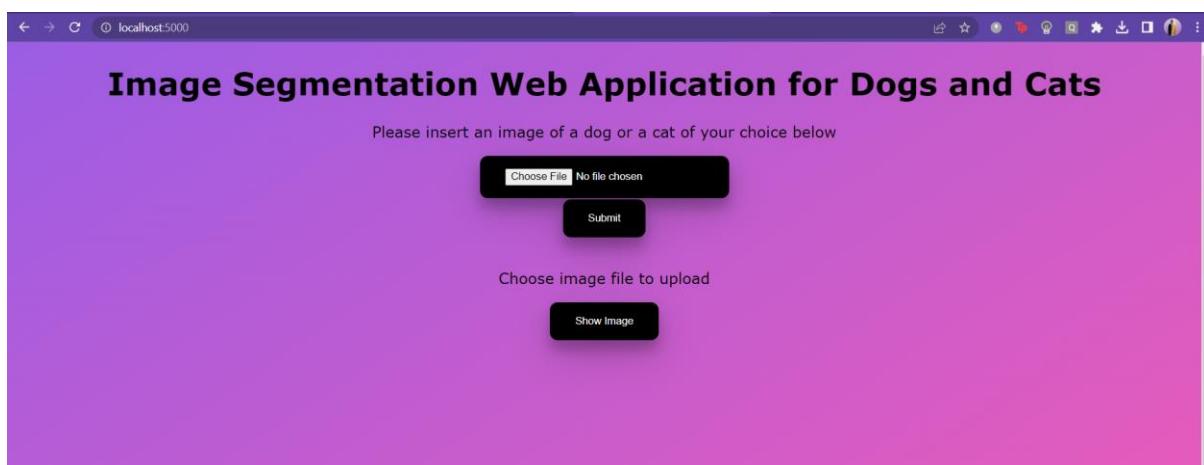


Figure 5.18 Homepage of the web application hosted on localhost IP address.



Figure 5.19 User chooses and uploads an image of a cat for segmentation.

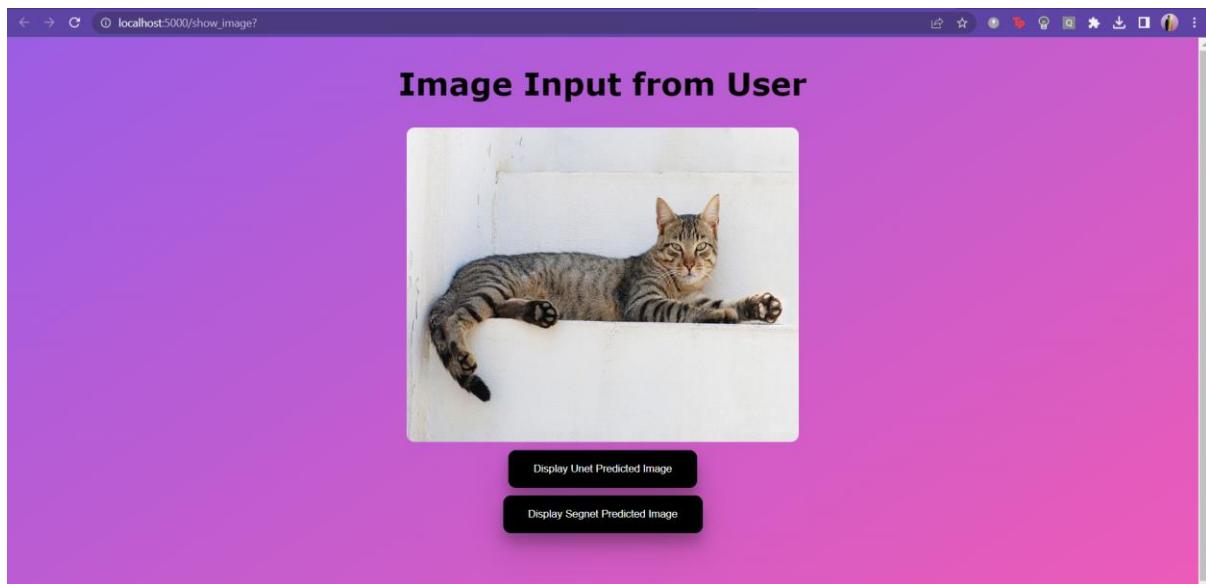


Figure 5.20 Display the input image to user for verification.

Figure 5.20 above shows the page that shows the user the image that they have input and allows for user to verify and ensure that this is indeed the image that they inputted earlier. Once the verification has been completed, users then have the choice to either present the Unet segmented image or the Segnet segmented image. Users will then be redirected to the pages that show the segmented image using the respective model. Figure 5.21 and Figure 5.22 below shows the segmented image output for both models respectively. Users too have the option to

view each implementation of the model on Google Colab. It is assumed that the user is more experienced if they choose to do so.

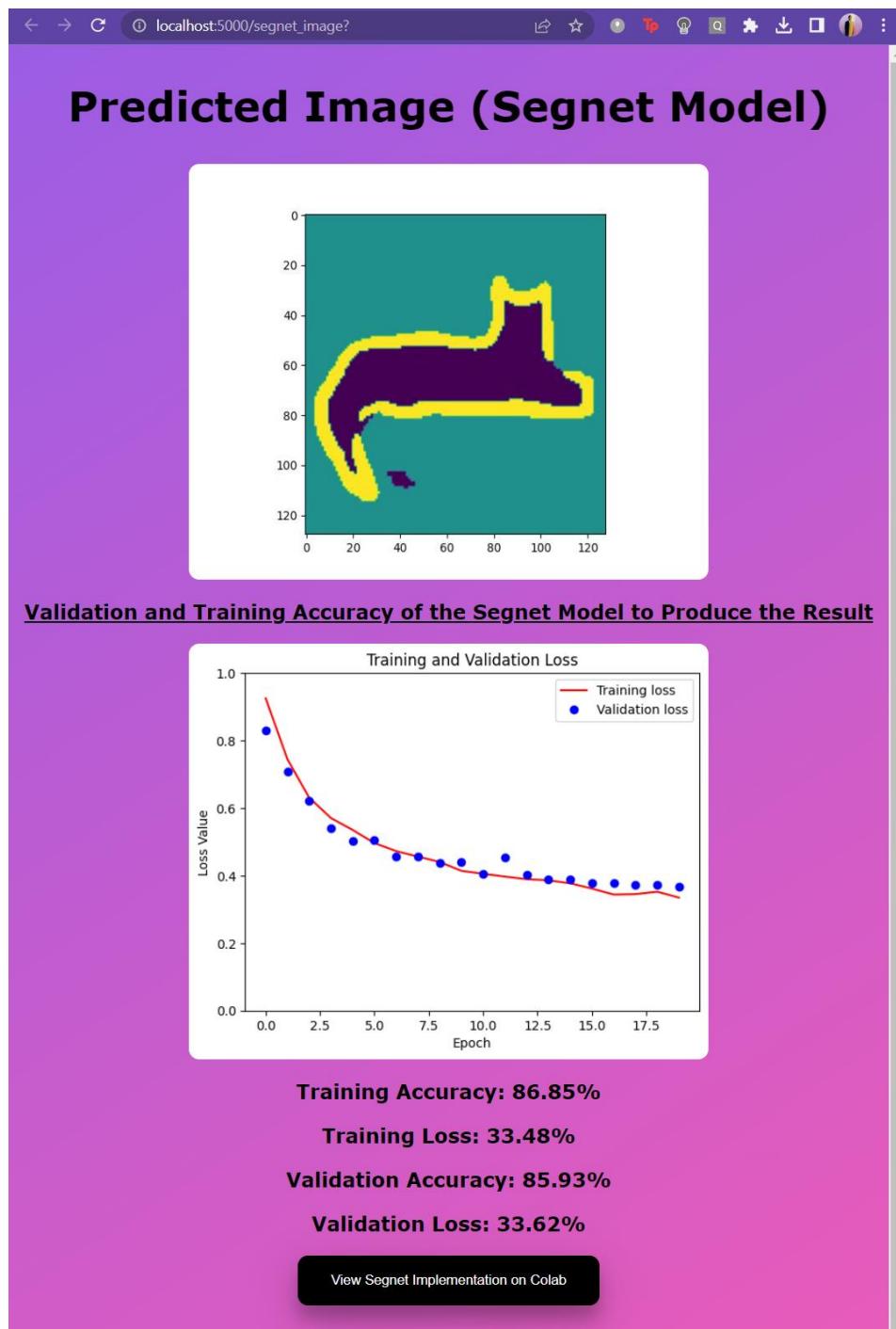


Figure 5.21 Segnet Model output for the input image and its accuracy and loss values

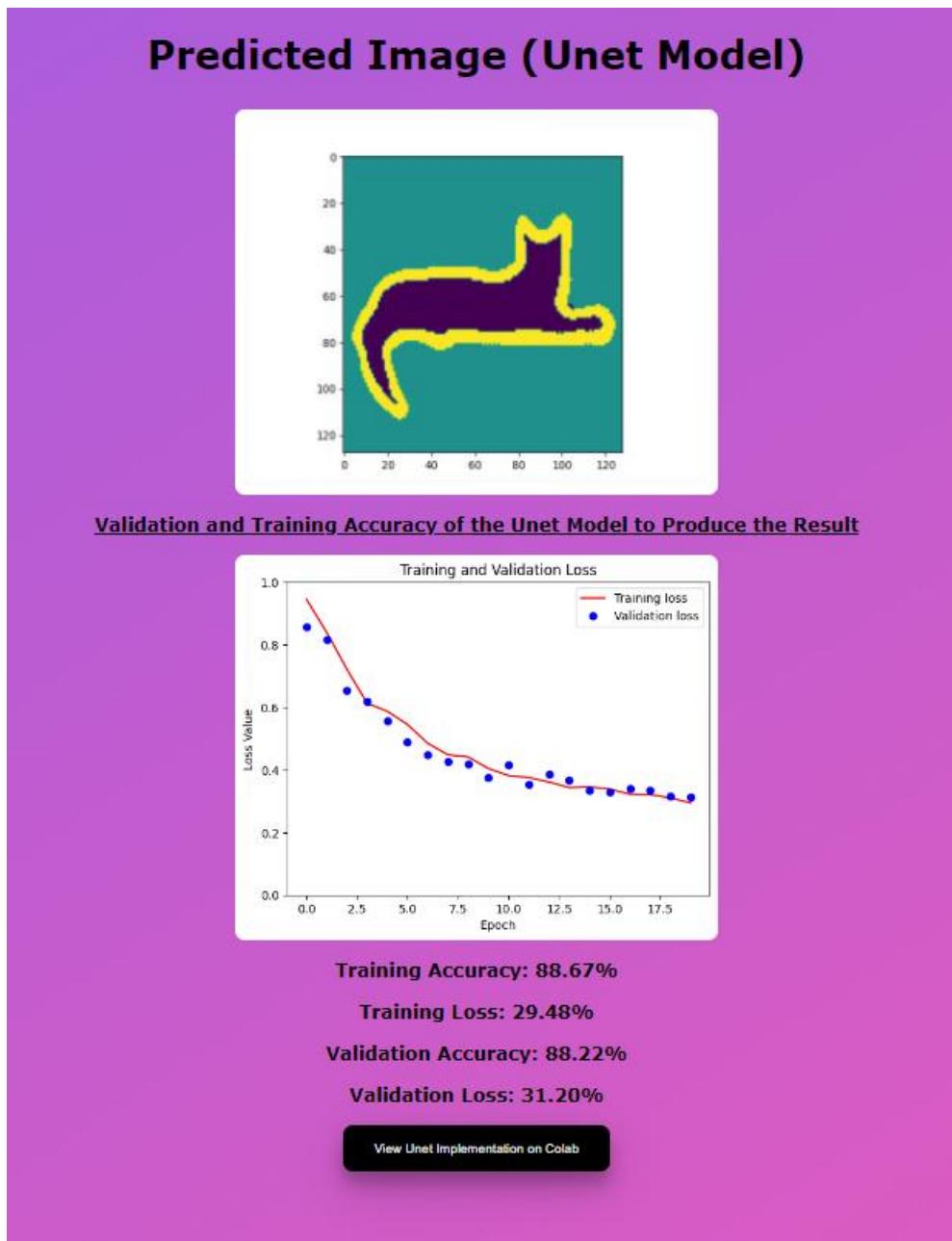


Figure 5.22 Unet Model output for the input image and its accuracy and loss values

5.13 Implementation Issues and Challenges

There are some problems that were encountered while developing the suggested system. The model training is one of the biggest obstacles. A powerful Graphical Processing Unit (GPU) is required because training a CNN model takes a lot of time since the model needs time to learn about the images that are sent to it. By using the Google Colab to train the model rather than a local machine, this problem is partially resolved. The GPU is offered in Google Colab, but this does not address the problem of the local machine's high RAM usage. Additionally, the usage limit in Google Colab without a PRO subscription is set to a maximum Bachelor of Computer Science (Honours)

Faculty of Information and Communication Technology (Kampar Campus), UTAR

of 12 hours and a maximum of 90 minutes of inactivity. To get the desired number of epochs in this situation, the training model must be prepared with the selected images while performing constant checks.

There was also a time constraint in implementing the Unet model and Segnet model from scratch as time was not really favourable especially considering the other courses this semester and it was somewhat mentally demanding to not only learn about implementing the Unet and Segnet model but also making it work with the dataset for this project(Oxford IIT Pet Dataset). There were also periods where the errors encountered took days to solve and that definitely set back the progress but eventually the errors were handled. The pretrained Unet model did not take as much time during the training period but a significant amount of time was required in understanding the concept of transfer learning and implementing the MobileNetV2 encoder blocks for the Unet model. A series of documentation on MobileNetV2 had to be researched upon before executing the implementation of the pretrained Unet model.

Finally with the web application, Flask is a new domain of discovery in implementing the web application therefore time was needed to understand the framework. Once the functionalities of Flask were understood, it also took some time to allow the HTML files to interact with the Flask backend operations. Some of the common issues faced were the scenario where users could input an image but it would not be detected by the models in the flask environment or the models did manage to produce an output but that output could not be translated into the HTML environment. Anyhow, these errors have been solved with time but it was definitely mentally demanding to ensure the implementation of the web application was successful.

Chapter 6

System Evaluation and Discussion

6.1 Testing with the Experimental Models and Hyperparameters

Before loading the trained models for usage within the system, a few approaches were taken to introduce an element of experimentation. The elements for the experimentation were the variety of deep learning models used (Segnet, Unet and pretrained Unet), transfer learning with the usage of a pretrained model, optimizers, epoch values and loss functions. Each of these elements played a part in determining the usage of a model that was accurate and acceptable to produce a segmented image of the dog or cat. In order to test the accuracy of the models with different parameters, some of the parameters were set to a constant variable while the parameter under testing was set as a manipulated variable which can be further understood by observing the Table 6.1, 6.2, 6.2 and 6.4 below.

Table 6.1 Summary experimental models (Model)

No	Model	Epoch Value	Optimizer	Loss Function	Accuracy (%)
1.	Pretrained Unet	20	Adam	SCE	89.68
2.	Unet from scratch	20	Adam	SCE	89.22
3.	Segnet from scratch	20	Adam	SCE	84.87

The constant variables here are the epoch value, optimizers and the loss function. From this experiment, it can be observed that all 3 models have relatively similar accuracy in terms of its nature although the pretrained Unet and the Unet from scratch models tend to perform better in this scenario.

Table 6.2 Summary Experimental Models (Epoch Value and Model Type)

No	Model	Epoch Value	Optimizer	Loss Function	Accuracy (%)
1.	Pretrained Unet	5	Adam	SCE	75.78
2.	Pretrained Unet	15	Adam	SCE	82.45
3.	Pretrained Unet	25	Adam	SCE	88.90
4.	Pretrained Unet	30	Adam	SCE	89.68
5.	Unet from scratch	5	Adam	SCE	68.21

6.	Unet from scratch	15	Adam	SCE	81.02
7.	Unet from scratch	25	Adam	SCE	85.60
8.	Unet from scratch	30	Adam	SCE	89.75

The constant values are the optimizers and loss function. From this experiment it can be observed that the higher the epoch value, the better the accuracy of the model. This is because for every epoch, the model is trained with the training images from the dataset once again therefore inevitably improving accuracy but a proper epoch value must be chosen in order to ensure that there is no model overfitting. On the other hand, the pretrained Unet model tends to perform better at lower epoch values in comparison to the Unet from scratch because the pretrained Unet model implements a concept known as transfer learning meaning that the encoder blocks (MobileNetV2) of the pretrained Unet model has already been trained by another group of data scientist therefore it is expected that the pretrained Unet model will do better in terms of performance even though it has little exposure to the dataset of images whereas the Unet model from scratch has not been exposed to any images so it will require more epoch values before it can provide somewhat accurate results.

Table 6.3 Summary Experimental Models (Optimizer)

No	Model	Epoch Value	Optimizer	Loss Function	Accuracy (%)
1.	Unet from scratch	10	Adagrad	SCE	70.26
2.	Unet from scratch	10	RMSprop	SCE	68.81
3.	Unet from scratch	10	AdamW	SCE	70.02
4.	Unet from scratch	10	Adamax	SCE	69.98
5.	Unet from scratch	10	SGD	SCE	70.04
6.	Unet from scratch	10	Nadam	SCE	68.03
7.	Unet from scratch	10	Adam	SCE	79.89

The constant values here are the model, epoch value and loss function. This experiment serves the purpose to determine the best optimizer for segmenting the image and it is determined based on its validation accuracy. As observed, the optimizer that stands out is the Adam optimizer by providing an accuracy of 79.89% which is a huge margin in comparison to the other optimizers tested. Some of the optimizers implemented are not suitable for the usage of image segmentation due to its nature of optimizing the model (RMSprop and Adagrad).

Table 6.4 Summary Experimental Models (Loss Function)

No	Model	Epoch Value	Optimizer	Loss Function	Accuracy (%)
1.	Unet from scratch	10	Adam	SCE	79.89
2.	Unet from scratch	10	Adam	SC	68.98

The constant values here are the model, epoch value and optimizer. As observed the Sparse Categorical Crossentropy (SCE) loss function performs significantly better than the Sparse Crossentropy (SC) loss function. Therefore, the final hyperparameters chose based on the accuracy from the experimentations were the Adam optimizer, Epoch value 20 and the SCE loss function. Both Segnet and Unet models were used in the web application to give users the freedom of choice to pick either model for segmentation.

6.2 Testing with Testing Set for each model

Observations on the segmented result of the natural image shows that there are scenarios whereby the model does not face any issues in segmenting the region of interest (ROI) but there are instances where there were some difficulties encountered.

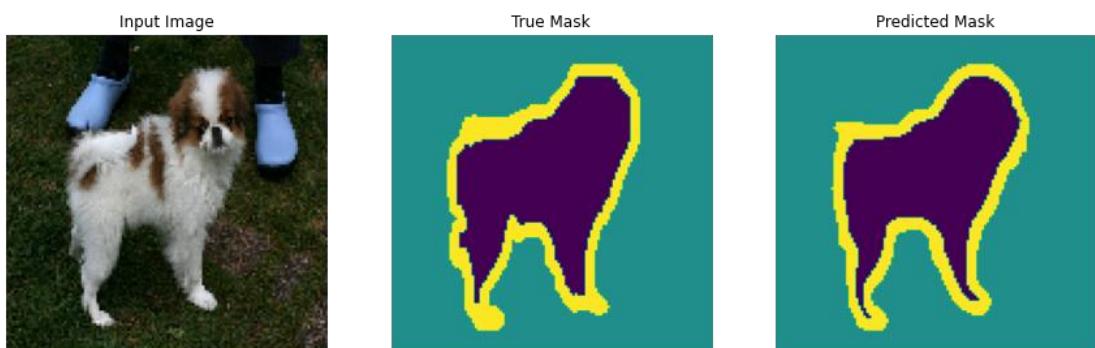


Figure 6.1 Segmentation Result of Natural Oxford IIIT Pet Dataset (Unet from scratch)

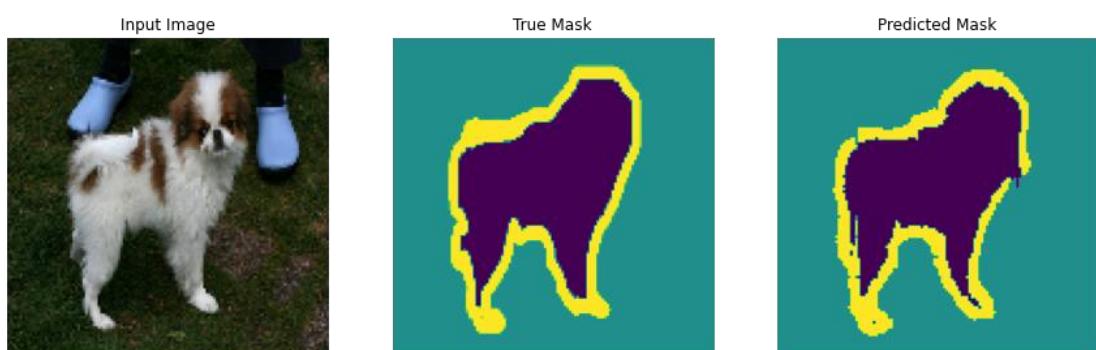


Figure 6.2 Segmentation Result of Natural Oxford IIIT Pet Dataset (pretrained Unet)

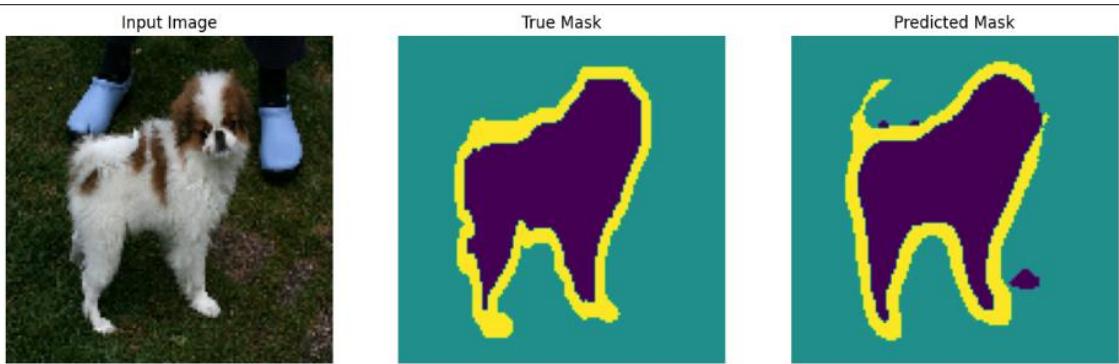


Figure 6.3 Segmentation Result of Natural Oxford IIT Pet Dataset (Segnet from scratch)

Figure 6.1, Figure 6.2 and Figure 6.3 above show the segmentation result of all three models given the fixed epoch value of 30 during training and it can be observed that the models did decently well in segmenting the dog and the final result is within the acceptable range in comparison to the true mask. This dog was relatively easy to segment because it can be easily seen and there is not a lot of distortion that makes it hard for the model to differentiate between the dog and the background. However, there are some instances whereby there is distortion within the input image and the models face some difficulties in segmenting the region of interest accurately.

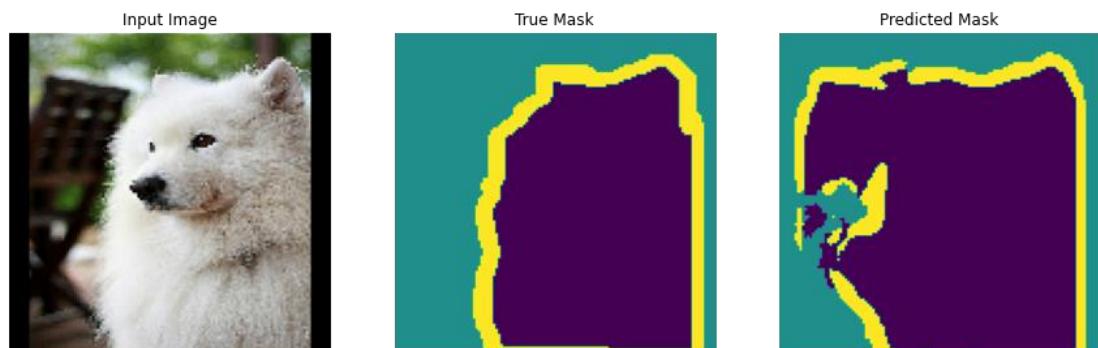


Figure 6.4 Distorted segmentation result (Unet from scratch)

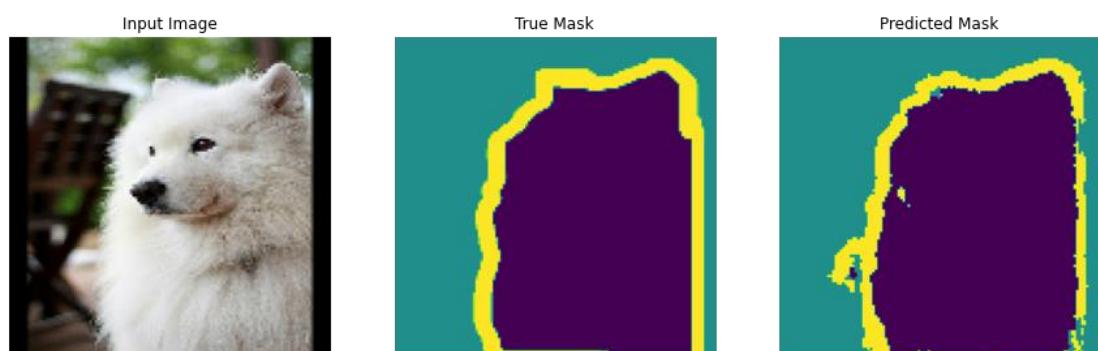


Figure 6.5 Distorted segmentation (pretrained Unet model)

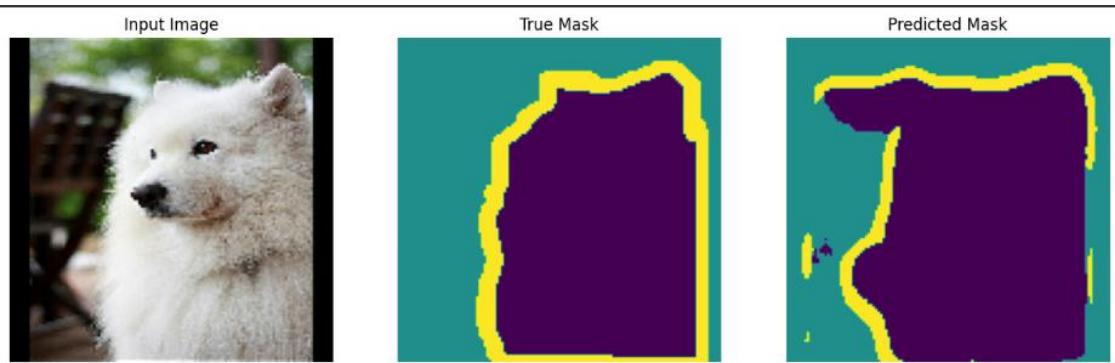


Figure 6.6 Distorted segmentation (Segnet model)

As observed in Figure 6.4, Figure 6.5 and Figure 6.6, sometimes the models are presented with some difficulties within the input image and therefore it cannot segment the region of interest (ROI), accordingly. This image for example has a blurry background therefore the model has some difficulties in separating the blurry background from the clear region of interest. Even though the pretrained model did a somewhat decent job, there is still room for improvement. More segmentation results for both models are shown in Appendix A1.

6.3 Testing the existence of False Positives

The models were all trained and tested with the Oxford IIT Pet Dataset meaning that the dataset consists of images of dogs and cats which all have somewhat similar traits. It is worth remembering that the user is given the freedom to input any image of their choice and it is inevitable that some users may input images that are not of the cat or dog origin. This testing serves the purpose to discover the scenario whereby users input any other image types (rabbit, horse, deer etc) and to observe the output the application provides to the user. If the images that are not of the cat or dog origin can still be segmented somewhat accurately, then this will fall under the false positive category. Figure 6.7, 6.8, 6.9, 6.10, 6.11 and 6.12 below are the screenshots of the relevant screenshots to test the scenario of inputting an image that is not of the cat or dog origin and to observe the outputs.

Image Input from User



Display Unet Predicted Image

Display Segnet Predicted Image

Figure 6.7 Rabbit Input Image

Predicted Image (Segnet Model)

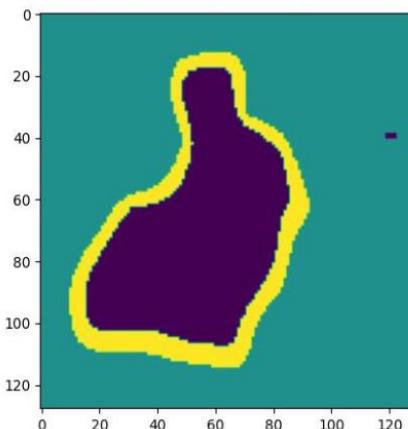


Figure 6.8 Segmented Rabbit Image from Segnet

Predicted Image (Unet Model)

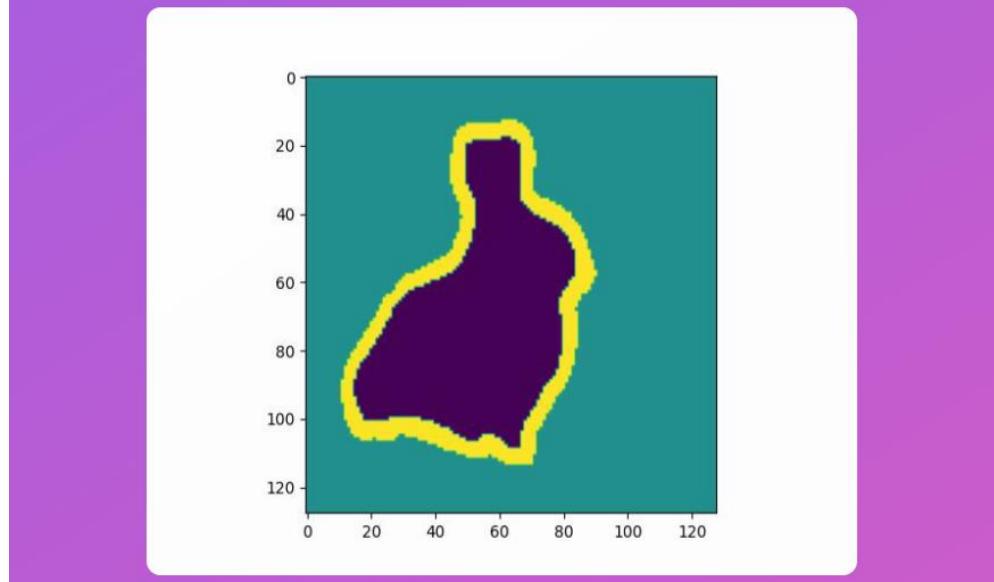
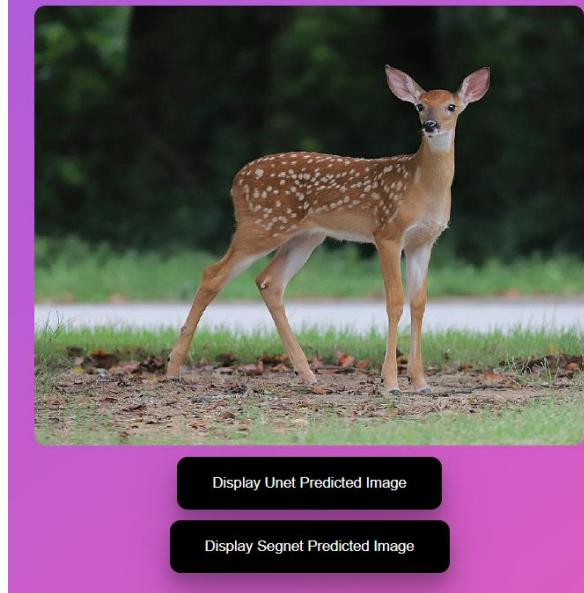


Figure 6.9 Segmented Rabbit Image from Unet

Image Input from User



Display Unet Predicted Image

Display Segnet Predicted Image

Figure 6.10 Deer Input Image

Predicted Image (Segnet Model)

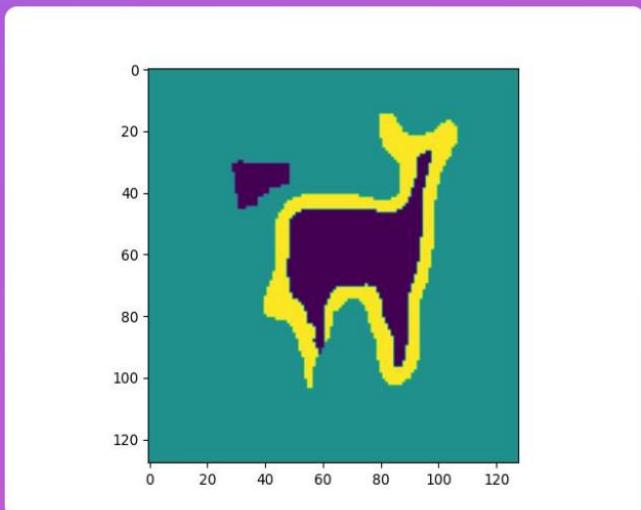


Figure 6.11 Segmented Deer Image from Segnet

Predicted Image (Unet Model)

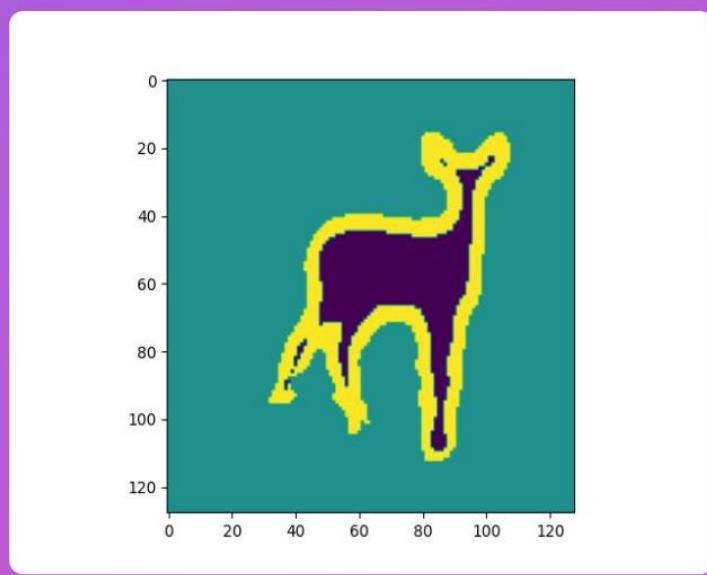


Figure 6.12 Segmented Deer Image from Unet

As observed from the figures above, the models (Unet and Segnet) do indeed segment these images that are not from the cat or dogs origin and this is because these animals share similar traits to cats and dogs. They are both four legged animals and share similar facial features as well therefore the model cannot differentiate the type of animal but rather the features that these

animals have. Other animals that share similar features with the animals above will also obtain similar segmentation results. One way to overcome this is by broadening the domain of acceptability for these models and including a variety of other animals as well therefore allowing it to accurately segment animals other than cats or dogs.

Chapter 7

Conclusion and Recommendation

7.1 Conclusion & Novelty

In conclusion, this work has successfully managed to review three models (Unet, pretrained Unet and Segnet) which can be implemented into an application. Both the Unet and Segnet models have been implemented into the application giving user the freedom to perform segmentation according to their preference of model implementation. Existing image segmentation software implement conventional image segmentation algorithms whereas this project implemented the usage of deep learning techniques which provides users the convenience of minimum intervention.

7.2 Recommendation

There are some recommendations applicable to the context of this project. Mainly, the accuracy of the segmentation can be further improved to higher values via the manipulation of hyperparameter values, implementation of transfer learning techniques and using a dataset that is subjected to more images and with longer epoch values. The application interface can be more user-friendly and intuitive in the future.

REFERENCES

1. M. Tyagi, "Image segmentation : Part 1," *Medium*, 19-Jul-2021. [Online]. Available: <https://towardsdatascience.com/image-segmentation-part-1-9f3db1ac1c50>. [Accessed: 01-Sep-2022].
2. "4SmartMachine," *4SmartMachines*, 2019. [Online]. Available: <https://www.4smartmachines.com/>. [Accessed: 01-Sep-2022].
3. K. McGuinness, "Interactive Segmentation Tool," *Interactive segmentation tool*, 2009. [Online]. Available: <http://web.archive.org/web/20110825021314/http://kspace.cdvp.dcu.ie/public/interactive-segmentation/>. [Accessed: 01-Sep-2022].
4. "Imagej," *ImageJ Wiki*, 2012. [Online]. Available: <https://imagej.net/software/imagej>. [Accessed: 01-Sep-2022].
5. Walsh, Joseph & O' Mahony, Niall & Campbell, Sean & Carvalho, Anderson & Krpalkova, Lenka & Velasco-Hernandez, Gustavo & Harapanahalli, Suman & Riordan, Daniel, "Deep Learning vs. Traditional Computer Vision", 2019 Computer Vision Conference (CVC), 2019, doi: 10.1007/978-3-030-17795-9_10.
6. S. Sivagami, P. Chitra, G. S. R. Kailash and S. R. Muralidharan, "Unet Architecture Based Dental Panoramic Image Segmentation," 2020 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET), 2020, pp. 187-191, doi: 10.1109/WiSPNET48689.2020.9198370.
7. S. Peek, "What is agile scrum methodology?," *Business News Daily*, 2022. [Online]. Available: <https://www.businessnewsdaily.com/4987-what-is-agile-scrum-methodology.html>. [Accessed: 03-Sep-2022].
8. Abràmoff, M.D., Magalhães, P.J. and Ram, S.J., 2004. Image processing with ImageJ. *Biophotonics international*, 11(7), pp.36-42
9. Ferreira, T. and Rasband, W., 2012. ImageJ user guide. *ImageJ/Fiji*, 1, pp.155-161.
10. McGuinness, K. and O'Connor, N.E., 2011. Toward automated evaluation of interactive segmentation. *Computer Vision and Image Understanding*, 115(6), pp.868-884.

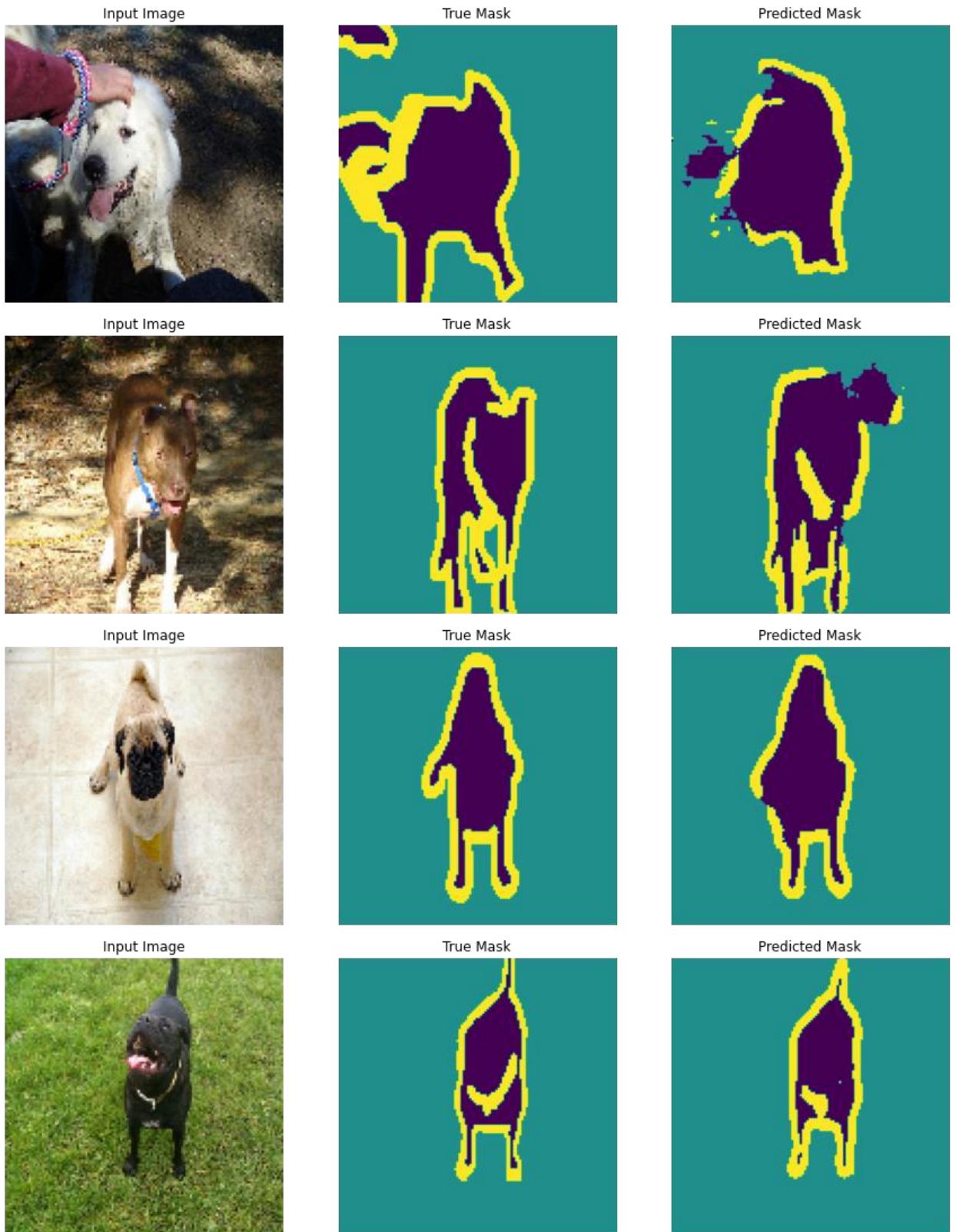
11. O. M. Parkhi, Andrea Vedaldi, A. Zisserman, and C. W. Jawahar, “The Oxford-IIIT Pet Dataset,” *Visual Geometry Group - University of Oxford*, 2012. [Online]. Available: <https://www.robots.ox.ac.uk/~vgg/data/pets/>. [Accessed: 01-Dec-2022].
12. A. Kumar, “Keras - categorical cross entropy loss function,” *Data Analytics*, 28-Oct-2020. [Online]. Available: <https://vitalflux.com/keras-categorical-cross-entropy-loss-function/>. [Accessed: 02-Dec-2022].
13. J. Zhang, “Unet line by line explanation,” *Medium*, 18-Oct-2019. [Online]. Available: <https://towardsdatascience.com/Unet-line-by-line-explanation-9b191c76baf5>. [Accessed: 02-Dec-2022].
14. C. Dilmegani, “What is data augmentation? techniques, examples & benefits,” *AIMultiple*. [Online]. Available: <https://research.aimultiple.com/data-augmentation/>. [Accessed: 02-Dec-2022].
15. Ronneberger, O., Fischer, P. and Brox, T., 2021. “U-Net: Convolutional Networks for Biomedical Image Segmentation.” [online] arXiv.org. Available at: <<https://arxiv.org/abs/1505.04597>> [Accessed 01-Dec 2022]
16. M. Fezan, “Understanding of semantic segmentation & how Segnet model work to perform semantic segmentation,” *Medium*, 24-Oct-2019. [Online]. Available: <https://medium.com/@fezancs/understanding-of-semantic-segmentation-how-Segnet-model-work-to-perform-semantic-segmentation-5c426112e499>. [Accessed: 27-Apr-2023].

APPENDIX

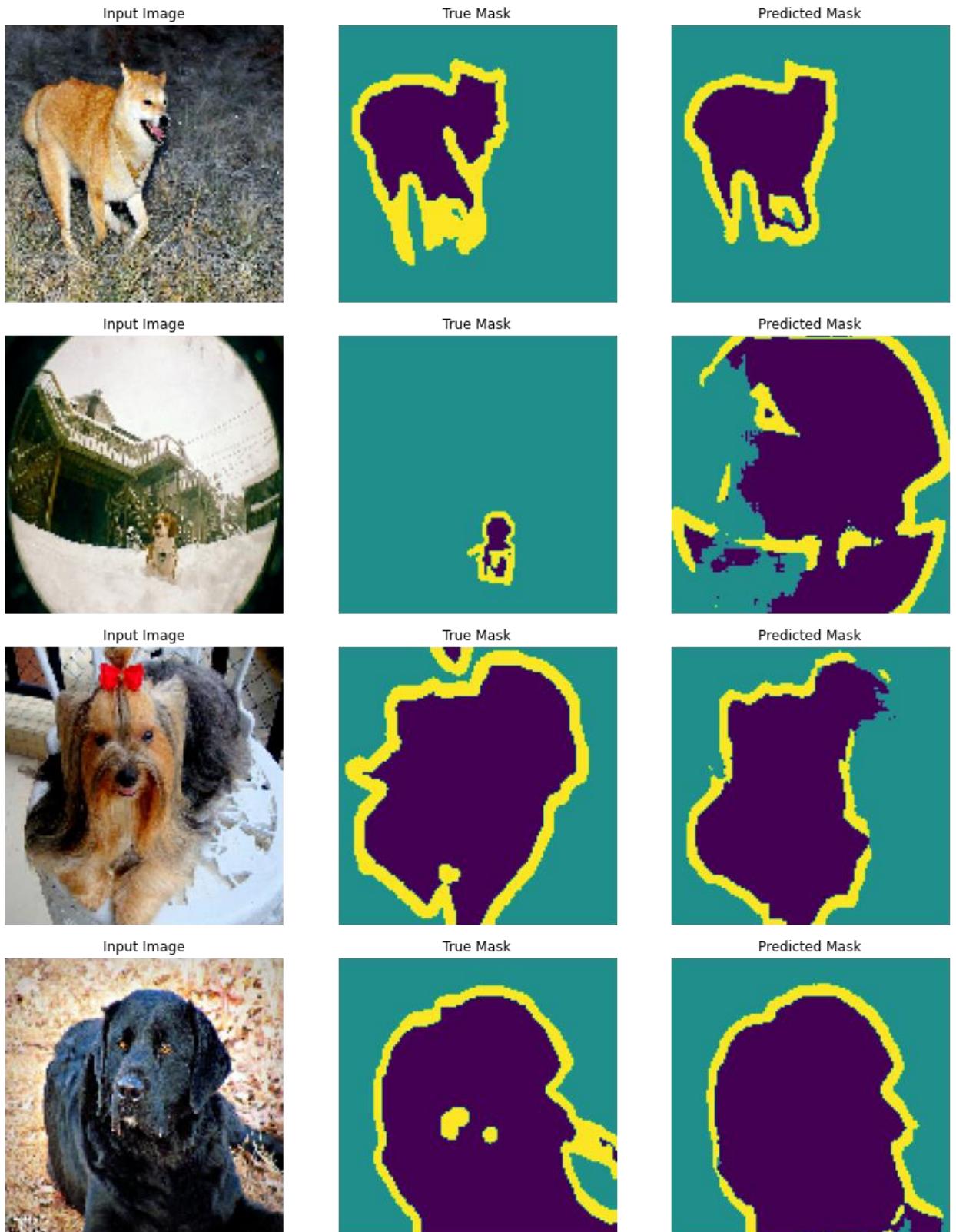
Appendix A1: Segmentation results of both models

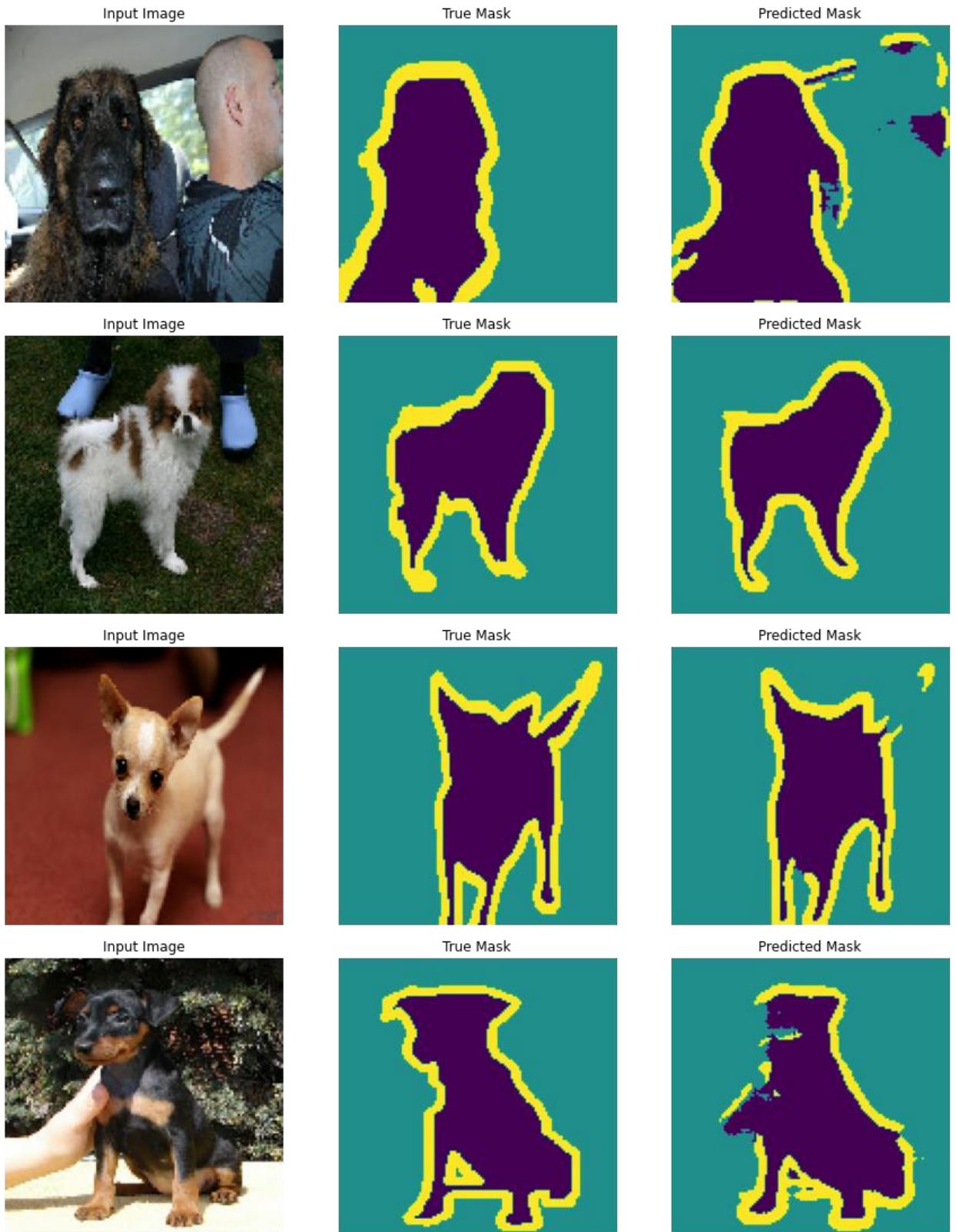
Unet model from scratch



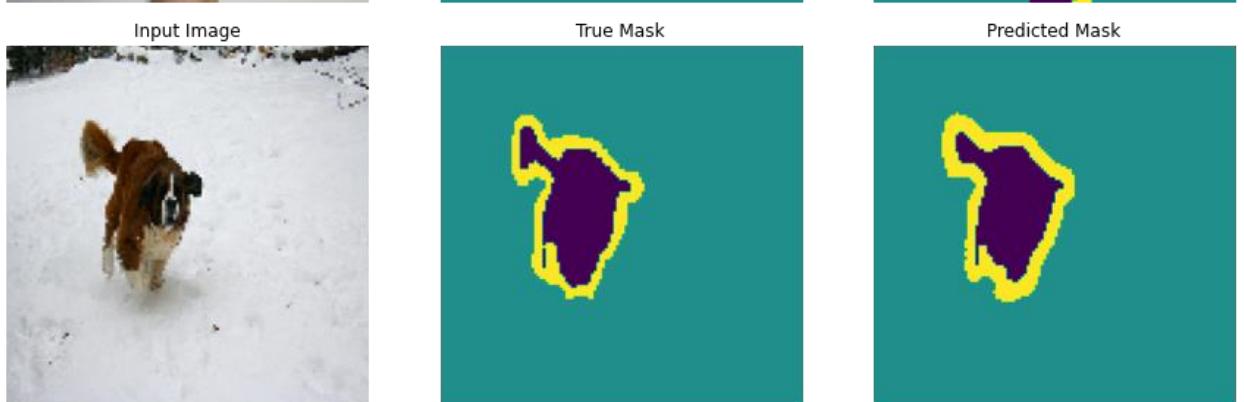
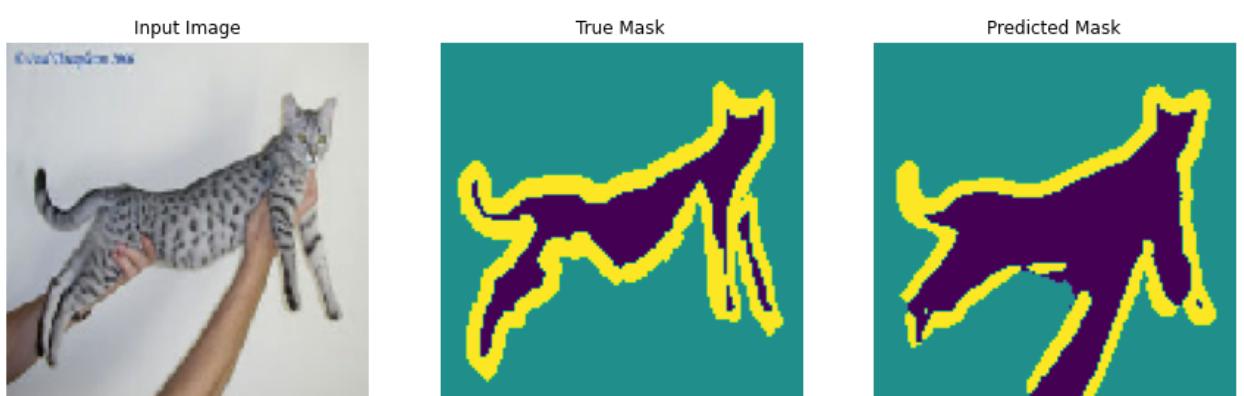
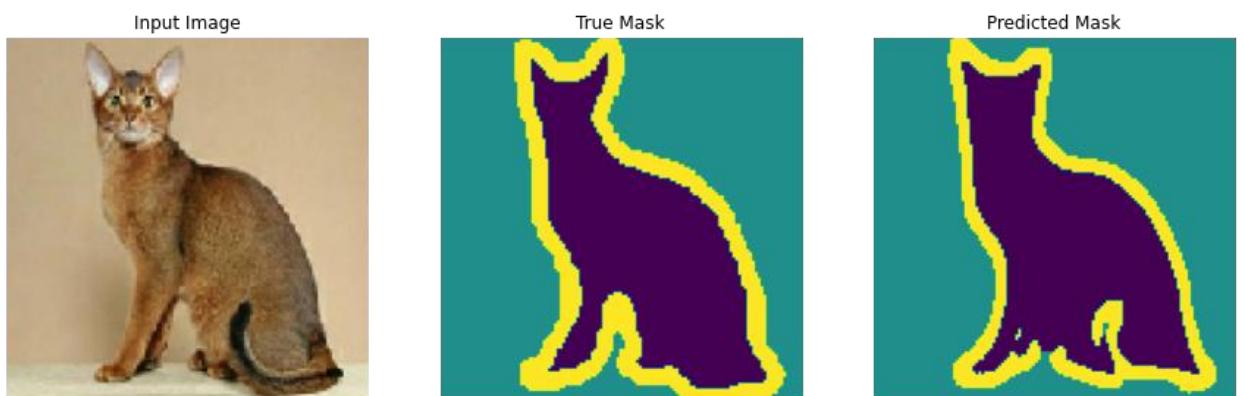


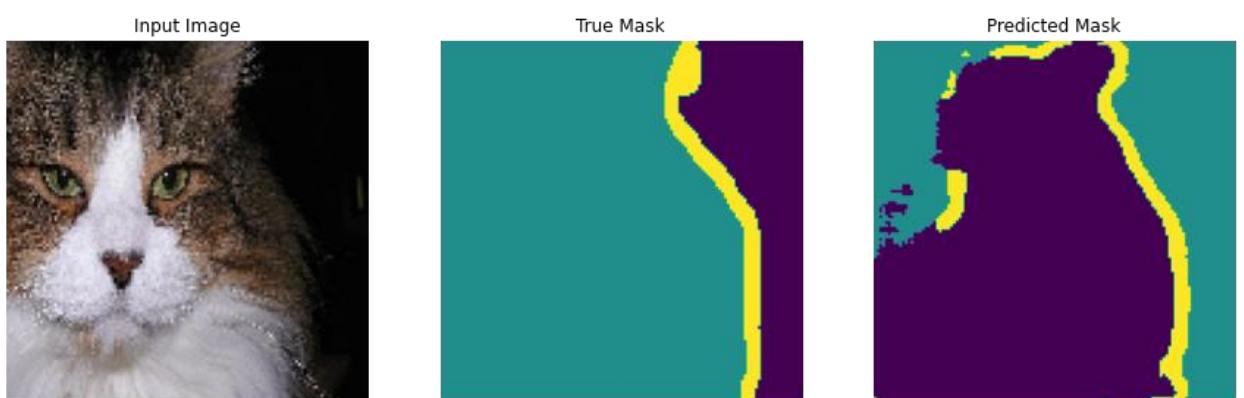


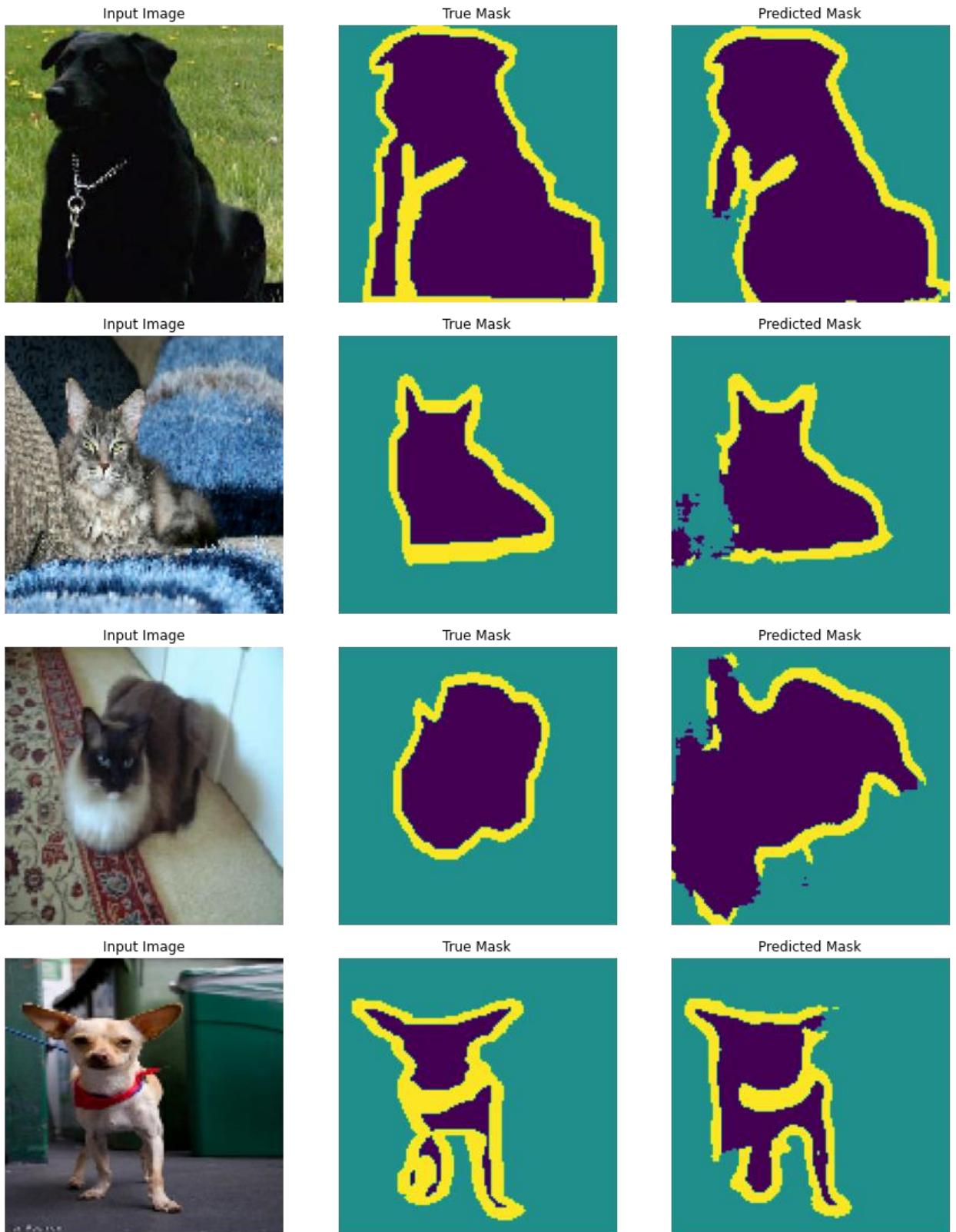




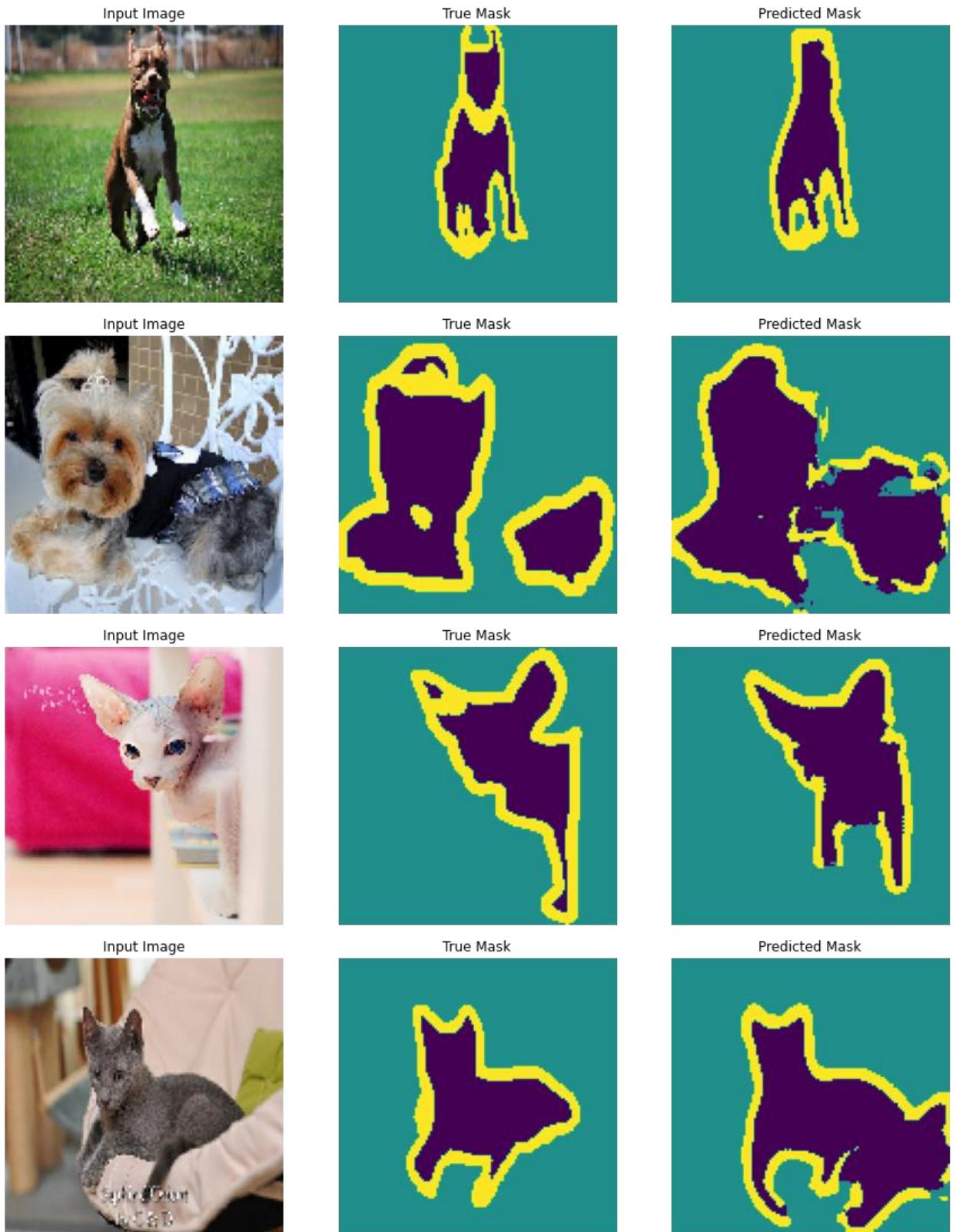


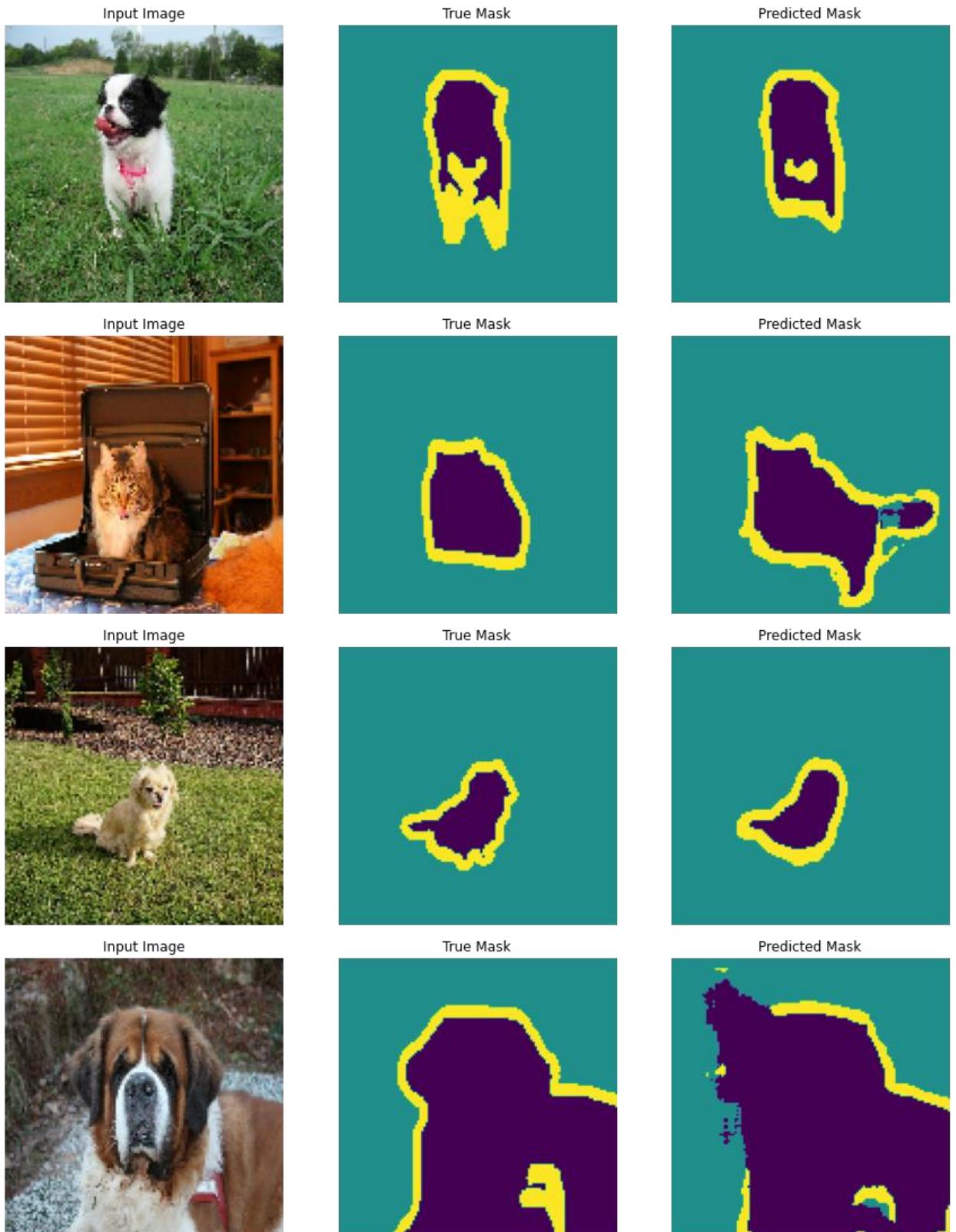


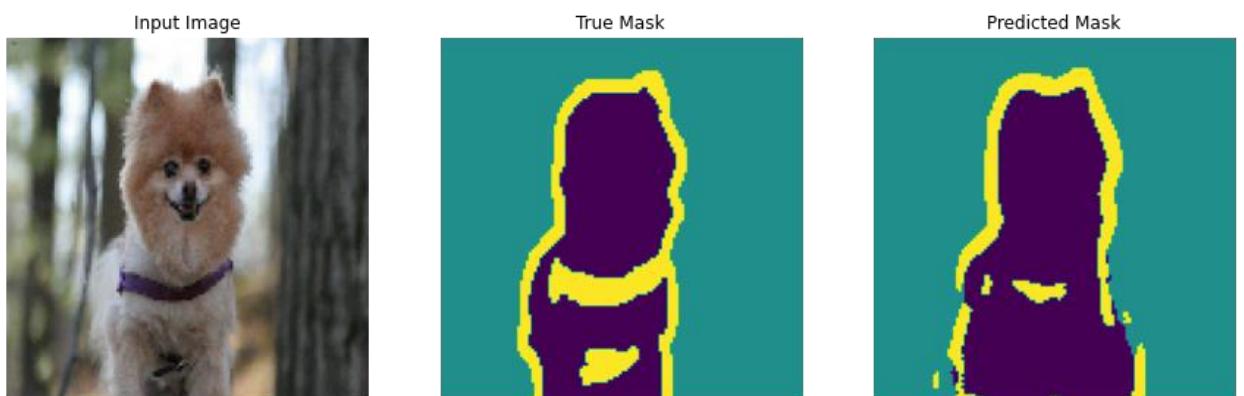


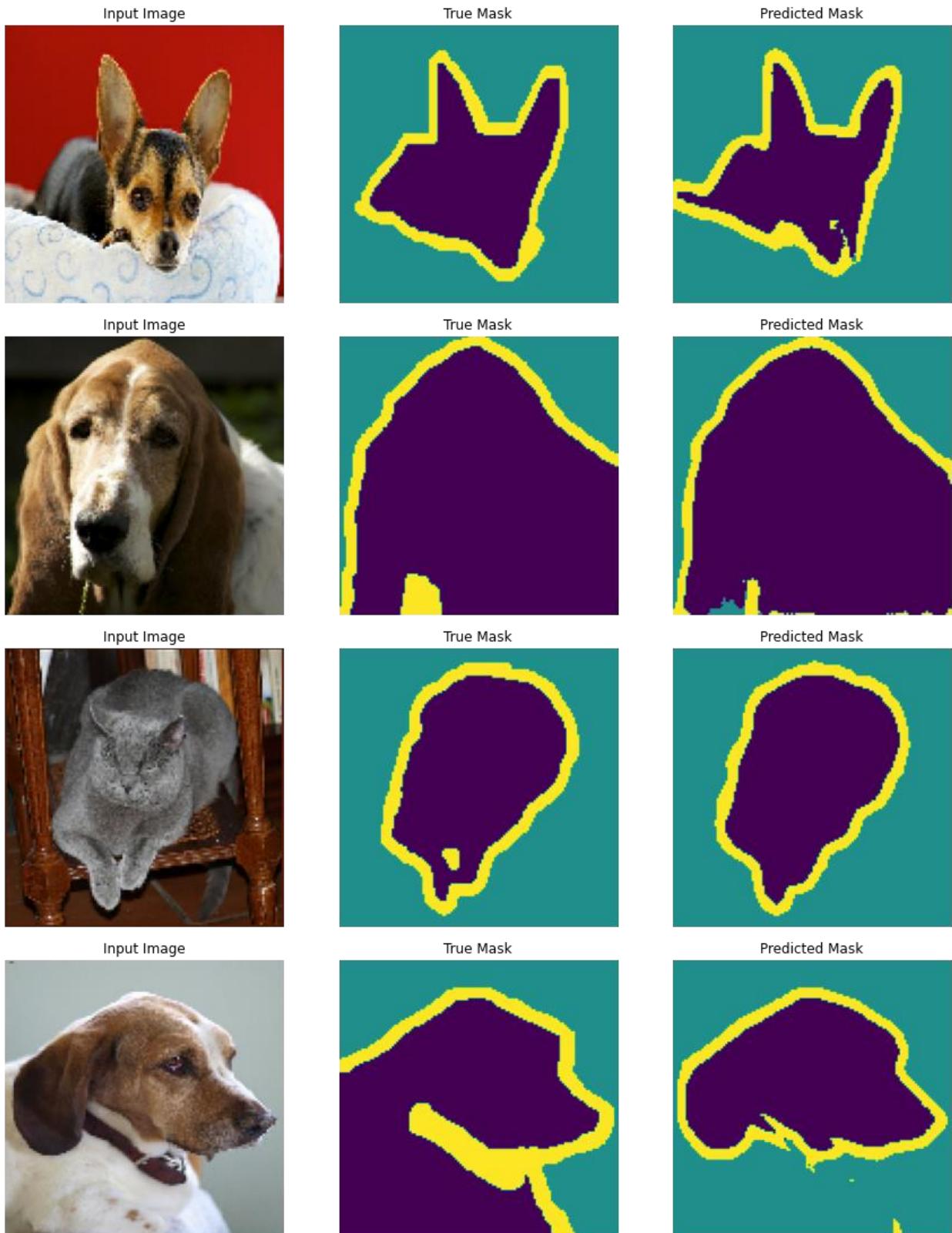


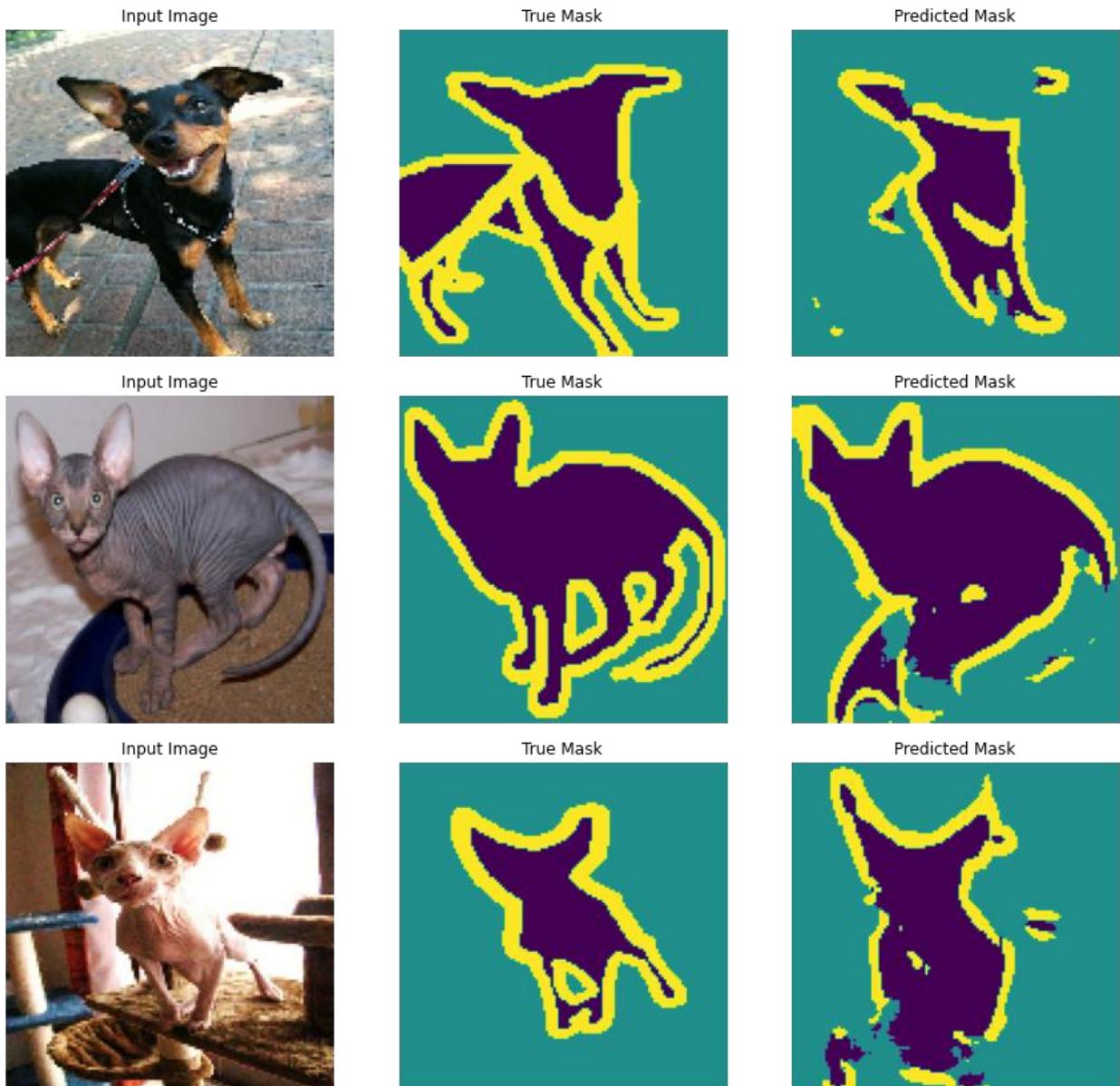






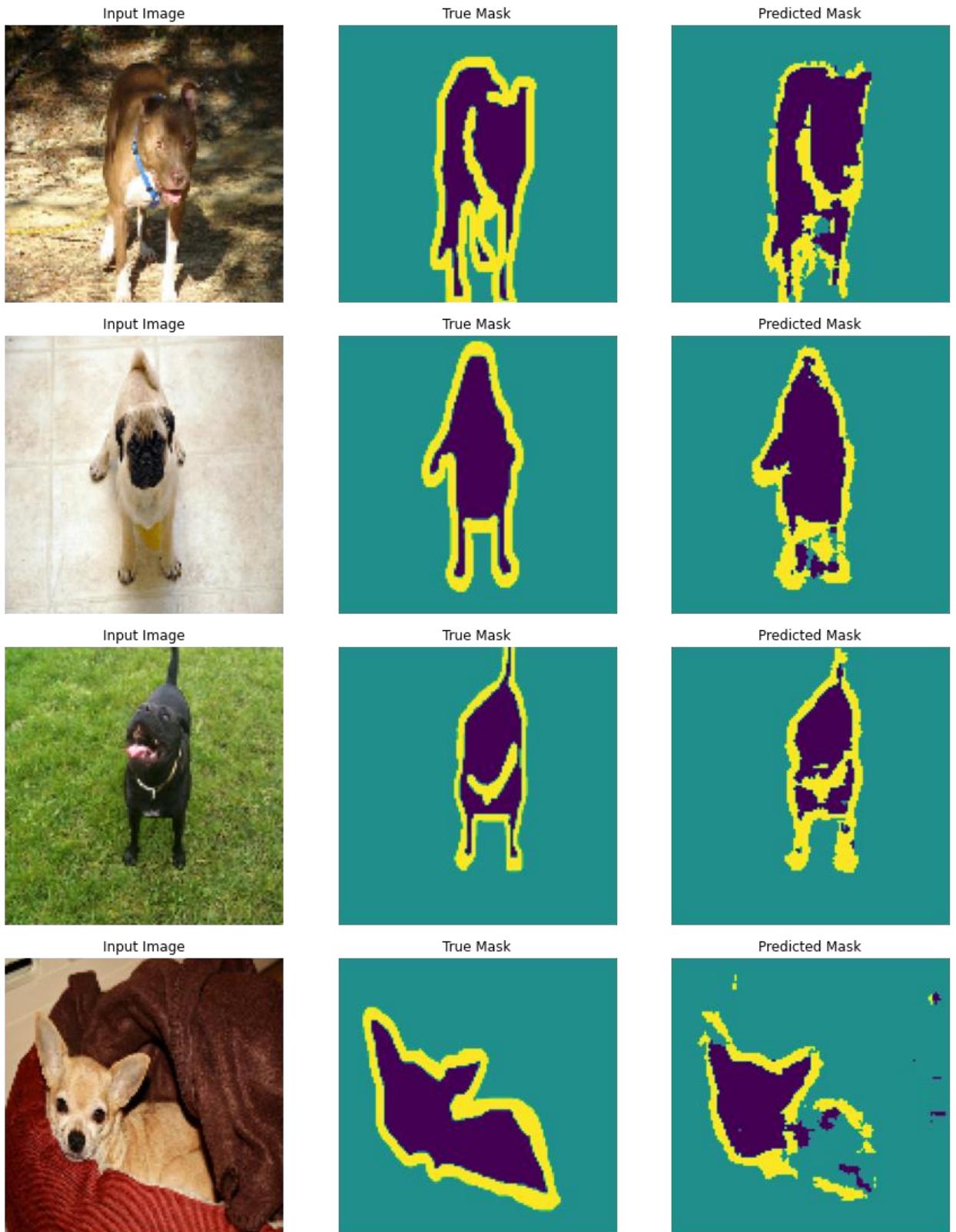




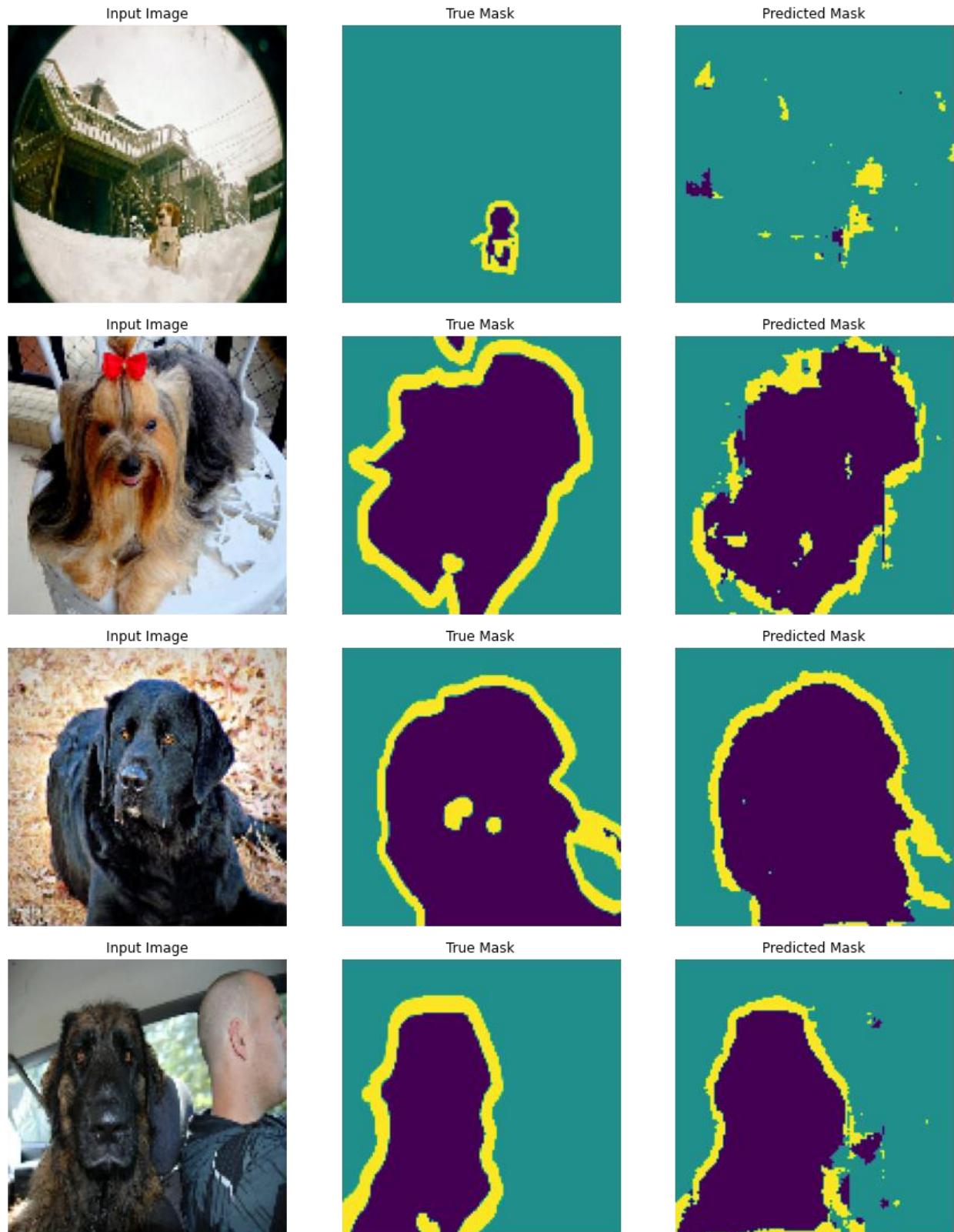


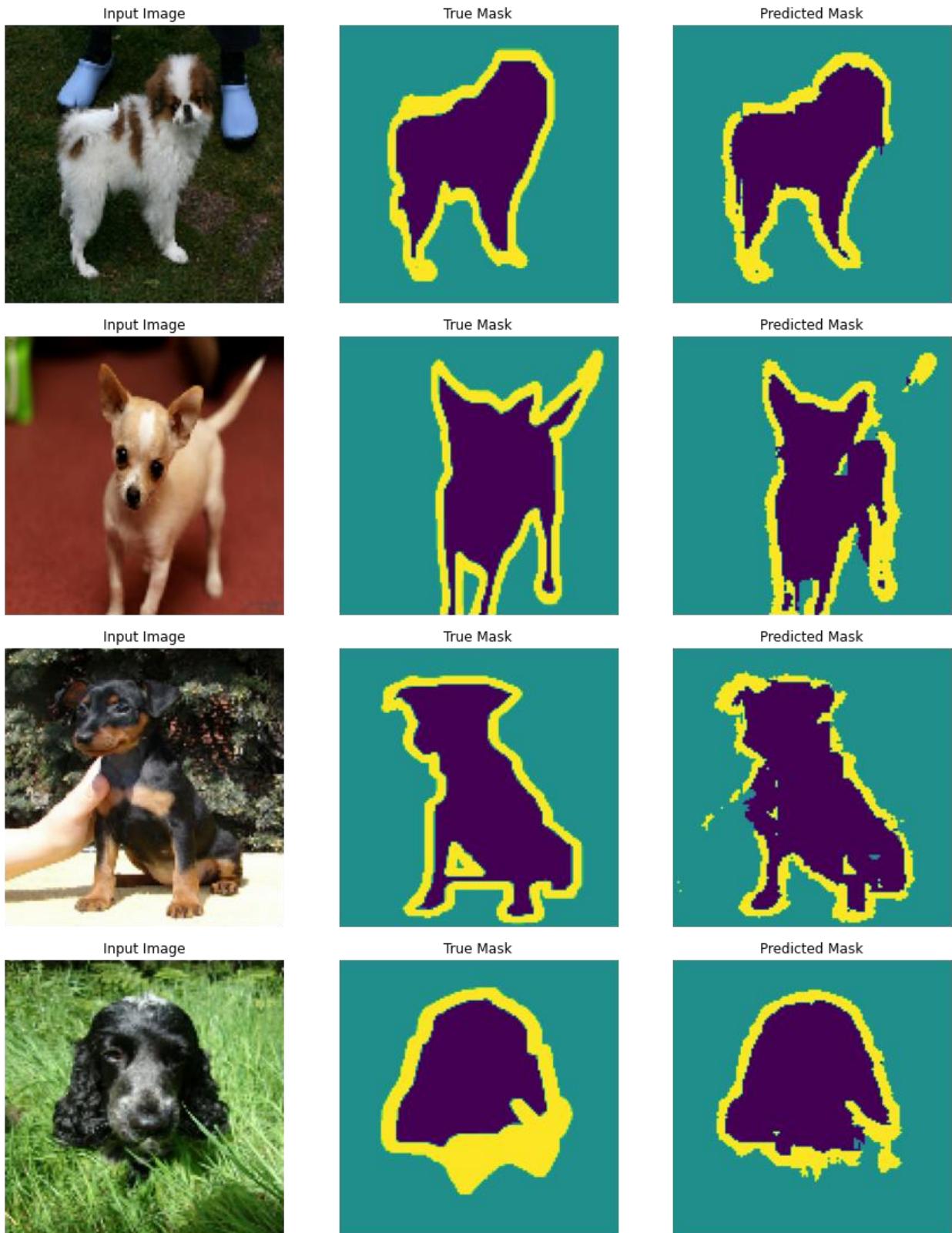
Pre-trained model (MobileNetV2)



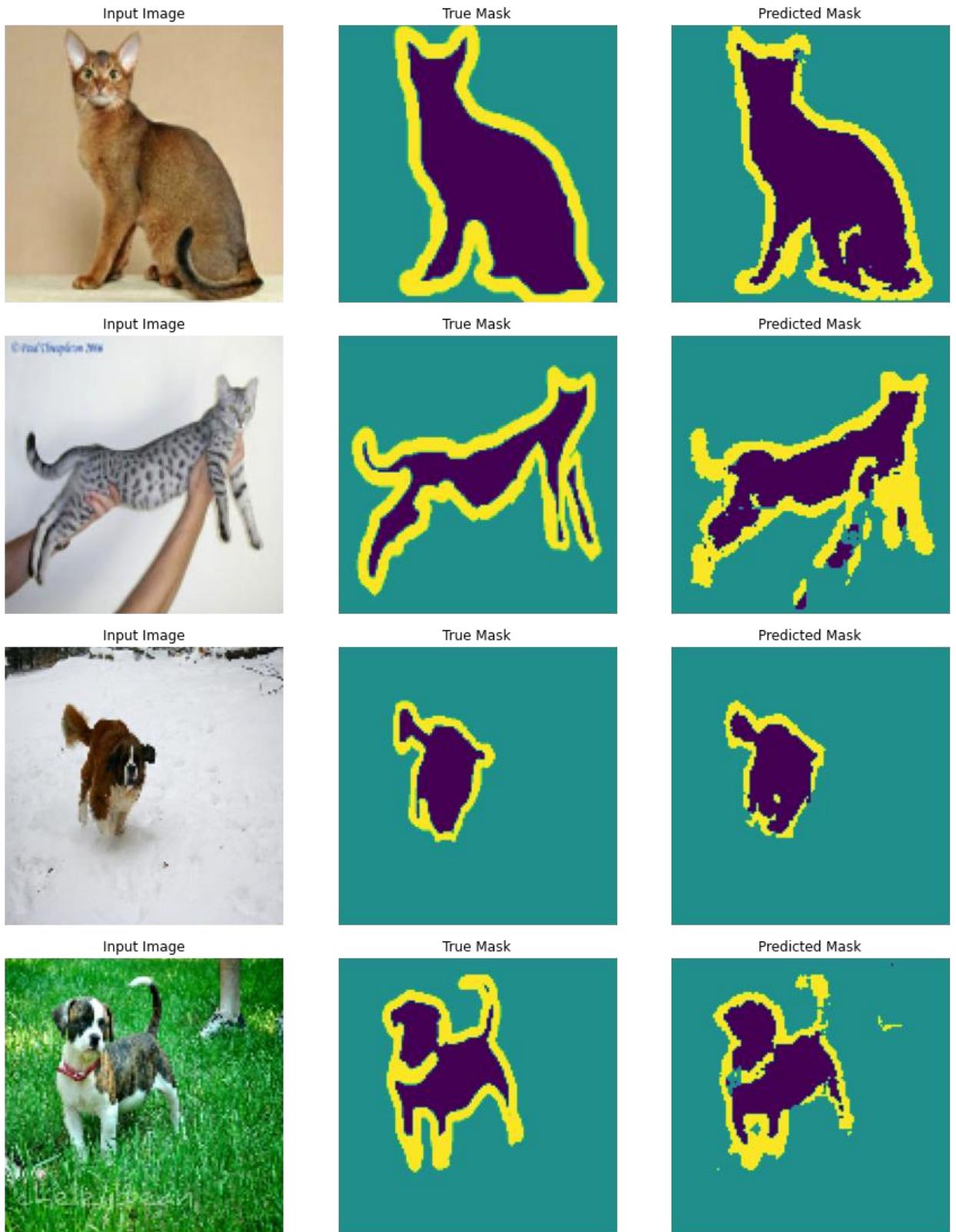




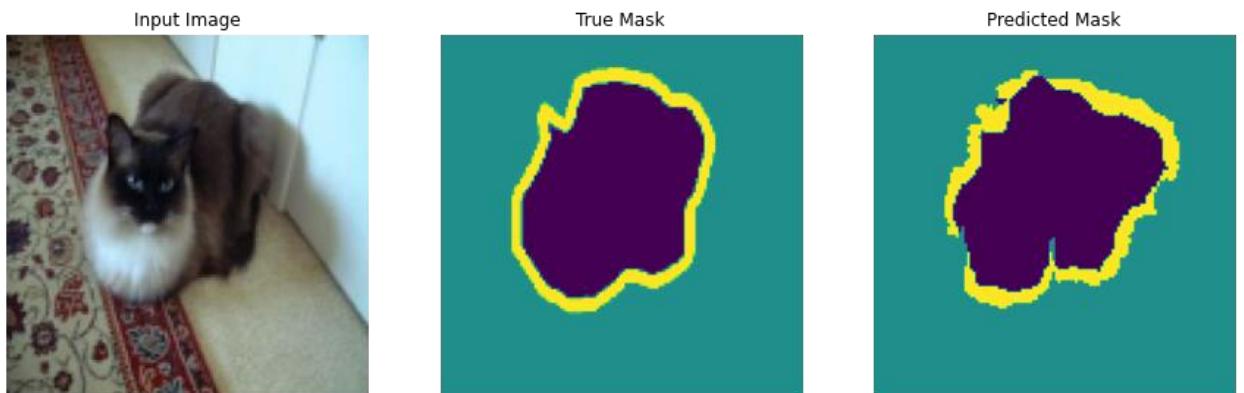


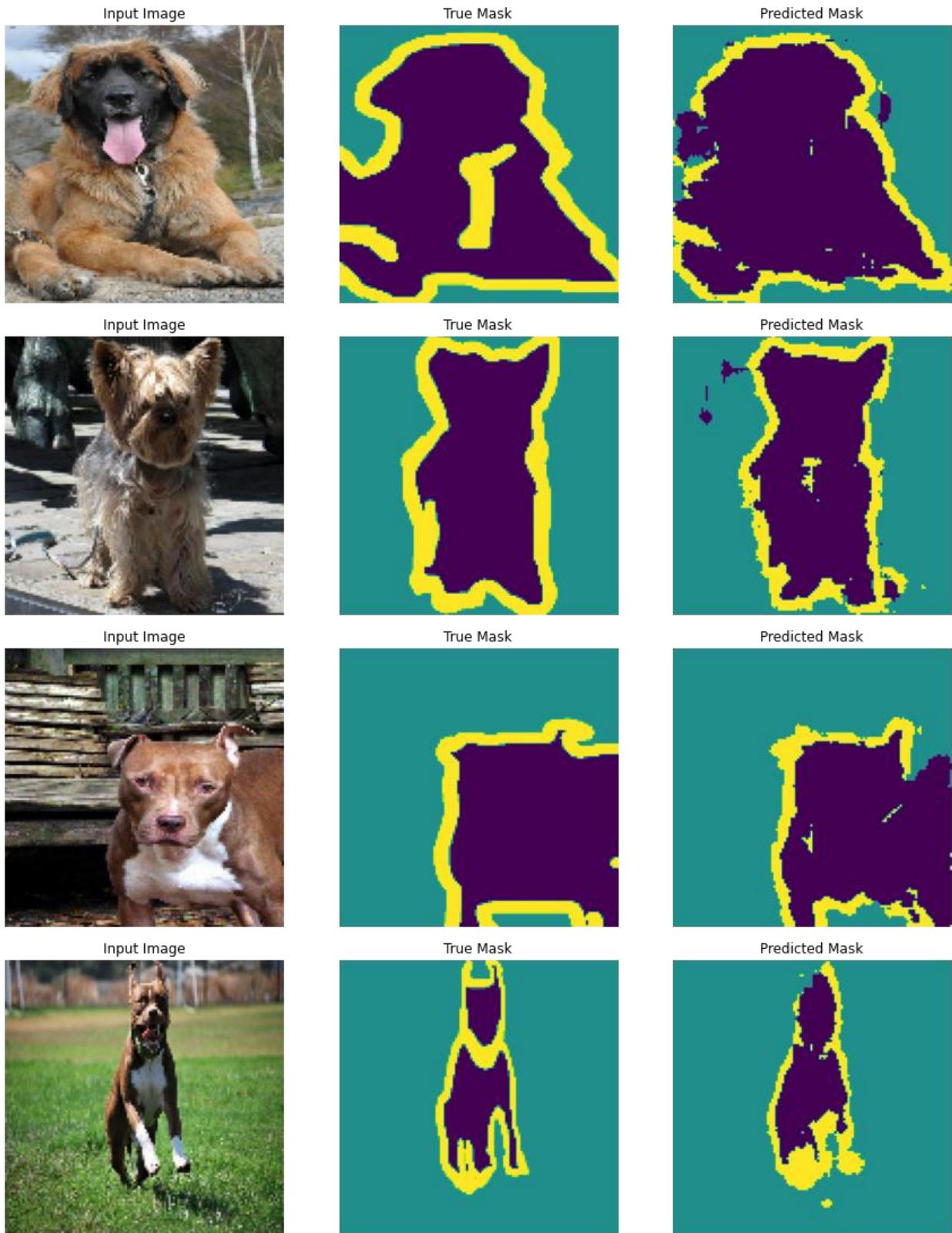




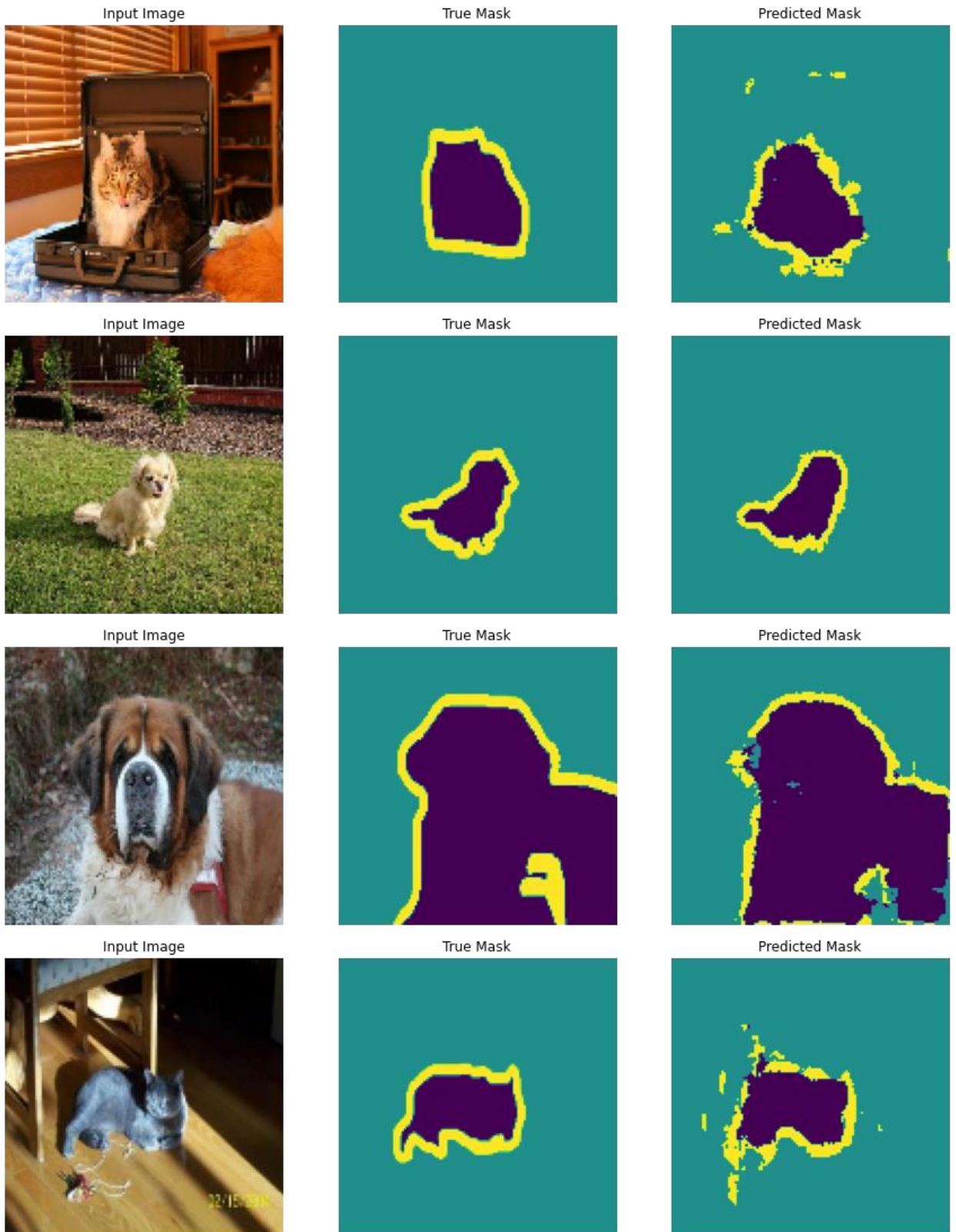


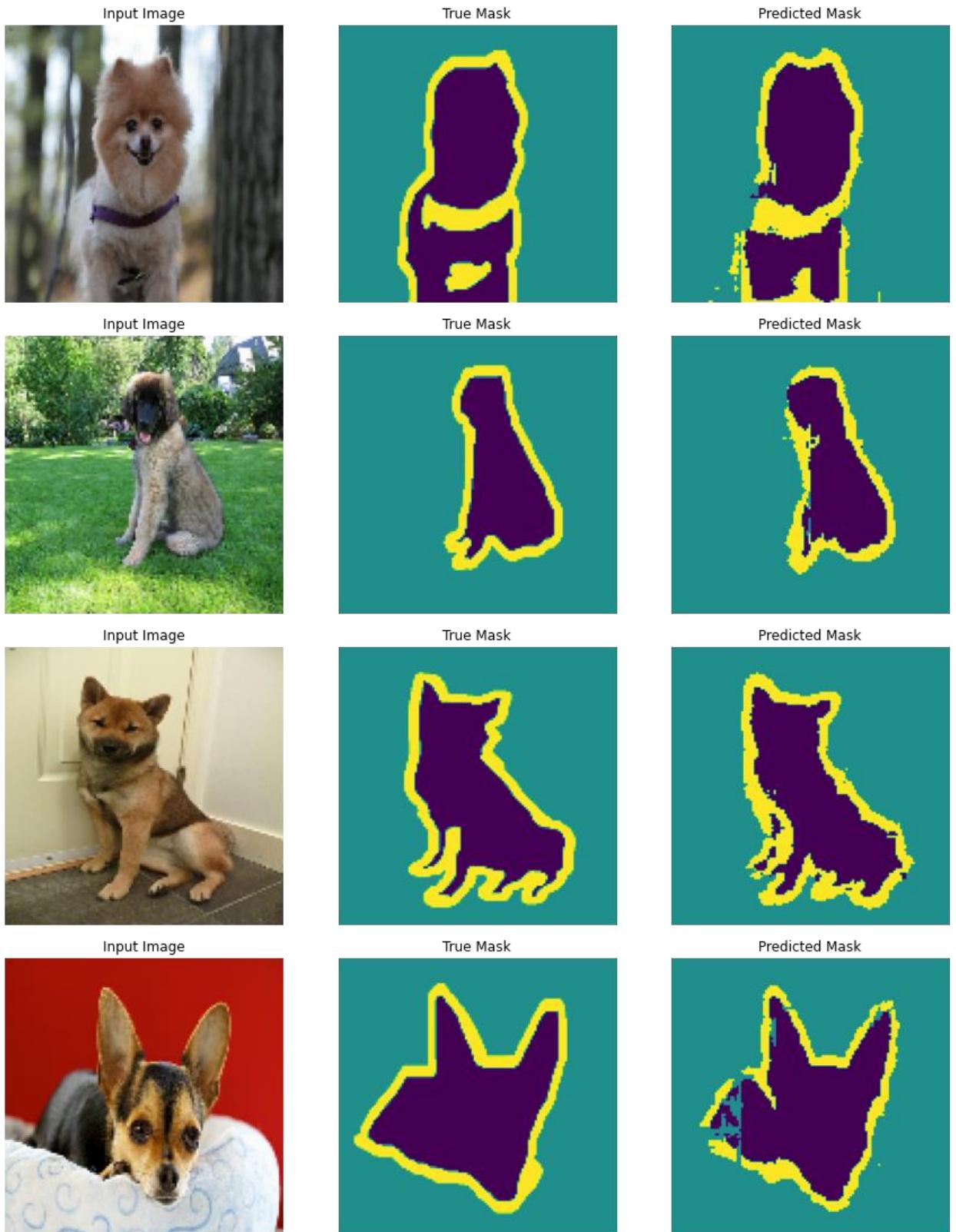




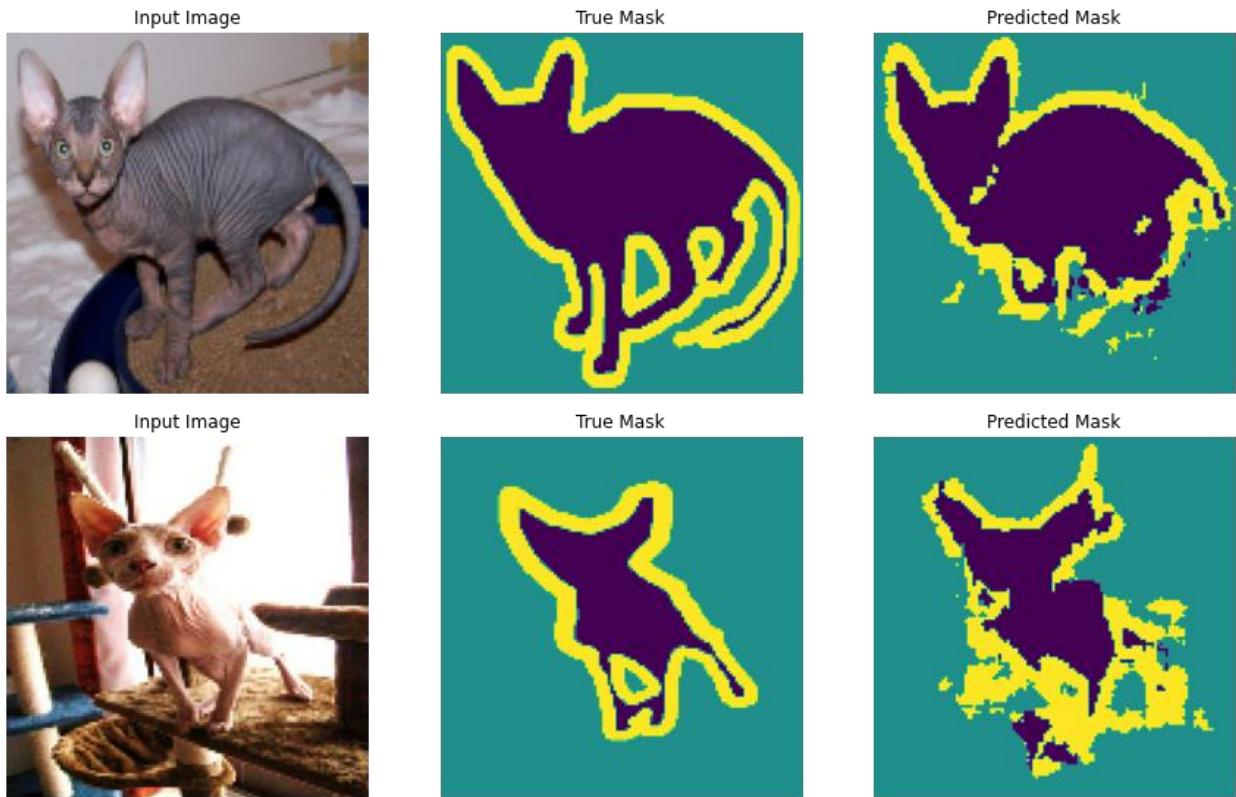






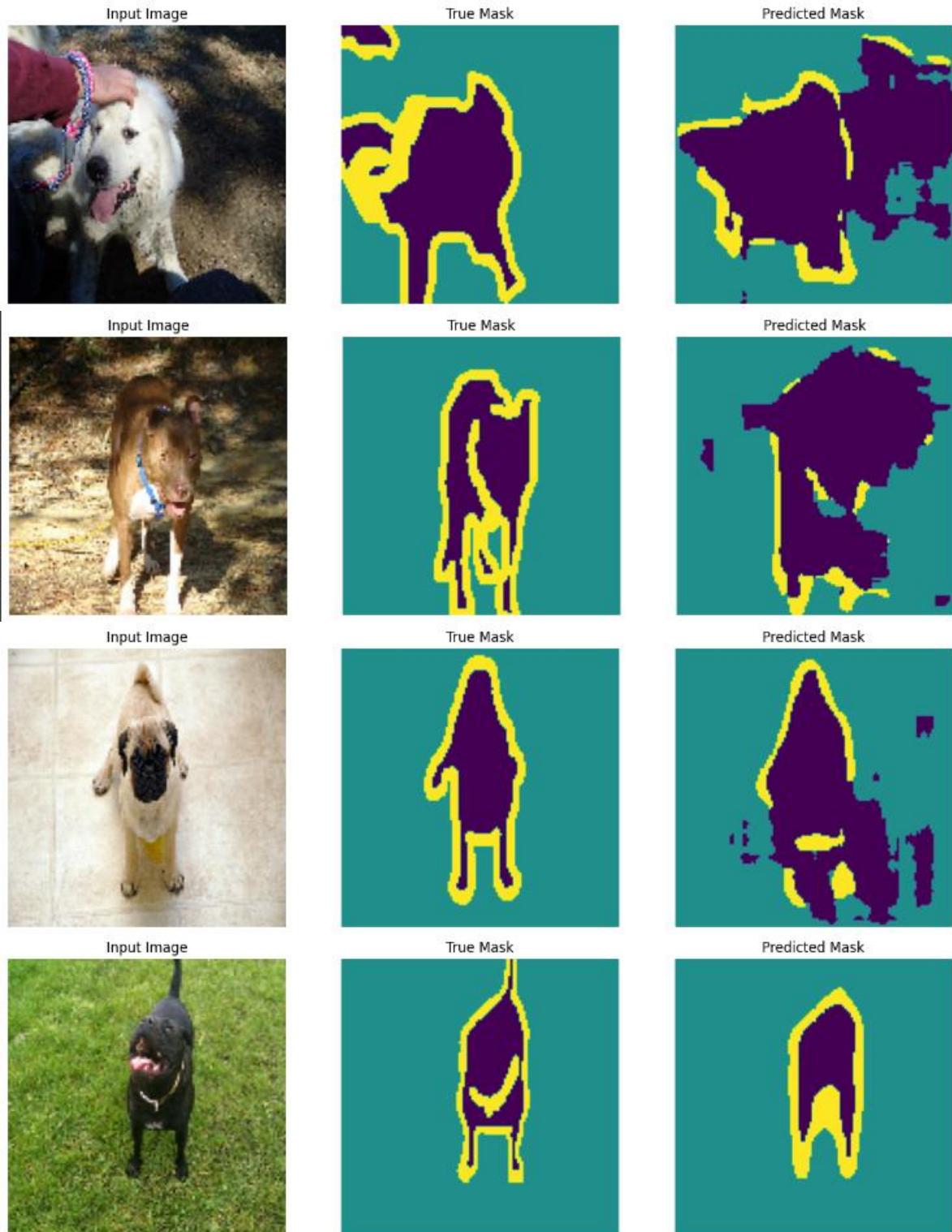


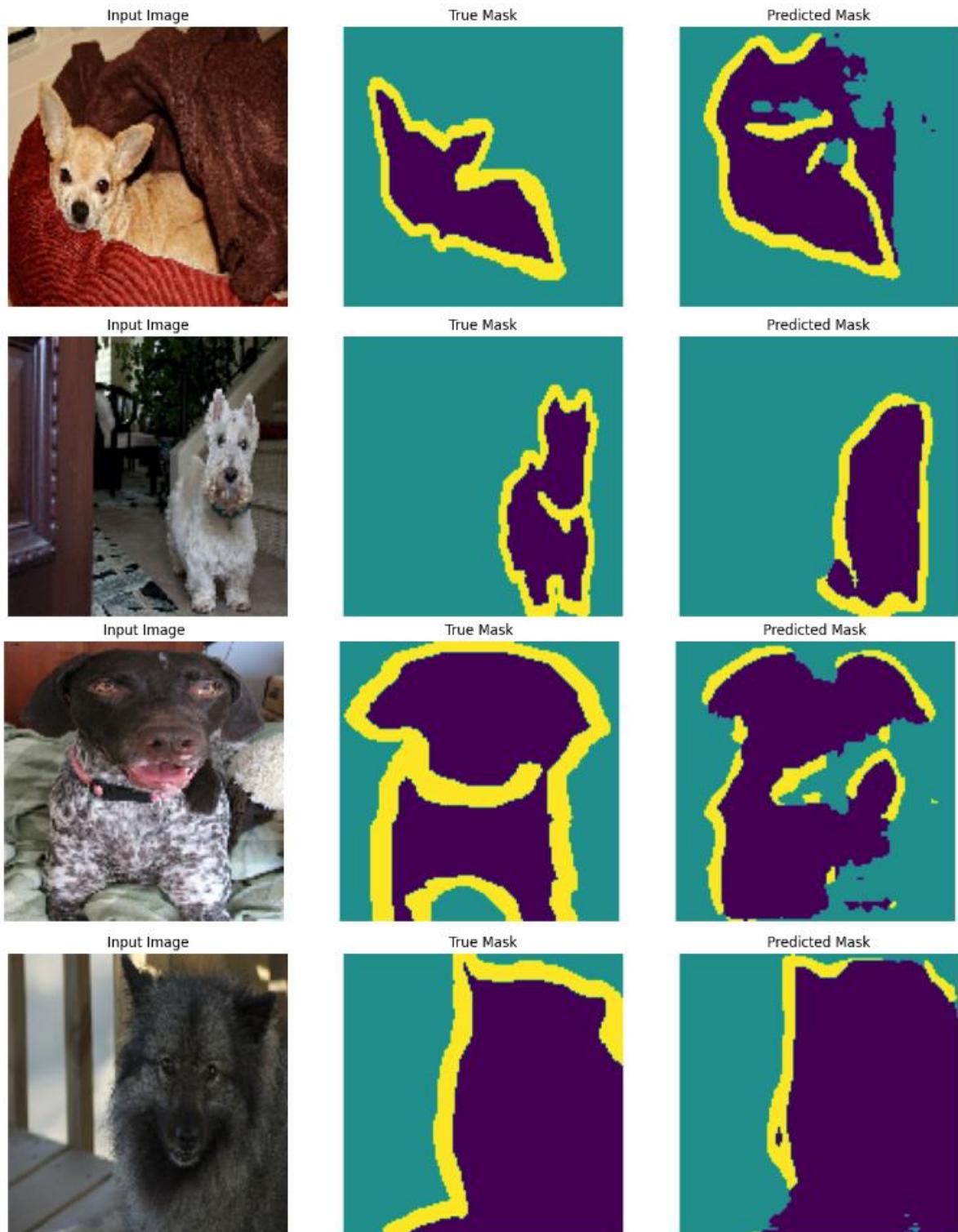


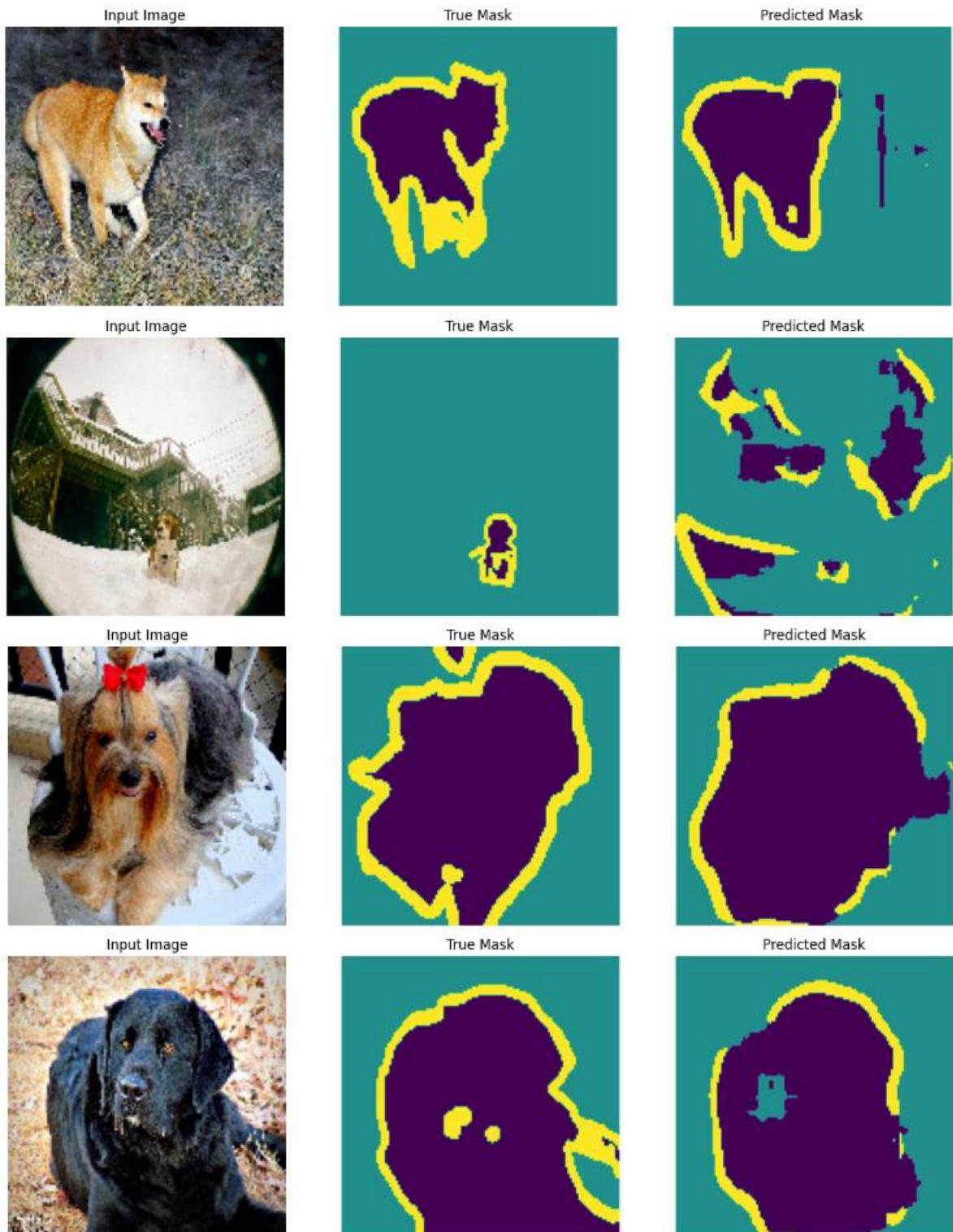


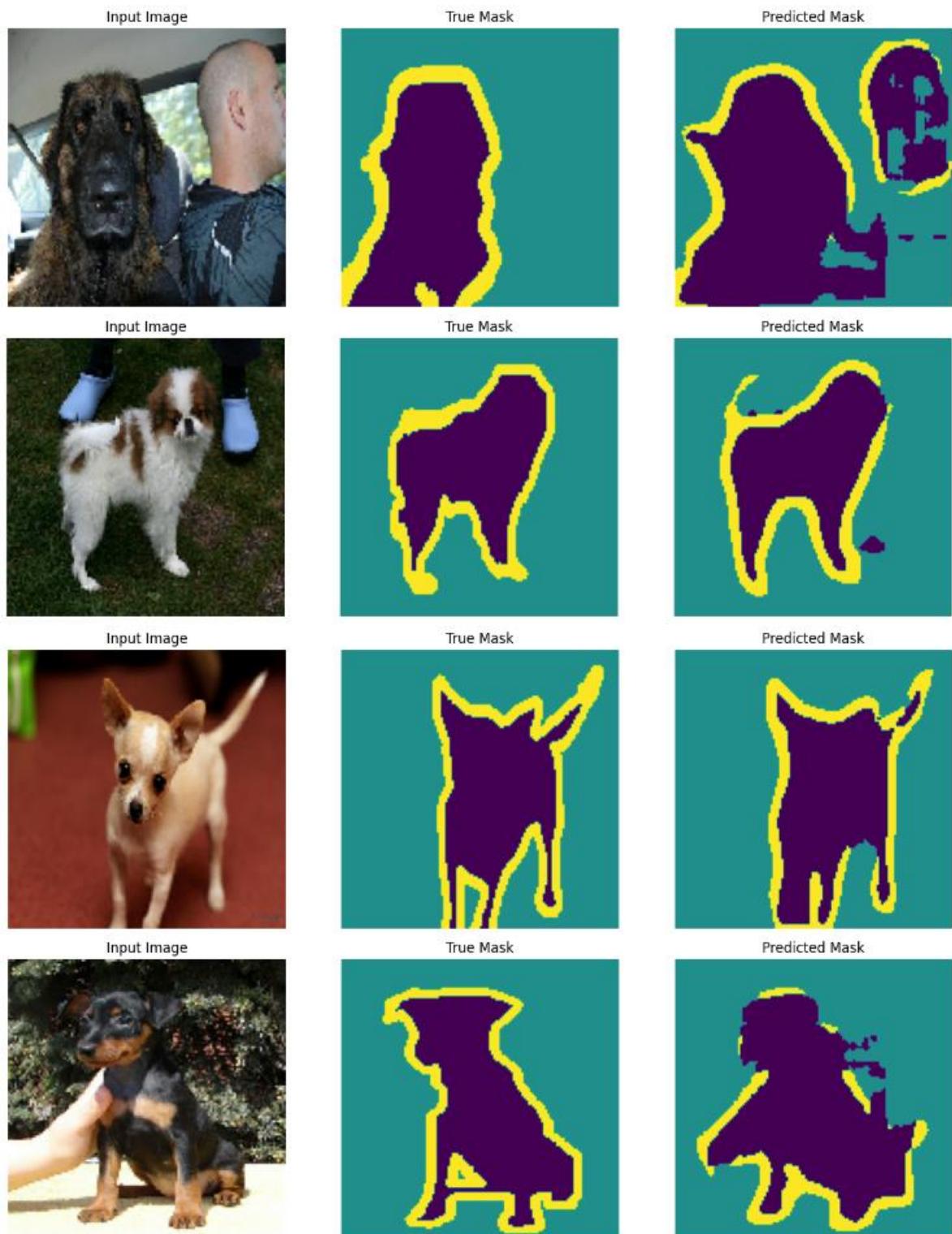
Segnet Model

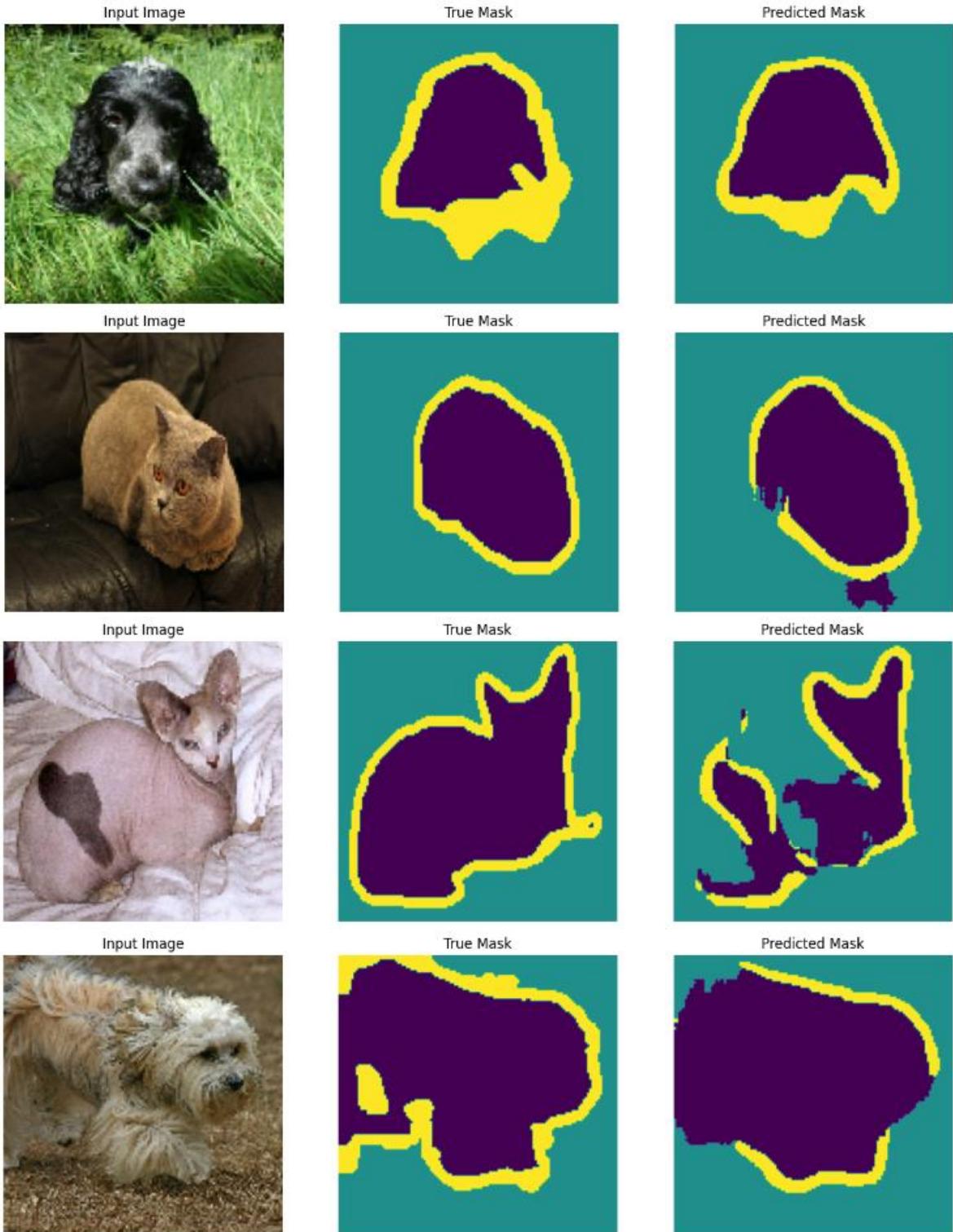


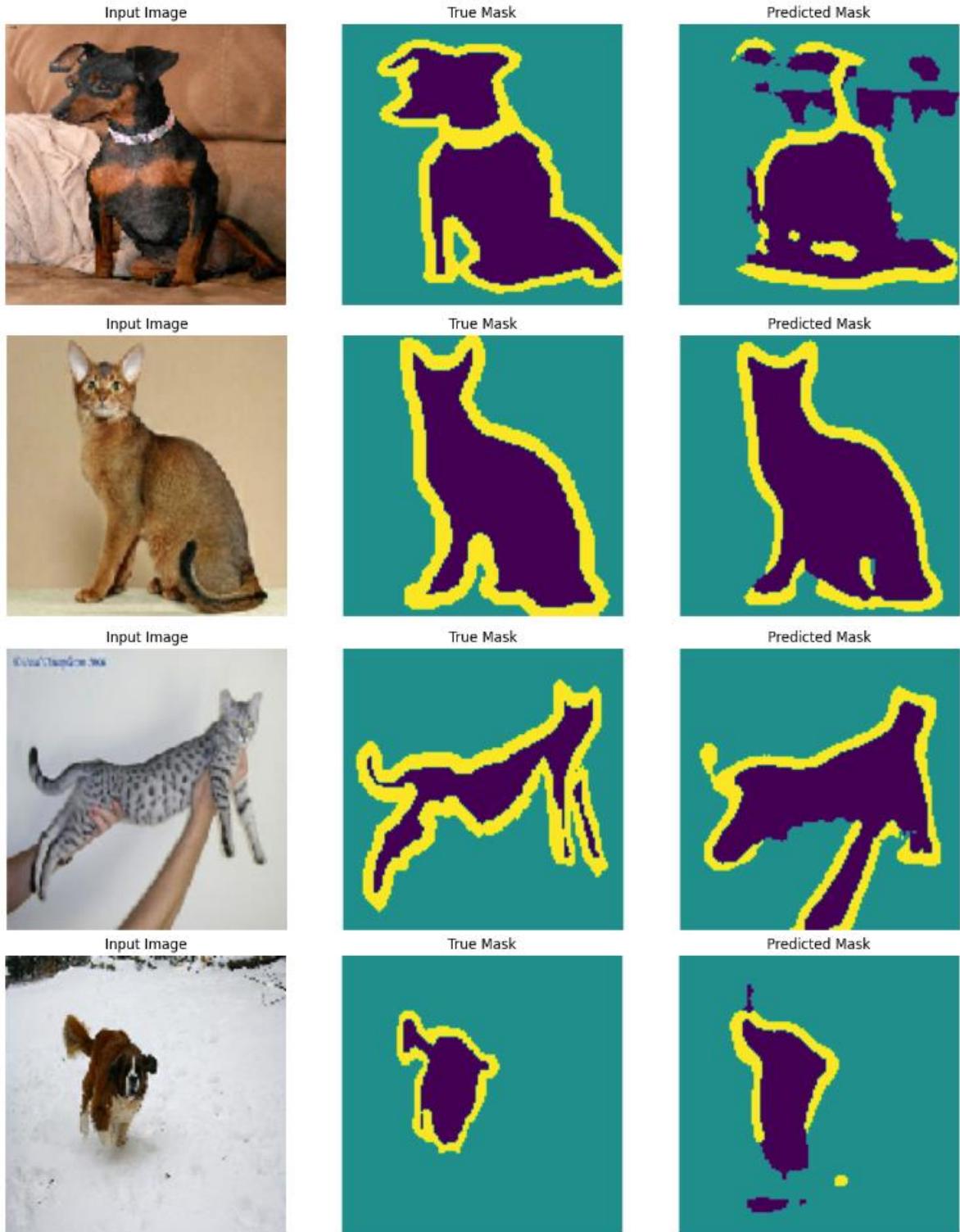


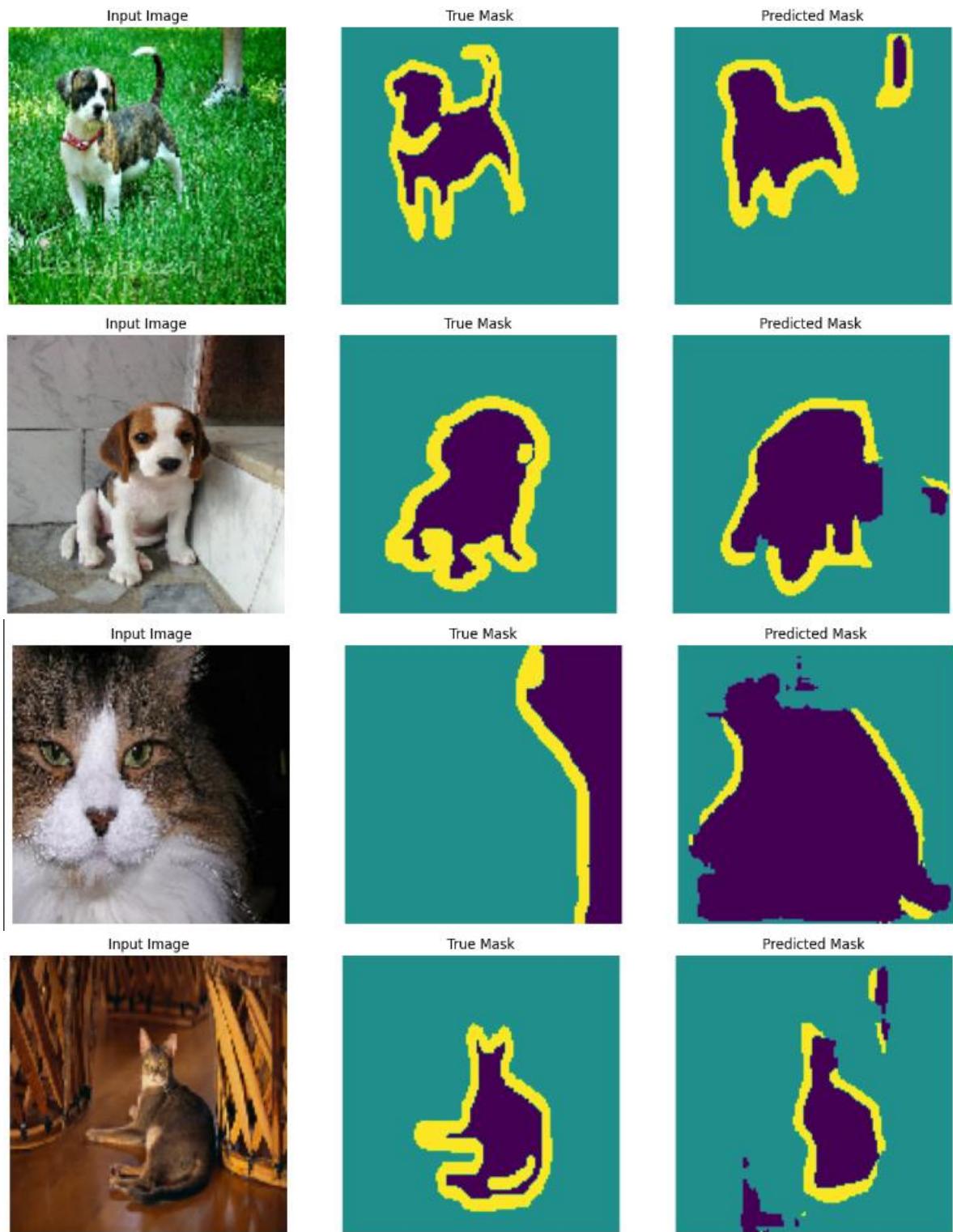


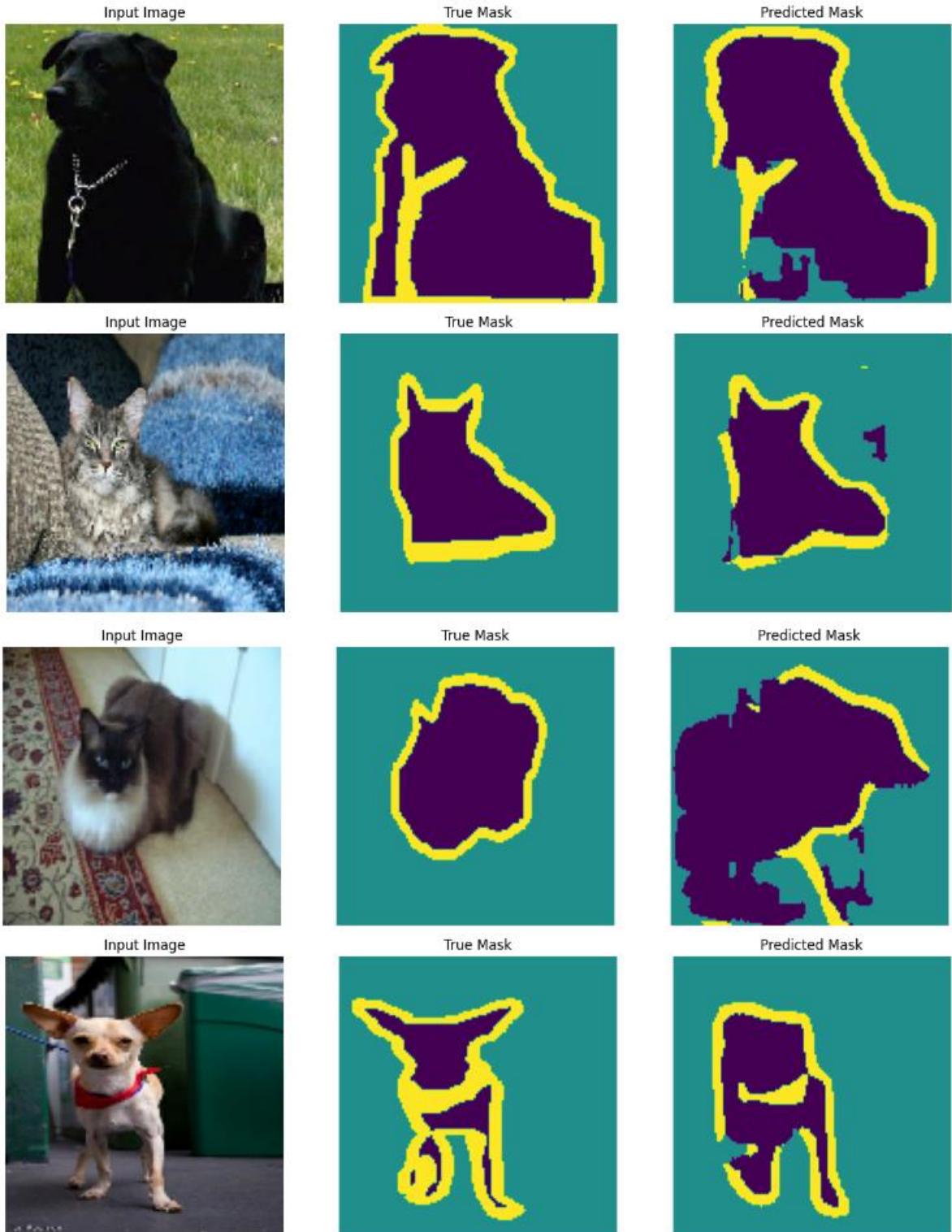


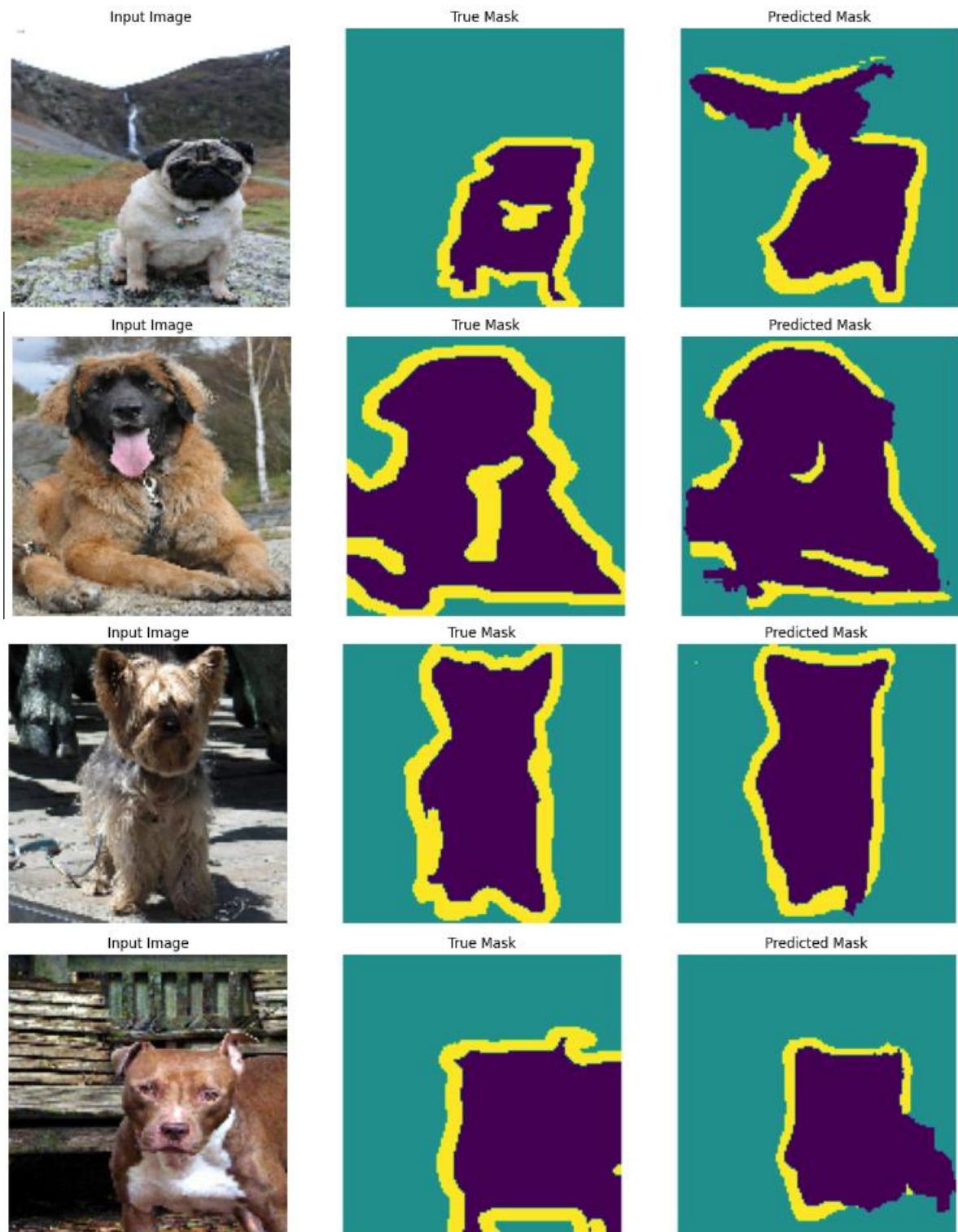


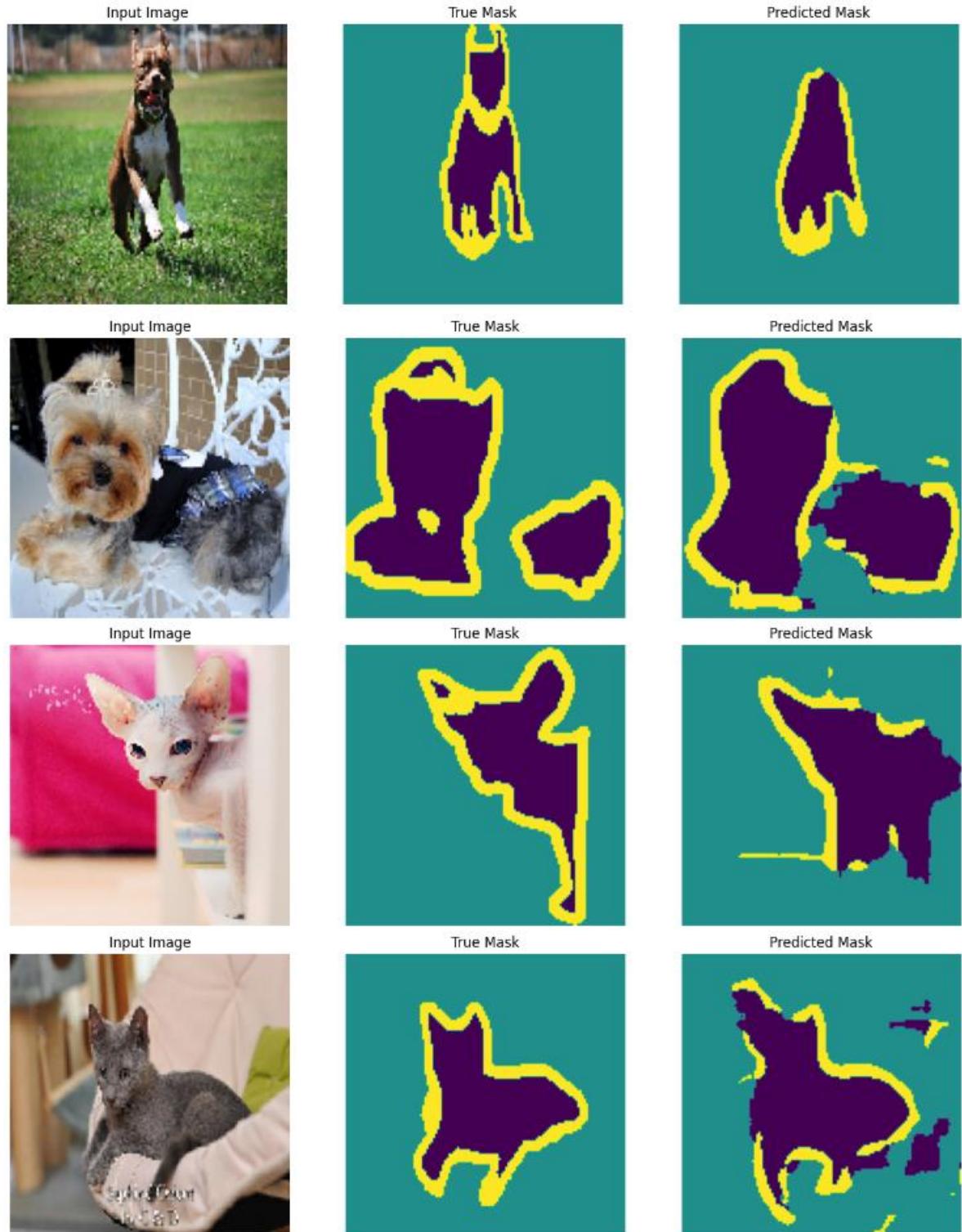


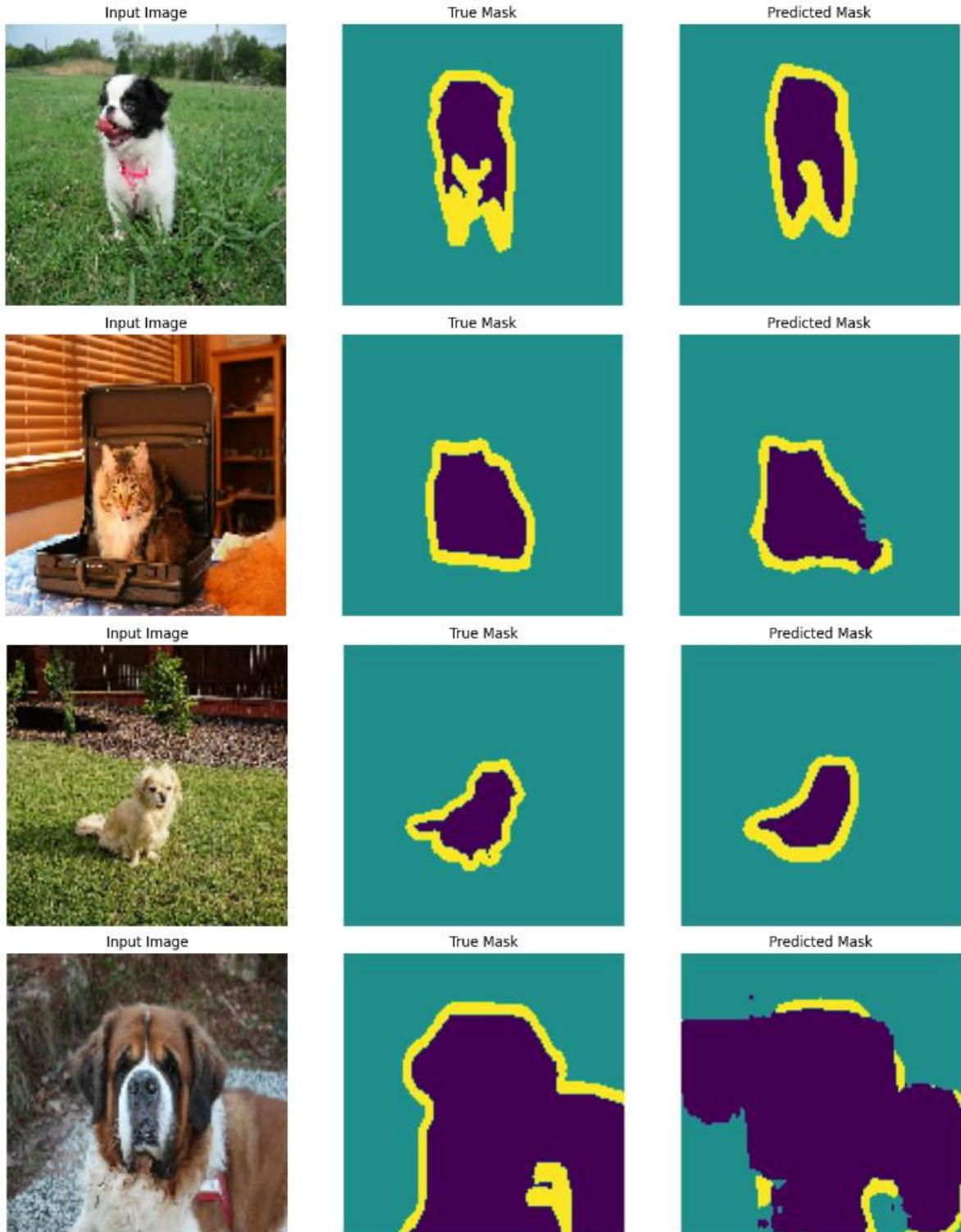


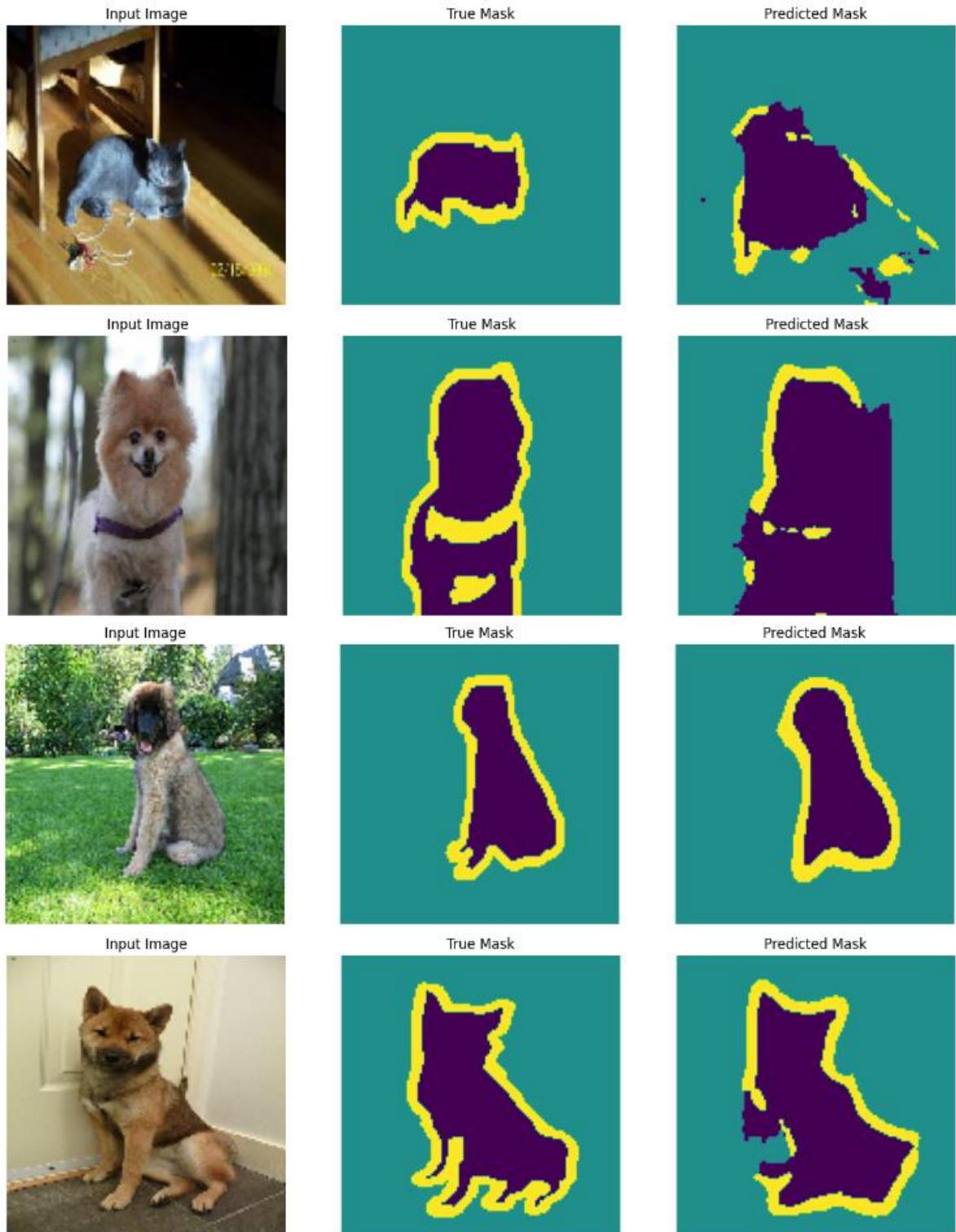


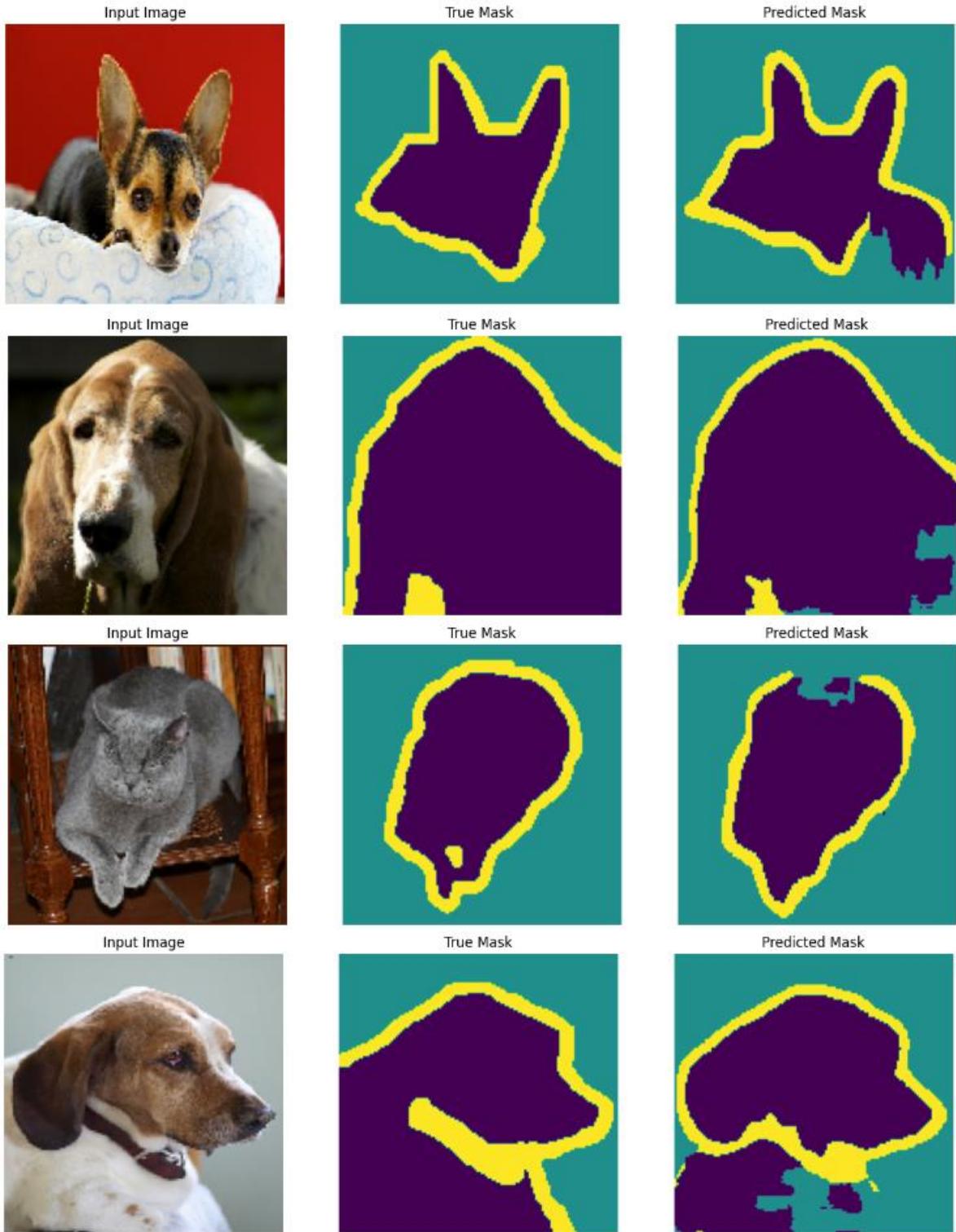


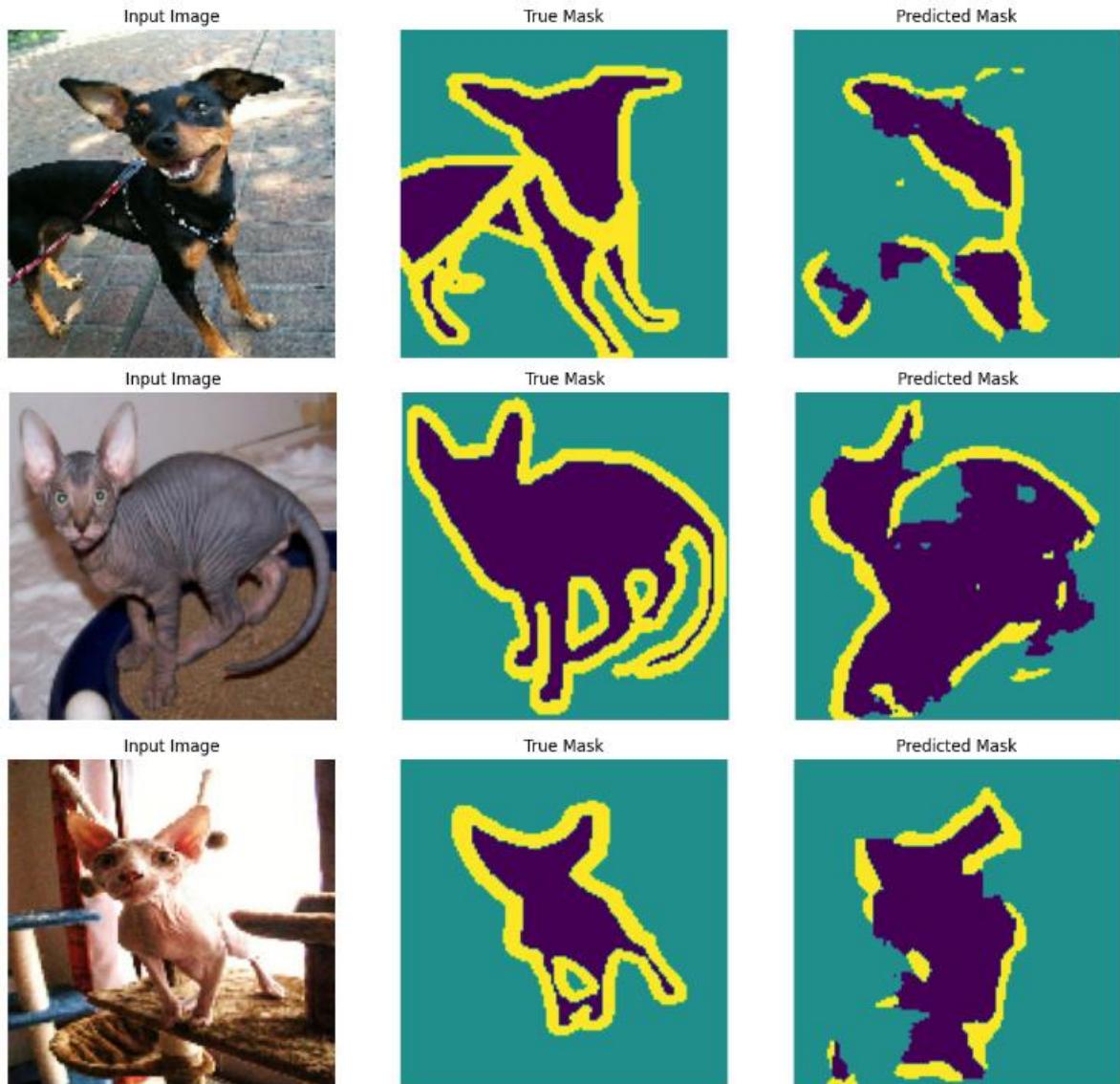












FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: T3, Y3	Study week no.: 2
Student Name & ID: Shreevishal A/L N.Rajandhiran, 19ACB04290	
Supervisor: Dr. Mogana Vadiveloo	
Project Title: Object Based Segmentation and Analysis using Deep Learning Algorithm for Cats and Dogs Images	

1. WORK DONE

Model research and further training

2. WORK TO BE DONE

To build the segnet model based on content researched

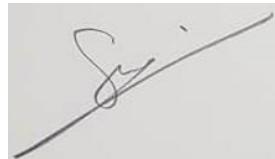
3. PROBLEMS ENCOUNTERED

Usage limit on Google Colab

Excessive documentation to research

4. SELF EVALUATION OF THE PROGRESS

Gained significant knowledge on deep learning models and its implementation



V. mogana vadiveloo

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: T3, Y3	Study week no.: 4
Student Name & ID: Shreevishal A/L N.Rajandhiran, 19ACB04290	
Supervisor: Dr. Mogana Vadiveloo	
Project Title: Object Based Segmentation and Analysis using Deep Learning Algorithm for Cats and Dogs Images	

1. WORK DONE

Testing the model and its validation

2. WORK TO BE DONE

To test the segnet model

3. PROBLEMS ENCOUNTERED

Usage limit on Google Colab

Excessive documentation to research

4. SELF EVALUATION OF THE PROGRESS

Gained significant knowledge on deep learning models and its implementation



V. mogana

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: T3, Y3	Study week no.: 6
Student Name & ID: Shreevishal A/L N.Rajandhiran, 19ACB04290	
Supervisor: Dr. Mogana Vadiveloo	
Project Title: Object Based Segmentation and Analysis using Deep Learning Algorithm for Cats and Dogs Images	

1. WORK DONE

Model refinements

2. WORK TO BE DONE

To test on all existing models

3. PROBLEMS ENCOUNTERED

Usage limit on Google Colab

Excessive documentation to research

4. SELF EVALUATION OF THE PROGRESS

Gained significant knowledge on deep learning models and its implementation



V. mogana

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: T3, Y3	Study week no.: 8
Student Name & ID: Shreevishal A/L N.Rajandhiran, 19ACB04290	
Supervisor: Dr. Mogana Vadiveloo	
Project Title: Object Based Segmentation and Analysis using Deep Learning Algorithm for Cats and Dogs Images	

1. WORK DONE

Deployments of the model

2. WORK TO BE DONE

To build the web application

3. PROBLEMS ENCOUNTERED

Difficulty in implementing the framework from Flask

4. SELF EVALUATION OF THE PROGRESS

Gained experience on the Flask framework



V. mogana

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: T3, Y3	Study week no.: 10
Student Name & ID: Shreevishal A/L N.Rajandhiran, 19ACB04290	
Supervisor: Dr. Mogana Vadiveloo	
Project Title: Object Based Segmentation and Analysis using Deep Learning Algorithm for Cats and Dogs Images	

1. WORK DONE

Progress on model deployment

2. WORK TO BE DONE

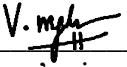
To build the web application

3. PROBLEMS ENCOUNTERED

Difficulty in implementing the framework from Flask

4. SELF EVALUATION OF THE PROGRESS

Gained significant knowledge on deep learning models and its implementation



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: T3, Y3	Study week no.: 12
Student Name & ID: Shreevishal A/L N.Rajandhiran, 19ACB04290	
Supervisor: Dr. Mogana Vadiveloo	
Project Title: Object Based Segmentation and Analysis using Deep Learning Algorithm for Cats and Dogs Images	

1. WORK DONE

Final model deployment and web application design finalization

2. WORK TO BE DONE

Report writing

3. PROBLEMS ENCOUNTERED

4. SELF EVALUATION OF THE PROGRESS

Gained knowledge when it comes to preparing accurate reports



V. mogana vadiveloo
Supervisor's signature

Student's signature

POSTER



FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Project Supervisor: Dr. Mogana Vadiveloo
Project Developer: Shreevishal A/L N.Rajandhiran

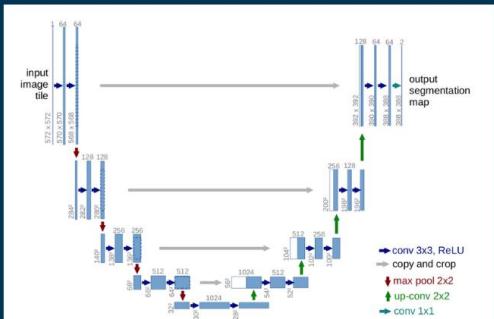
Introduction

This web application allows for users to segment cats and dogs as the region of interest in natural images

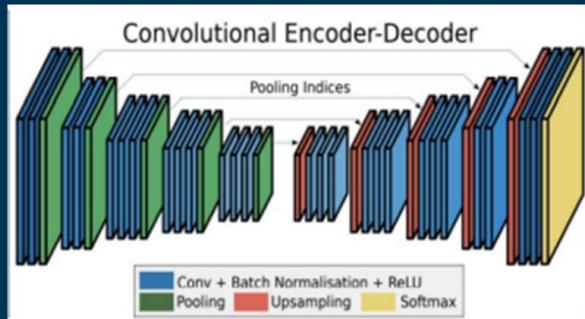
Objective

To develop a web application that automatically performs object-based image segmentation with minimal user input

Usage of Segnet and Unet Models



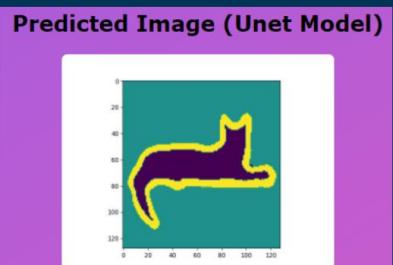
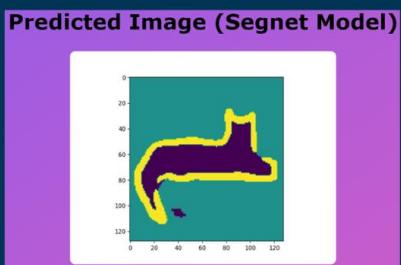
Unet architecture



Segnet architecture

Web Application Implementation

Automatic!



Unet Accuracy: 88.22 %
Segnet Accuracy: 85.93 %

PLAGIARISM CHECK RESULT

Object Based Segmentation and Analysis using Deep Learning Algorithm for Cats and Dogs Images

ORIGINALITY REPORT

5%	3%	4%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	dokumen.pub Internet Source	1%
2	McGuinness, K.. "A comparative evaluation of interactive segmentation algorithms", Pattern Recognition, 201002 Publication	<1%
3	Anjani Raj Yadlapalli, Ninad Mohite, Vijayant Pawar, Shelly Sachdeva. "Artificially Intelligent Decentralized Autonomous Organization", 2019 4th International Conference on Information Systems and Computer Networks (ISCON), 2019 Publication	<1%
4	repositorio.unicamp.br Internet Source	<1%
5	www.kaggle.com Internet Source	<1%
6	uhra.herts.ac.uk Internet Source	<1%

7	www.mdpi.com Internet Source	<1 %
8	"Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries", Springer Science and Business Media LLC, 2022 Publication	<1 %
9	"Computer Vision – ECCV 2018", Springer Nature America, Inc, 2018 Publication	<1 %
10	Rommel M. Lim, Francisco Emmanuel T. Munsayac, Nilo T. Bugtai, Renann G. Baldovino. "A Predictive Tool for Heart Disease Diagnosis using Artificial Neural Network", 2021 IEEE 13th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM), 2021 Publication	<1 %
11	fict.utar.edu.my Internet Source	<1 %
12	etda.libraries.psu.edu Internet Source	<1 %
13	www.altera.co.jp Internet Source	<1 %
14	www.amazon.com Internet Source	<1 %

		<1 %
15	Kim, Dongik, Yujin Jung, Kar-Ann Toh, Byungjun Son, and Jaihie Kim. "An empirical study on iris recognition in a mobile phone", <i>Expert Systems with Applications</i> , 2016. Publication	<1 %
16	libguides.ntu.edu.sg Internet Source	<1 %
17	Changjian Zhou, Sihan Zhou, Jinge Xing, Jia Song. "Tomato Leaf Disease Identification by Restructured Deep Residual Dense Network", <i>IEEE Access</i> , 2021 Publication	<1 %
18	bugs.launchpad.net Internet Source	<1 %
19	open.uct.ac.za Internet Source	<1 %
20	scyr.kpi.fei.tuke.sk Internet Source	<1 %
21	Andreas W. Liehr, Heike S. Rolfs, Klaus J. Buchenrieder, Ulrich Nageldinger. "Generating MARTE allocation models from activity threads", <i>2008 Forum on Specification, Verification and Design Languages</i> , 2008 Publication	<1 %

22	Cuong Ly, Adam M. Olsen, Ian J. Schwerdt, Reid Porter, Kari Sentz, Luther W. McDonald, Tolga Tasdizen. "A new approach for quantifying morphological features of U3O8 for nuclear forensics using a deep learning model", Journal of Nuclear Materials, 2019 Publication	<1 %
23	Nillmani, Neeraj Sharma, Luca Saba, Narendra N. Khanna, Mannudeep K. Kalra, Mostafa M. Fouda, Jasjit S. Suri. "Segmentation-Based Classification Deep Learning Model Embedded with Explainable AI for COVID-19 Detection in Chest X-ray Scans", Diagnostics, 2022 Publication	<1 %
24	elib.dlr.de Internet Source	<1 %
25	ssebook.pusan.ac.kr:8010 Internet Source	<1 %
26	www.coursehero.com Internet Source	<1 %
27	Thessa T. J. P. Kockelkorn, Cornelia M. Schaefer-Prokop, Gracijela Bozovic, Arrate Muñoz-Barrutia et al. "Interactive lung segmentation in abnormal human and animal chest CT scans", Medical Physics, 2014 Publication	<1 %

- 28 Ting Zhang, Muhammad Waqas, Yu Fang, Zhaoying Liu, Zahid Halim, Yujian Li, Sheng Chen. "Weakly-Supervised Butterfly Detection Based on Saliency Map", Pattern Recognition, 2023 <1 %
Publication
-
- 29 Sherry Verma*, Latika Duhan, Monica Chaudhry. "Evaluation of Convolutional Neural Network Model for Classifying Red and Healthy Eye", International Journal of Innovative Technology and Exploring Engineering, 2019 <1 %
Publication
-
- 30 link.springer.com <1 %
Internet Source

Exclude quotes Off
Exclude bibliography Off

Exclude matches Off

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1 of 1



**FACULTY OF INFORMATION AND COMMUNICATION
TECHNOLOGY**

Full Name(s) of Candidate(s)	Shreevishal A/L N.Rajandhiran
ID Number(s)	19ACB04290
Programme / Course	BACHELOR OF COMPUTER SCIENCE (HONOURS)
Title of Final Year Project	Object Based Segmentation and Analysis using Deep Learning Algorithm for Cats and Dogs Images

Similarity	Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)
Overall similarity index: <u>5</u> %	
Similarity by source Internet Sources: <u>3</u> % Publications: <u>4</u> % Student Papers: <u>0</u> %	No comments.
Number of individual sources listed of more than 3% similarity: <u>-</u>	
Parameters of originality required and limits approved by UTAR are as Follows: (i) Overall similarity index is 20% and below, and (ii) Matching of individual sources listed must be less than 3% each, and (iii) Matching texts in continuous block must not exceed 8 words <i>Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.</i>	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

V. my

Signature of Supervisor

Name: _____

Date: _____

Signature of Co-Supervisor

Name: _____

Date: _____



UNIVERSITI TUNKU ABDUL RAHMAN
FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY
(KAMPAR CAMPUS)
CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	19ACB04290
Student Name	Shreevishal A/L N.Rajandhiran
Supervisor Name	Dr. Mogana Vadiveloo

TICK (✓)	DOCUMENT ITEMS
Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.	
X	Front Plastic Cover (for hardcopy)
✓	Title Page
✓	Signed Report Status Declaration Form
✓	Signed FYP Thesis Submission Form
✓	Signed form of the Declaration of Originality
✓	Acknowledgement
✓	Abstract
✓	Table of Contents
✓	List of Figures (if applicable)
✓	List of Tables (if applicable)
X	List of Symbols (if applicable)
✓	List of Abbreviations (if applicable)
✓	Chapters / Content
✓	Bibliography (or References)
✓	All references in bibliography are cited in the thesis, especially in the chapter of literature review
✓	Appendices (if applicable)
✓	Weekly Log
✓	Poster
✓	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
✓	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.



(Signature of Student)

Date: 28th April 2023