

School of Computing

FACULTY OF ENGINEERING AND
PHYSICAL SCIENCES



UNIVERSITY OF LEEDS

Final Report

Surgical Scene Understanding in Laparoscopic Hysterectomy via Deep Learning: Anatomy and Surgical Instrument Segmentation

Rashad Hosseini

**Submitted in accordance with the requirements for the degree of
BSc Computer Science (Industrial)**

2023/2024

COMP3931 Individual Project

The candidate confirms that the following have been submitted:

Items	Format	Recipient(s) and Date
<i>Final Report</i>	<i>PDF file</i>	<i>Uploaded to Minerva (30/04/2024)</i>
<i>Link to online code repository</i>	<i>URL</i>	<i>Sent to supervisor and assessor (30/04/2024)</i>
<i>Demo Video</i>	<i>URL</i>	<i>Sent to supervisor and assessor (30/04/2024)</i>

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Rashad Hosseini)



Summary

This project focuses on enhancing surgical scene understanding for laparoscopic hysterectomy procedures. Hysterectomy is a common surgery which involves the removal of the uterus. This surgery has its difficulties as the surgeon has to identify anatomical structures and navigate surgical tools within the body by only using live camera feeds.

The goal of this project is to leverage state-of-the-art computer vision methods to improve scene understanding. Central to this goal is the development of a segmentation model which is capable of segmenting, classifying and highlighting different surgical instruments and anatomical structures.

Through an iterative approach and multiple experimentations, this project aims to implement a simple segmentation model and refine the performance of this model through incremental changes. The final objective is achieving an effective segmentation model for laparoscopic hysterectomy procedures capable of delineating surgical tools and anatomies, hence contributing to the advancements of laparoscopic surgical techniques.

Acknowledgements

I want to thank my supervisor Dr. Duygu Sarikaya for her guidance, and support throughout this project.

I would also like to thank my assessor Arash Bozorgchenani for the valuable feedback and insights during the assessor meeting.

Finally, I would like to thank my family for the support they have provided me.

Table of Contents

Summary	iii
Acknowledgements	iv
Table of Contents	v
Chapter 1 – Introduction and Background Research	1
1.1 Introduction	1
1.2 Aims and Objectives	1
1.3 Background	2
1.3.1 Image Segmentation.....	2
1.3.2 Convolutional Neural Networks (CNNs)	4
1.3.3 Convolutional Encoder-Decoder Architecture.....	5
1.3.4 U-Net Architecture	6
1.3.5 How Neural Networks Learn	6
1.3.6 Challenges	8
1.3.7 Existing Solutions.....	9
1.3.8 Technologies Used.....	10
Chapter 2 – Methodology	11
2.1 Dataset	11
2.1.1 Dataset Acquisition.....	11
2.1.1 Dataset Evaluation	12
2.2 Data Preparation.....	12
2.2.1 Converting RGB Values to Float	12
2.2.2 Resizing Images and Masks	12
2.2.3 Standardizing Mask Data.....	13
2.2.4 Converting Mask Data to One-Hot Encoded Format	13
2.3 Train, Test, Validation	13
2.4 Data Visualization	13
2.5 Data Augmentation.....	14
2.6 Technology and Code Setup Organization.....	14
2.7 Building the Model.....	15
2.7.1 Model’s Architecture	15

2.7.2 Different Loss Functions for The Model.....	16
2.7.3 Adaptable Learning Rate	17
2.7.3 Early Stopping	17
2.8 Evaluation Metrics	17
2.8.1 Pixel-Wise Accuracy.....	17
2.8.2 Intersection Over Union (IoU)	18
2.8.3 Dice Coefficient	18
2.9 Prototype Implementation	19
Chapter 3 – Results and Experiments	20
3.1 Experiments Setup.....	20
3.2 Experiments.....	20
3.2.1 Adding Data Augmentation	21
3.2.2 Using Focal Loss	22
3.2.3 Using Weighted Categorical Cross Entropy	23
3.3 Results Visualization	24
3.4 Final Prototype.....	25
3.4.1 Prototype Evaluation.....	25
3.4.2 Prototype Limitations.....	26
3.5 Segmentation on Real-Time Video.....	27
Chapter 4 – Discussion	28
4.1 Conclusion	28
4.2 Challenges	28
4.2.1 Custom Function Implementations	28
4.2.2 Configuring TensorFlow with GPU	28
4.2.3 Computational Time and Resources.....	29
4.3 Limitations	29
4.4 Ideas for Future Work	29
4.4.1 Attention U-Net.....	29
4.4.2 Hyper Parameter Optimization	30
4.4.3 Improving the Dataset	30
4.4.4 Video Segmentation Using RNNs	30
4.4.5 Integration with Phase Detection Models	30
References	31
Appendix A – Self Appraisal	36
A.1 Critical Self-Evaluation	36

A.2 Personal Reflection and Lessons Learned	37
A.3 Legal, Social, Ethical and Professional Issues	37
A.3.1 Legal Issues	37
A.3.2 Social Issues.....	37
A.3.3 Ethical Issues	38
A.3.4 Professional Issues	38
Appendix B – External Materials	39
Appendix C – Additional Material.....	40
Appendix D – Links to Deliverables	44

Chapter 1 – Introduction and Background Research

1.1 Introduction

The field of minimally invasive surgery has great potential to be revolutionized by advances in computer vision and deep learning, which have made surgical scene understanding possible. These technologies offer surgeons improved visual and cognitive support, which can boost human capabilities and lead to more favourable outcomes for patients [1]. Laparoscopic hysterectomy is a common gynaecological treatment that is distinguished by its extensive anatomical complexities and reliance on precise instrument movement inside confined operative spaces. This is one area where such improvements can make a significant difference.

Hysterectomy is one of the most performed surgical procedures in the UK, with an average of 23,056 cases per year [3]. In a laparoscopic hysterectomy, the uterus is removed via a series of tiny abdominal incisions, all under the guidance of a camera-equipped laparoscope. In contrast to open surgery, laparoscopic surgery poses difficulties with visualization, depth perception, and spatial awareness. Surgeons have a narrow field of view; therefore, they must rely extensively on their experience to understand the anatomy and precisely use the surgical instruments [2]. Through the utilisation of image analysis and machine learning, surgeons can gain crucial insights into the surgical scene, hence improving their operational efficiency and decision-making abilities.

However, complex computational techniques and algorithms are needed to achieve robust segmentation in laparoscopic hysterectomy. This is where computer science and healthcare intersect. Computer scientists are essential for developing and enhancing algorithms for machine learning, image processing, and feature extraction that are especially suited to the difficulties presented by surgical environments. Computer vision systems can help surgeons identify anatomical landmarks, track surgical instruments, and optimize procedural workflow by analysing camera feeds and extracting relevant details about the surgical scene. This improvement in the surgeon's perceptual abilities may result in more accuracy, effectiveness, and safety throughout the procedure [1].

1.2 Aims and Objectives

In this final year project, the focus lies on leveraging computer vision for surgical scene understanding specifically tailored to laparoscopic hysterectomy. Through the development and implementation of advanced algorithms for surgical instruments and anatomy segmentation, the objective is to create and analyse a model which can perform image segmentation on surgical images. The system aims to mitigate challenges associated with limited visualization and improve overall procedural performance.

Throughout this report, the methodology, implementation strategies, experimental findings, and future ideas are detailed in the pursuit of enhancing scene understanding via deep learning. By connecting the gap between computer science and surgical practice, the aim is to contribute to the advancement of minimally invasive surgery and pave the way for safer, more efficient, and more effective healthcare delivery.

1.3 Background

1.3.1 Image Segmentation

Image segmentation is a task in computer vision aimed at classifying each pixel in an image into distinct categories based on different attributes such as colour, intensity, texture, or semantic content. Image segmentation turns the unprocessed images into a more organized and comprehensible representation by assigning each pixel to a unique category [4]. This method facilitates subsequent analysis and understanding of the content of the image, enabling tasks such as object detection, and scene understanding. There are different techniques for segmenting images, and each has benefits and drawbacks of its own. These strategies can be broadly divided into two categories: deep learning methods and classical methods [5].

1.3.1.1 Classical Methods

Traditional image segmentation techniques segment an image using established rules and manually created features. These approaches, which include region-based segmentation, thresholding, and edge detection, have been in widespread usage for decades [5].

Classical segmentation methods come with inherent strengths. They are computationally efficient, making them particularly suitable for processing simple scenes. Additionally, their intuitive nature renders them easy to understand and implement. In scenarios with well-defined object boundaries and uniform backgrounds, these methods can be highly effective. However, classical segmentation methods also face certain limitations. They often struggle when confronted with complex scenes, varying lighting conditions, and obstructions, resulting in less accurate segmentation outcomes. Moreover, manual tuning of parameters is necessary, rendering them less adaptable to diverse datasets and environments. Additionally, these methods have limited capability to capture high-level semantic information, which may hinder their performance in tasks requiring detailed scene understanding [5].

1.3.1.2 Deep Learning Methods

Neural networks are used by deep learning-based image segmentation techniques to automatically discover complex patterns in data. They have become highly effective tools for image segmentation jobs, providing cutting edge results across a range of applications.

Deep learning methods exhibit notable strengths. They possess the capability to learn hierarchical representations directly from raw images, removing the need for handcrafted features. Moreover, they demonstrate adaptability to diverse datasets and complex scenes, making them versatile in various applications. Additionally, deep learning models excel at capturing high-level semantic information, resulting in more accurate and robust segmentation results. Deep learning methods also present certain limitations. They require large, annotated datasets for training, which can be costly to acquire. Moreover, these methods are computationally demanding during both training and inference, particularly for deep architectures and large-scale datasets [6].

Two popular approaches within deep learning-based image segmentation are:

Semantic Segmentation: Semantic segmentation assigns a class label for each of the pixels in an image, effectively segmenting the image into distinct regions for different categories [7].

Instance Segmentation: Instance segmentation extends semantic segmentation assigns class labels to each of the pixels but also distinguishes between individual object instances. This finer-grained segmentation integrates semantic segmentation and object detection to identify object boundaries and classify each instance separately [7].

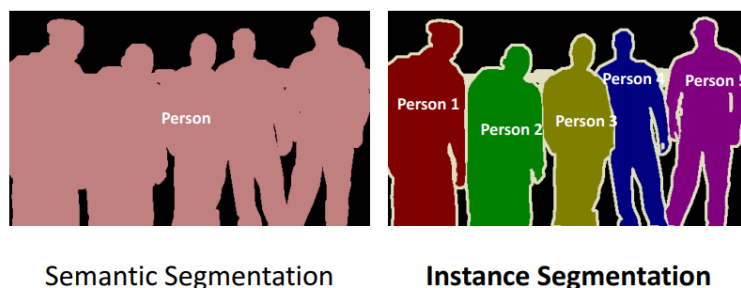


Figure 1 Semantic and Instance Segmentation [8]

In this report, we explore the utilization of supervised deep learning methodologies in the domain of semantic image segmentation, with an emphasis on Convolutional Neural Networks (CNNs) [9], alongside the convolutional encoder-decoder architecture.

1.3.2 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) revolutionized the field of computer vision by providing powerful tools for processing structured grid data, particularly images. These networks are tailored to exploit the spatial locality and hierarchical structure present in visual data, making them ideal for tasks like image segmentation. CNNs typically comprise multiple layers, each serving a specific function in feature extraction and abstraction [9]. The core layers of CNNs include convolutional layers, pooling layers and activation functions.

1.3.2.1 Convolution Layers

Convolution layers are specifically designed to detect patterns and features within localized regions of input data. These layers are made up of filters that slide over the input image, calculating element-wise multiplication and aggregation operations to extract relevant features. Filters perform feature extraction using the input pixels which are spatially close together. By learning filters during training, convolutional layers can identify hierarchical patterns, edges, textures, and shapes present in the input, enabling the network to capture meaningful representations [11].

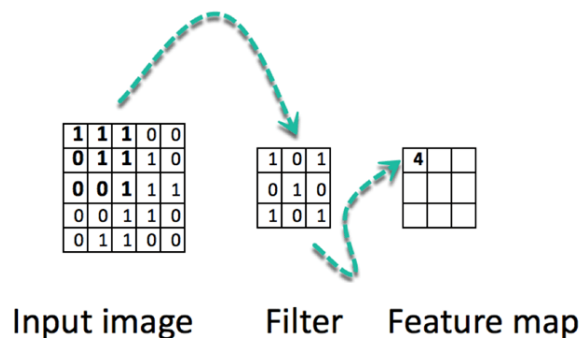


Figure 2 Convolutional Layers [10]

1.3.2.2 Pooling Layers

Pooling layers decrease the spatial dimensions of feature and preserve important information. Typically inserted after convolutional layers, pooling layers combine the regional information by down sampling the input through operations like max pooling or average pooling. This down sampling helps achieve translation consistency, making the network resilient to small variations in the input image[11].

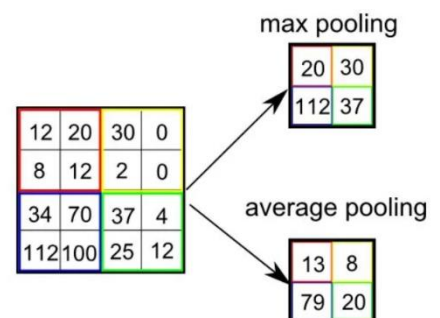


Figure 3 Pooling Layers [11]

1.3.2.3 Activation Functions

In a neural network layer, activation functions are nonlinear transformations that are applied to each neuron's output. They introduce nonlinearity into the model, allowing neural networks to learn complex relationships and patterns. The ReLU activation function is becoming increasingly popular because of how well it mitigates the vanishing gradient issue while remaining straightforward [12]. SoftMax is another popular activation function that is especially helpful for multi-class classification tasks in the output layer of a neural network [13]. SoftMax function normalizes the output of neurons to a distribution over multiple classes, making it suitable for assigning class probabilities. The formulas for both activation functions are shown in Equation 1.

$$\text{ReLU: } f(x) = \max(0, x) \quad \text{SoftMax: } \sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

Equation 1 ReLU and SoftMax Activation Functions [13]

1.3.3 Convolutional Encoder-Decoder Architecture

The convolutional encoder-decoder architecture is an essential framework for semantic segmentation tasks, comprising two essential components: the encoder network and the decoder network. In this design, the encoder network systematically reduces the dimension of the input image and extracts hierarchical features of increasing abstraction through convolutional and pooling layers. This structured feature extraction process enables the encoder to encode complex semantic data about the input image, laying the groundwork for precise segmentation [14]. In contrast, the decoder network, operating in the latter stage of the architecture, restores the feature maps to their original dimensions by up sampling and incorporates contextual information from the encoded features to refine segmentation masks. This hierarchical encoding and decoding process embodies the essence of the convolutional encoder-decoder architecture, allowing the model to capture the local and global information and preserve spatial information, making it effective across diverse domains [14].

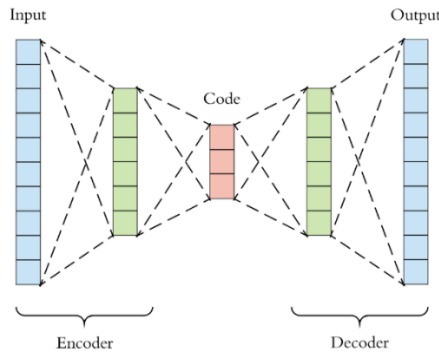


Figure 4 Encoder-Decoder Architecture [15]

1.3.4 U-Net Architecture

An excellent example manifestation of the Convolutional Encoder-Decoder architecture is the U-Net model, known for its adeptness in precise object localization and segmentation. It is a convolutional neural network architecture tailored for biomedical image segmentation [16], although it has found applications in various other domains. Its unique U-shaped architecture which is made up of a reducing chain on the left and an expansive chain on the right, with skip connections between corresponding layers implemented as concatenation operations facilitating precise localization and feature fusion. They enable the network to capture fine-grained details by combining low-level and high-level feature maps from different stages of the architecture. This design facilitates effective propagation of features and preserves spatial information, making U-Net well-suited for tasks requiring precise segmentation, such as medical image analysis and semantic segmentation.

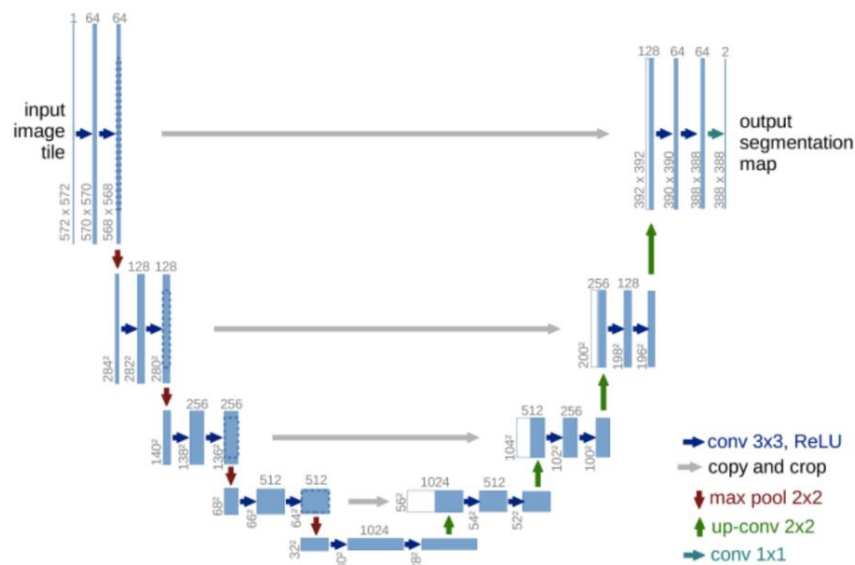


Figure 5 U-Net Architecture [16]

1.3.5 How Neural Networks Learn

Neural networks learn by continually modifying their parameters to reduce a loss function, which calculates the difference between the model's predictions and the ground truth labels. The loss function calculates the difference between the network's predictions and the actual targets during training by using the input data to make predictions [17]. This difference, which is commonly called the "loss," indicates how well the model is doing with the task at hand. The network adjusts its parameters in a way that lowers the loss using methods like gradient descent and backpropagation, essentially "learning" from its errors. Neural networks gradually increase their predictive accuracy and learn to make more accurate predictions on unseen data by exposing the network to training data repeatedly and modifying its parameters to minimize the loss.

1.3.5.1 Loss Functions

In this project, we will explore a selection of loss functions tailored specifically for image segmentation. These loss functions are chosen for their suitability in measuring the difference between predicted segmentations and ground truth masks, thereby guiding the training process of neural networks to achieve accurate segmentation results.

1.3.5.1.1 Cross Entropy

Cross Entropy Loss operates by quantifying the difference between the predicted probability distribution and the ground truth distribution for each pixel in the segmentation map. Mathematically, Cross Entropy Loss is calculated as the negative logarithm of the predicted probability assigned to the correct class label [18]. Specifically, for each pixel, the loss is then calculated by the negative logarithm of the distribution assigned to the ground truth category.

The cross-entropy between two distributions, for example A and B, can be calculated as shown in Equation 4. A is the target distribution and B is the prediction of the target [19]. In the equation, each x in X is a class label.

$$H(A, B) = - \sum_x^X P(x) \times \log (B(x))$$

Equation 4 Categorical Cross Entropy

1.3.5.1.2 Focal Loss

Focal loss emerges as a powerful loss function in the realm of image segmentation. This loss function, introduced by Lin et al [20], for object detection, has gained attention for its ability to mitigate the impact of dominant classes while focusing on the classification of challenging instances. Unlike Traditional loss function, Focal Loss dynamically adjusts the contribution of each of the classes to the overall loss based on its predicted probability. By decreasing the effect of the loss for well classified classes and amplifying it for challenging ones, Focal loss shifts the focus of the model to train toward learning from difficult cases [20]. This approach not only helps mitigate the dominance of frequent categories, but also improves the model's ability to classify rare or challenging instances.

Focal Loss has a unique formula which incorporates two key parameters: Alpha (α) and Gamma (γ). The loss function is defined in Equation 5 [21].

$$FL (p_t) = -\alpha (1 - p_t)^\gamma \times \log (p_t)$$

Equation 5 Focal Loss

Here p_t represents the predicted probability of the correct class label. The term $(1 - p_t)^\gamma$, acts as modulation factor, adjusting the loss contribution based on the predicted probability. The γ parameter, controls the rate at which the loss is reduced for confident classes. A higher value of γ , increases the

focus, prioritizing the classification of challenging instances. Moreover, the α parameter, serves as a weighting factor for each class [21]. By adjusting for different classes, we can configure the loss function to address class imbalance issues. For instance, having a higher alpha for minority classes can increase their effect on the loss, hence balancing the training process and improving the overall performance.

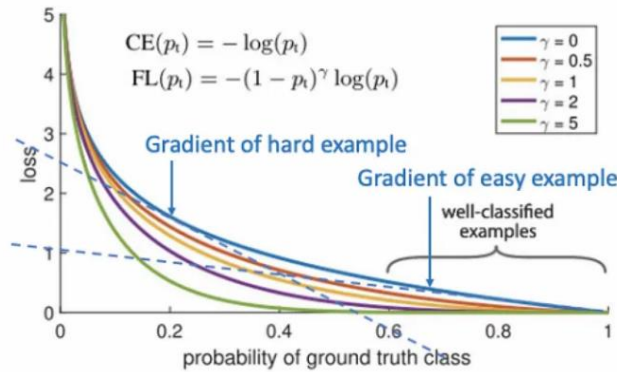


Figure 7 Focal Loss Graph [21]

1.3.5.2 Optimizers

For training deep learning models, optimizers play an important role. They iteratively adjust the model parameters to reduce the cost function. The main aim of optimizers is to find the optimal parameters which gives the lowest possible loss, thus improving model's performance. This process is typically achieved through backpropagation, a crucial algorithm in neural network training. It calculates the gradients of the cost function regarding the model's parameters. These gradients allow the optimizers change the parameters in a way that slowly lowers the loss cost multiple iterations and guiding the network towards a convergence.

1.3.5.2.1 Adam Optimizer

There are many optimizers used in deep learning, however, the Adam optimizer stands out as popular choice for its efficiency and effectiveness. Adam uses momentum, which accelerates the optimization process by calculating and using previous gradients. Momentum allows the optimizer to keep a velocity-like variable, which helps it navigate through the optimization landscape more efficiently [22]. This makes Adam an excellent optimizer for training neural networks on large-scale datasets, where achieving fast convergence is crucial.

1.3.6 Challenges

1.3.6.1 Overfitting

In the image segmentation context, especially with architectures like U-Net, the risk of overfitting is high due to the large number of parameters involved. U-Net, by having numerous layers and feature maps, has a high model capacity capable of tracking intricate details while training. However, having

this large number of parameters, increases the probability of the model memorizing the training set rather than learning generalizable patterns. Hence the model may struggle to generalize properly to unseen information, and it may exhibit poor performance for our test dataset. There are several ways to tackle the issue of overfitting.

1.3.6.1.1 Dropouts

Dropout is technique commonly used in deep learning that helps prevent overfitting by randomly deactivating neurons while training. This encourages neuros to learn more independently and improves the model's generalization ability and reduces the chance of memorizing the training data [23].

1.3.6.1.2 Reducing Number of Layers

Reducing number of layers in depth of a neural network is a simple way to address overfitting. By making the model more simple, fewer parameters are involved, lowering the risk of memorizing the training data. This approach improves generalization and efficiency of the model without losing the model's ability to capture relevant patterns [24].

1.3.6.1.3 Data Augmentation

Data Augmentation is a powerful method for addressing overfitting in our model by artificially increasing the diversity of our training data. By applying a selection of transformations such as rotation, scaling, flipping and cropping to the original images and masks, we are able to introduce new variations that model can learn from. This helps prevent the model from memorizing the specific patterns in the training data and forces the model to learn generalizable patterns [25]. Moreover, data augmentation expands the training dataset, providing the model with more examples for learning without the need for collecting additional images and masks.

1.3.7 Existing Solutions

As part of the background research for our project, we conducted extensive research of existing solutions for surgical tools and anatomy segmentation in the context of laparoscopic hysterectomy. However, there were no specific solutions tailored to this precise application. Despite this, we found relevant studies focusing on anatomy segmentation using very complex models such as *DeepLab* [26] and *Segformer* [27]. These models are both state-of-the-art models designed for generic semantic segmentation tasks. DeepLab was created by the research team at Google in 2016 [26] and it is based on a deep convolutional neural network. It has been widely used in various domains, including medical imaging as well. Segformer was recently introduced by the research team at ETH Zurich in 2021 [27]. Instead of using traditional convolutional neural network architecture, Segformer uses transformer, which has gained a lot of popularity in natural language processing tasks. For this project we are focusing on building a much simpler architecture from the ground up tailored to our dataset.

1.3.8 Technologies Used

1.3.8.1 Keras and TensorFlow

For this image segmentation project, we utilized a suite of powerful tools to implement and optimize the model. Leveraging the capabilities of Keras [28] and TensorFlow [29], industry standard deep learning frameworks, provided us a robust environment for developing and training our convolutional neural networks (CNNs).

1.3.8.2 Weights and Biases

We decided to use the capabilities of Weights and Biases [30] for better tracking and evaluating our experiments. With Weights and Biases, we are able to gain valuable insights into the impact of various model changes, improving the process of informed decision making and iterative enhancements.

Chapter 2 – Methodology

2.1 Dataset

2.1.1 Dataset Acquisition

Dataset acquisition is an important step in any machine learning project, as the quality and diversity of data can drastically impact the model's performance. In context of image segmentation for surgical scene understanding, finding relevant datasets can be a challenge due to the specialized nature of the domain and the time and resources required for annotating the masks. After thorough investigation, one dataset that stood out for laparoscopic surgery was the AutoLaparo dataset. This dataset, created by researchers at the University of California, Irvine [31], consists of a collection of annotated laparoscopic videos and images captured from 21 hysterectomy surgeries.

The AutoLaparo Task 3 dataset offers a rich collection of 1800 images and corresponding masks, all with a high resolution of 1920 by 1080 [31]. These annotated images provide valuable insights into laparoscopic hysterectomy procedures, with annotations both for surgical instruments and anatomy of the uterus. The surgical instruments include common tools for this procedure: Grasping Forceps, LigaSure, Dissecting and Grasping Forceps, and Electric Hook.

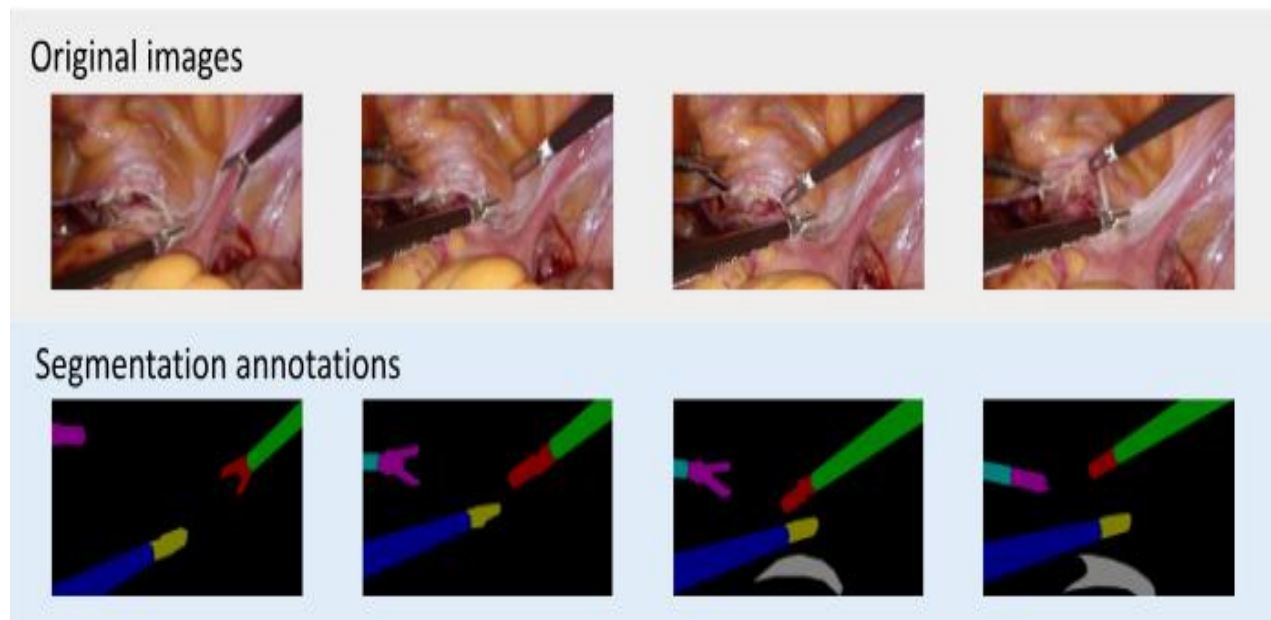


Figure 8 Dataset Examples [31]

2.1.1 Dataset Evaluation

As shown in Figure 9, the dataset exhibits class imbalance, with certain classes containing fewer instances compared to the rest. This poses challenges for training our model as it may exhibit a bias for the majority classes and struggle to classify and segment the minority categories. For example, the number of instances of labelled pixels for the class “Electric Hook (manipulator)” only takes up about 0.2% of all instances and compared to the more dominant categories like “Uterus” or “Background” it may potentially lead to a diminished segmentation performance. It is crucial for us to address the class imbalance to reduce the biases and ensure robust performance across all classes.

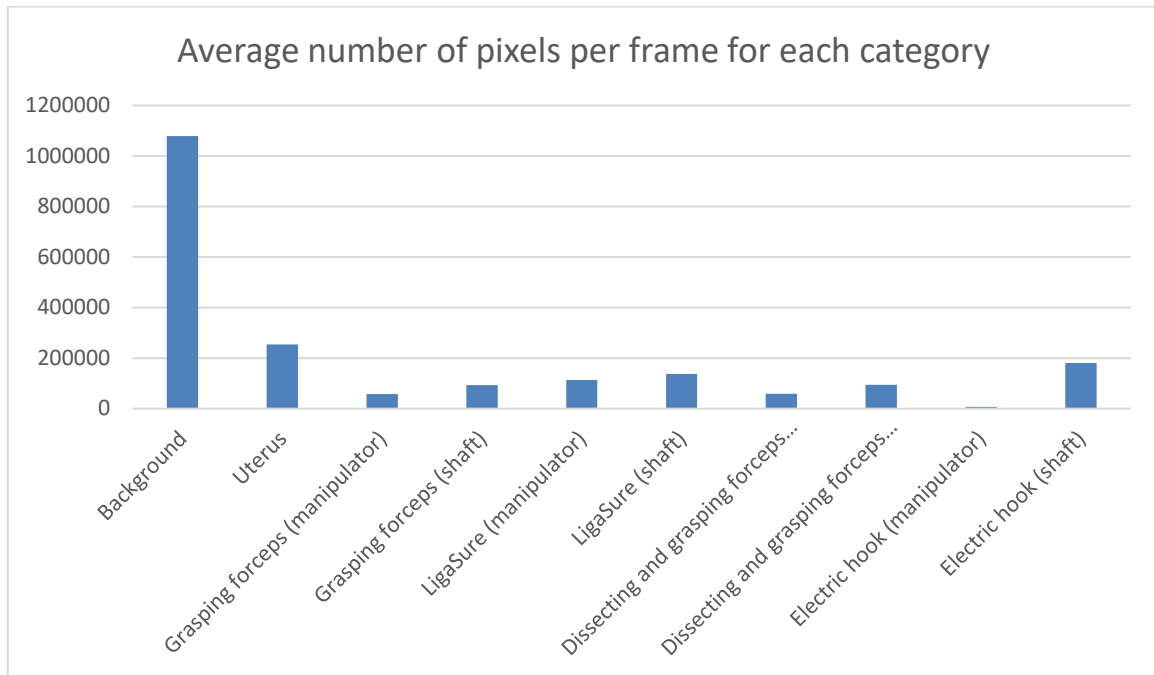


Figure 9 Dataset Analysis

2.2 Data Preparation

For the data preparation phase of our project, several processing steps were undertaken to ensure suitability and compatibility for training our model. These steps include:

2.2.1 Converting RGB Values to Float

The images in the AutoLaparo dataset are in RGB format, with pixel values ranging from 0 to 255 for each of the colour channels (Red, Green, Blue). To facilitate numerical computation, the RGB values were converted into numerical array, where each pixel value represented the intensity of the corresponding colour channel.

2.2.2 Resizing Images and Masks

Due to the high resolution of the images and masks in the dataset (1920 x 1080), training the model would be a significant computational challenge. To mitigate this and enhance the efficiency, both images and masks were resized to a smaller resolution of 256 by 256 pixels. This process aimed to

reduce the computational load while not losing any valuable information from the images and corresponding masks.

2.2.3 Standardizing Mask Data

In order to train our segmentation model, we had to adjust the label values in the masks data. Originally, the pixel values in the masks were multiples of 20 [31], such as 0, 20, 40 and so on with each number corresponding to a certain class. However, to simplify the segmentation tasks and ensure compatibility with our model, we divided these values by 20. As a result, the labels for each of the classes were transformed into consecutive integers such as 0, 1, 2, 3 and so forth. This allowed us to treat the segmentation problem as a multi class classification task.

2.2.4 Converting Mask Data to One-Hot Encoded Format

Finally, the masks were encoded in a one-hot encoded format to prepare them for training our model. In this encoding format, each pixel in the mask is represented by a binary vector, where each entry corresponds to a class label. The value 1 is assigned to the index corresponding to the class of the pixel, while other entries are set to 0. This representation provides clear class distinction for each pixel in the mask.

By performing these processing steps, we ensured that our training data is well prepared and standardized for training our model, laying the foundation for effective model development and evaluation [32].

2.3 Train, Test, Validation

To effectively evaluate the performance of our segmentation model, the dataset consisting of 1800 images and corresponding annotated masks was divided into distinct sets for training, validation and testing. 70% of our dataset was allocated to the training set, 15% to the validation set and the remaining 15% for the test set. This division strategy allows for comprehensive model training on a large portion of the data, while the validation and test sets serve as independent benchmarks for performance assessment. To maintain consistency and reproducibility in our experiments, the division of the data was performed using a random seed to ensure the division of data can be reliably recreated for future comparisons. By using this approach, we aim to develop a robust model for segmentation which can effectively generalize to unseen data and perform well in real-world applications [33].

2.4 Data Visualization

To visualize images alongside their corresponding masks after the changes that we had made in the processing steps, we used the argmax function. Given that now the masks are in a one-hot encoded format, we apply the argmax function along the class axis and determine the index of the maximum value for each pixel [34], effectively identifying the class label most suitable for that pixel. This

visualization process was implemented using Matplotlib [35], enabling us to display images and masks side by side.

$$\text{argmax}([0.1, 0.2, 0.6, 0.1]) = 2$$

Equation 6 Argmax Example

2.5 Data Augmentation

In the purpose of enhancing the robustness and generalization capabilities of our model, we employed data augmentation to our pipeline. Using augmentation techniques, we effectively expanded the diversity of our training dataset, hence exposing the model to various scenarios. Our data augmentation strategy consisted of several transformations to diversify the training data. We implemented functionality to randomly flip the images horizontally and/or vertically. We also added random rotations to the images and introduced variability in brightness levels to simulate different lighting conditions. By doing so we effectively doubled the amount of training data and provided the model with a richer and more diverse set of training examples.

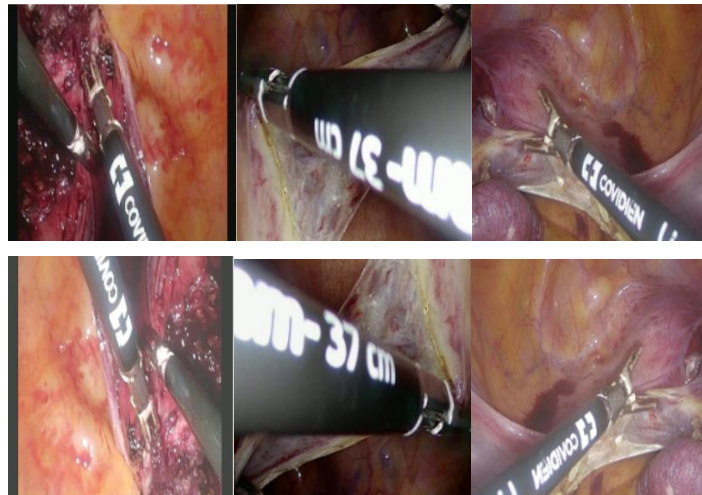


Figure 10 Data Augmentation Examples

2.6 Technology and Code Setup Organization

Given the significant computational time and resources required for training deep neural networks, we decided to leverage the computational capabilities of a dedicated 1050 Ti GTX Nvidia GPU [36]. This GPU served as a crucial asset in increasing the efficiency in model training and iteration cycles. To comprehensively keep track and analyse various training parameters, we used Weights and Biases, a versatile experiment tracking platform. Weights and Biases enabled us with real-time monitoring and visualization of model performance and training progress. Moreover, to ensure systematic code management and version control, the code was organized and uploaded to GitHub [37]. Allowing the code base to be structured and accessible, increasing the efficiency of development progression.

2.7 Building the Model

For this project, we employed a simple architecture as the foundation for our segmentation model inspired by the U-Net architecture [16]. We leveraged the Keras framework and its prebuilt functionalities for the design and construction process. The implementation of the model involved the integration of convolutional layers, ReLU activation functions, dropout layers and max pooling operations.

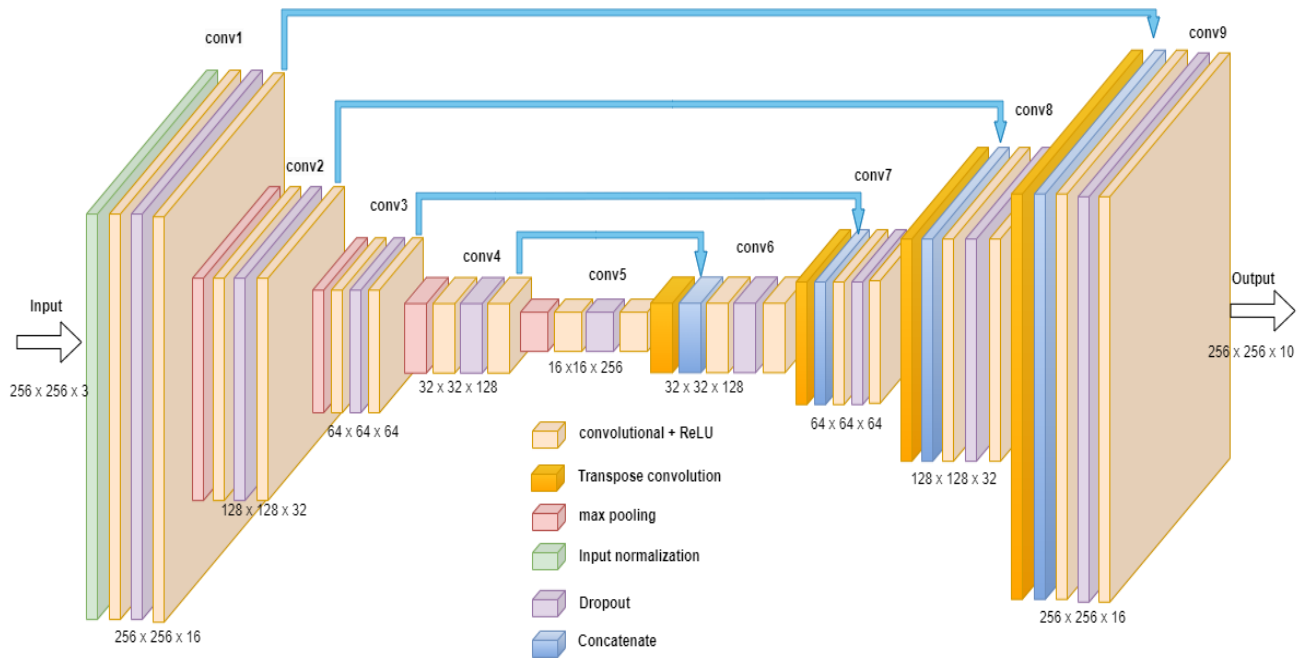


Figure 11 Model's Architecture

2.7.1 Model's Architecture

The architecture, as illustrated in Figure 11, was designed to capture intricate details within the input images and accurately identify surgical instruments and anatomy of the uterus. Commencing with the input images of dimensions 256 by 256 by 3, with 3 representing the standard RGB channels, the model goes through a series of transformational layers. The initial layer, performs input normalization by scaling the pixel values to a range between 0 and 1, hence facilitating a smoother convergence during the training process. Afterwards, a convolutions layer with 3 by 3 filters and a feature size of 16 is applied, resulting in feature maps of dimensions 256 by 256 by 16.

Following this a dropout layer is introduced to solve the overfitting issue by randomly deactivating a fraction of neurons while training. This process iterates through additional convolutional layers and max pooling operations, reducing the spatial dimensions while increasing the depth of the feature maps. This feature extraction process, finally results in a compact representation of the input image, captured in feature maps of dimensions 16 by 16 by 256 at the lowest level.

To continue with the segmentation process, transpose convolutional layers are implemented to scale up the feature maps, gradually restoring the resolution while reducing the depth of the feature maps.

This up sampling operation is complemented by concatenation with feature maps from earlier convolutional layers as shown in Figure 11, enabling the model to integrate both low-level and high-level features for more accurate segmentation.

The iterative process of convolutional, dropout and transpose convolutional layers continues until the resolution of the prediction matches the original input resolution of 256 by 256. The final convolutional layer equipped with the ReLU activation function, produces segmentation prediction with feature dimensions aligned with the number of segmentation classes, in this case 10.

Given that the segmentation masks predicted are encoded in a one-hot format, the SoftMax activation function emerged as the logical solution for the final layer. By applying the SoftMax activation, we ensured that the model's output probabilities were normalized across all classes, with each value indicating the pixel's likelihood of belonging to a specific class.

2.7.2 Different Loss Functions for The Model

For having an accurate optimization for the segmentation model, a curated selection of loss functions was assembled to distinguish the most effective configuration for enhancing the model's performance using the Adam optimizer. For this approach, we leveraged built-in loss functions within the Keras framework along with implementing custom loss functions tailored to our specific requirements.

2.7.2.1 Normal and Weighted Categorical Cross Entropy

The built-in "categorical_crossentropy" loss function from Keras was employed for the initial model compilation. This standard loss function served as the baseline for comparisons in order to improve the model's performance.

Additionally, we explored the application of weighted categorical cross entropy, a version devised to solve the class imbalance issue within the dataset. By assigning different weights to each of the categories based on their frequency, we tried to mitigate the effects of the class imbalance issues and refine the model's performance.

Determining the optimal weights for the loss function could be challenging [39]. We decided to use the actual frequencies calculated before while analysing the dataset as a starting point and through a trial-and-error approach, we adjusted these weights promoting accurate segmentation while mitigating the effects of the class imbalance. For this process we implement our own weighted version of the cross entropy which demanded careful consideration and extensive experimentation to achieve optimal performance.

2.7.2.2 Focal Loss

A custom implementation of the focal loss function was developed in order to solve the issue of class balancing and emphasising more on challenging examples in the dataset. Adjustments to the alpha and gamma parameter were made to try to tailor the loss function to focus the model's attention on the

difficult segmentation in the dataset hence, trying to improve the model's robustness and generalization abilities.

2.7.3 Adaptable Learning Rate

Learning rate is a crucial hyper parameter for any deep learning model. Setting it to low would cause the model to converge very slowly which not efficient. And setting the learning rate too high, would cause the model to never be able to find the minimum of the model. A good approach would be to start with a high learning rate and as the model learned the basic patterns, we decrease the learning rate so the model would be able to find the specific patterns as well, hence decreasing the overall loss [40]. For this model the learning initially was set to 0.001 which is a standard learning rate for the Adam optimizer and if there were no improvements to the validation loss over five consecutive epochs, the learning rate would decrease by 80% each time. This mechanism allows the model to navigate through potential plateaus or local minima more effectively and the model could potentially escape from suboptimal solutions. This approach also helped to prevent overfitting and increase the model's generalization performance by encouraging a smoother convergence.

2.7.3 Early Stopping

Another popular technique in deep learning is early stopping [41]. This mechanism allows the model to stop if there are no visible improvements to the loss function potentially reducing the overfitting of the model. For this project we decided to implement this feature as well so if there are no improvements to the validation loss after 10 consecutive epochs, the model would stop training as it indicates the model has reached its limit and any further training would be causing the model to overfit to the training data.

2.8 Evaluation Metrics

2.8.1 Pixel-Wise Accuracy

Evaluating image segmentation models is crucial for guaranteeing their efficiency across various applications. Pixel-wise accuracy is a basic evaluation metric in image segmentation models, with a measure of the model's ability for classifying individual pixels in an image. To compute pixel wise accuracy, each of the pixels in the predicted mask is compared to the corresponding pixel in the original annotation. Pixels that match the target are considered true positives, while pixels which are misclassified, are counted towards false positives and false negatives [42]. The pixel wise accuracy is then calculated by dividing the number of correctly classified pixels by the total number of pixels in an image, in this case 65536 ($256 * 256$).

While pixel-wise accuracy metric provides information about overall correctness, it often disregards crucial factors such as spatial precision and semantic consistency. Therefore, relying solely on accuracy is insufficient. It is essential to use a variety of evaluation metrics that consider spatial

overlap, and class-wise performance. This thorough strategy provides greater understanding of the strengths and limitations of our segmentation models, allowing educated and logical decision-making [42].

2.8.2 Intersection Over Union (IoU)

Intersection over Union (IoU): computed as the intersection area divided by the union area, IoU quantifies the overlap between the ground truth and the segmentation mask [43]. It is commonly used to evaluate the performance of segmentation models.

Mean Intersection over Union (mIoU): mIoU measures the average IoU across all classes, providing a broad evaluation of segmentation results across different categories. It is especially beneficial for multi-class segmentation tasks.

$$mIoU = \frac{1}{N} \sum_i^N IoU(i)$$

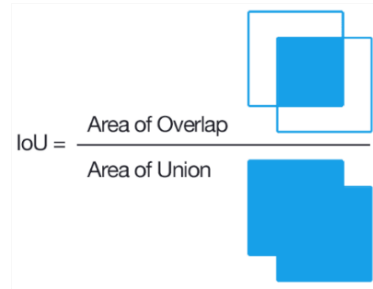


Figure 6 Intersection over Union [43]

2.8.3 Dice Coefficient

Dice Coefficient (DICE): The Dice coefficient, measures the similarity between the predicted segmentation and the ground truth mask by computing the intersection and dividing it by the sum of the distributions. It measures the spatial similarity between the two masks and is widely used in medical image segmentation and other fields requiring precise delineation of object boundaries [44]. If the Dice coefficient value is one, there is a perfect match between the prediction and ground truth.

$$DICE = \frac{|A \cap B| + 1}{|A| + |B| + 1}$$

Equation 2 Dice Coefficient

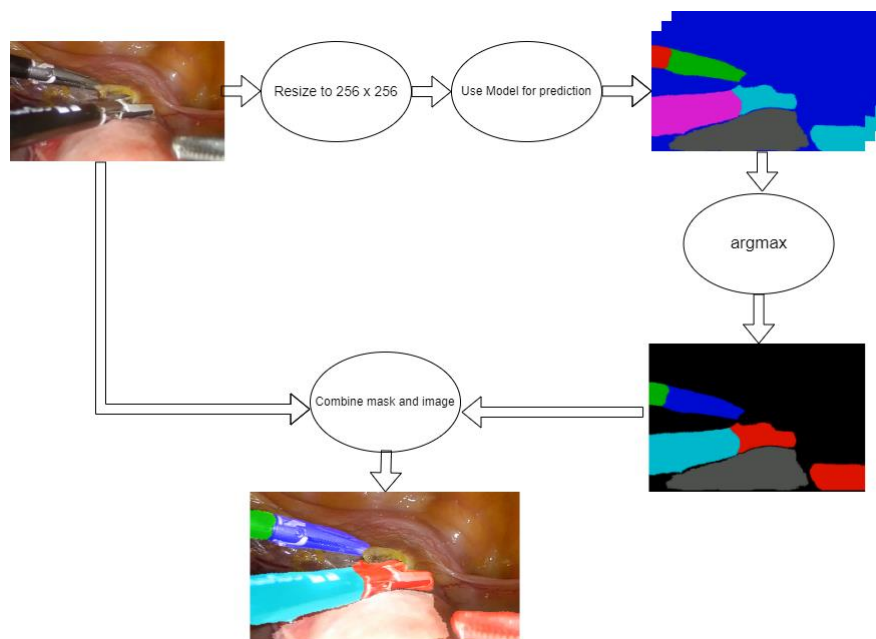
Mean Dice Coefficient (Mean Dice): Similar to mIoU, Mean Dice calculates the average Dice coefficient across all classes, offering a thorough assessment of segmentation performance across different classes. It provides valuable insights into the model's ability to accurately capture object boundaries and semantic information.

$$mDICE = \frac{1}{N} \sum_i^N DICE(i)$$

Equation 3 Mean Dice Coefficient

2.9 Prototype Implementation

With having a refined architecture for our segmentation model and having defined our variable and hyper-parameters for our subsequent experiments, we shifted our focus towards a practical implementation aimed at deploying the model for real-world surgical applications. For this task, we created a Python application/script designed to smoothly integrate our segmentation model with laparoscopic hysterectomy surgical images of any dimensions. The application is capable of accepting images and generating predictions in real-time. Upon inputting the surgical images, the application firstly resizes the image to 256 by 256 pixels and then loads the segmentation model. Afterwards, the image is put into a batch and the results are predicted from the model. The prediction result will be a 256 by 256 by 10 array and using the argmax function, the result is transformed to a 256 by 256 array with entries ranging from 0 to 9, with each number indicating a unique class. The application then converts these values to pre-defined colours and a RGB mask overlay of 256 by 256 by 3 is created as a result. The produced mask is now upscaled to match the original resolution of the image and it is drawn on top of the original image using an opacity of 80 percent for better visualizations. The resulting image is then saved and showed to the user.



Prototype Architecture

Subsequently the application was extended to accommodate the segmentation of surgical videos as well. With the help of the Numpy [45] and OpenCV [46] libraries, we created a seamless workflow for processing video sequences, with the additional ability for viewing the segmentation predictions in real time. This real-time capability holds significant potential for applications in live surgical context, where surgeons can benefit from immediate visual feedback.

Chapter 3 – Results and Experiments

3.1 Experiments Setup

The experiments were conducted using the TensorFlow framework and leveraging an NVIDIA GTX 1050 Ti graphics card. The training process incorporated adaptive learning rate techniques and early stopping as discussed earlier. To systematically monitor and evaluate the experiments, a custom dashboard was created using Weights and Biases platform, enabling us to comprehensively track the training progress, model performance metrics and visualize different graphs, facilitating efficient analysis and comparison of the results for different experiments. This step helped us by providing a solid foundation for subsequent analysis of following experiments.

3.2 Experiments

To have a baseline for our experiments, the initial run used the built-in categorical cross entropy function within Keras framework. The overarching objectives of the subsequent experiments is to enhance the performance of the segmentation of the model relative to this baseline. By introducing modifications to the model, the aim is to iteratively refine the performance and achieve better segmentation results. Through comparison with the baseline results, the efficacy of each modification can be assessed using the segmentation metrics introduced earlier.

Table 1 Base Cross Entropy Results

Model	mIoU	Mean DICE	Pixel Accuracy
Base cross entropy	0.5897	0.6478	0.9351

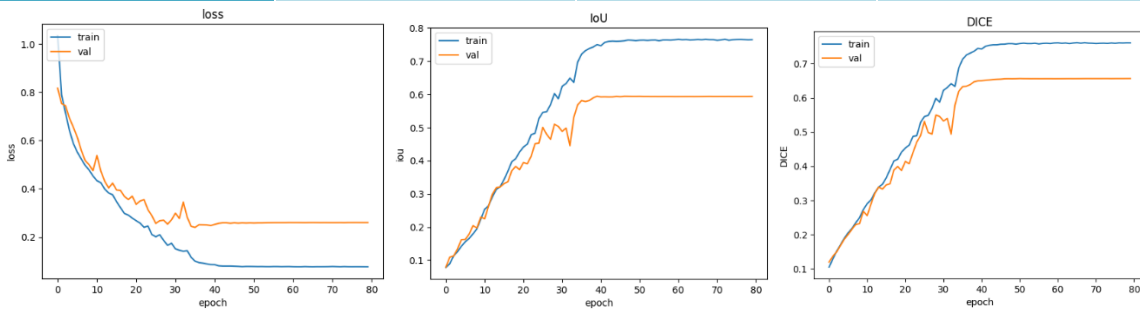


Figure 12 Base Cross Entropy Results

As seen from the results in Figure 12, the training metrics, including Mean IoU and Mean DICE score exhibit consistent improvement over epochs, however a noticeable gap emerge between the training and validation results after about 30 epochs. This discrepancy suggests the presence of overfitting, showing the model is excessively tailored to the training data, hindering its ability to generalize

testing and validation data. This can also be seen with the loss graph as after the 30 epochs there is no visible improvements to the validation loss.

3.2.1 Adding Data Augmentation

In this next experiment, we decided to apply the data augmentation techniques discussed earlier to the training dataset to address the overfitting observed in the initial run. By applying the data augmentation, our goal was to diversify the training dataset and expose the model to a wider range of images.

Table 2 Augmented Cross Entropy Results

Model	mIoU	Mean DICE	Pixel Accuracy
Augmented cross entropy	0.6604	0.7042	0.9478

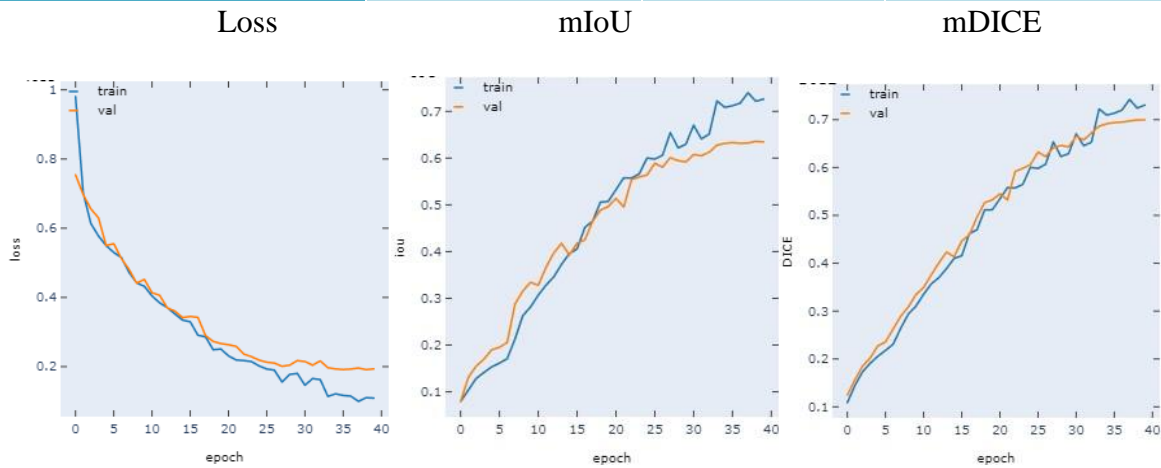


Figure 13 Augmented Cross Entropy Results

Following the addition of data augmentation and subsequent retraining of the model, noticeable improvements in segmentation performance were observed. The Mean IoU (mIoU) increased from 59% to 64%, along with the DICE coefficient score rising from 65% to 70%. These improvements signify the efficacy of data augmentation steps in enriching the dataset and improving better learning of diverse patterns present in images. Moreover, the analysis of the training and validation curves shown in Figure 13 revealed a noticeable reduction in the gap between the respective metrics. This reduction shows a significant importance in reducing the overfitting of the model and demonstrates improved generalization capabilities.

While the overall performance metrics of the model, including mIoU and DICE score, indicate promising results comparable to other similar papers in this field [47]. A more detailed analysis of categorical performance reveals certain issues. The segmentation model exhibits significant difficulties in accurately segmenting some of the minor categories, as evidenced by the results shown in Table 3. For example, the class “Hook-M” demonstrates a particularly low IoU score of only 3% and a DICE coefficient score of 8%. This discrepancy can be attributed to the class imbalance within

the AutoLaparo dataset, as previously discussed. Following experiments aim to address this class imbalance issue and improve model’s performance across all categories.

Table 3 Class Specific Segmentation Metrics

Category Metrics	Background	Grasping-M	Grasping-S	LigaSure-M	LigaSure-S	Dissecting & Grasping -M	Dissecting & Grasping -S	Hook-M	Hook-S	Uterus
IoU	0.9461	0.5900	0.6999	0.7653	0.8371	0.6152	0.6646	0.0315	0.7356	0.7190
DICE	0.7042	0.6279	0.6756	0.8307	0.8877	0.6720	0.7732	0.0826	0.7297	0.7967

3.2.2 Using Focal Loss

In this experiment, focal loss was employed as the cost function to address the class imbalance issue observed in the segmentation model. Two configurations of the focal loss function, A and B, were tested to evaluate their effects in improving the performance compared to the categorical cross entropy cost function. For configuration A, we utilized alpha and gamma values of 0.3 and 3.5 respectively. These values were suggested as optimal parameters for general focal loss from a paper published by Wang et.al. [49]. For configuration B, we employed the original alpha and gamma values of 0.25 and 2.0 which was introduced and used in the original paper introducing focal loss [20].

Table 4 Focal Loss Results

Model	mIoU	Mean DICE	Pixel Accuracy
Focal Loss A	0.6333	0.5061	0.9302
Focal Loss B	0.6401	0.5679	0.9389

Category Metrics	Background	Grasping-M	Grasping-S	LigaSure-M	LigaSure-S	Dissecting & Grasping -M	Dissecting & Grasping -S	Hook-M	Hook-S	Uterus
IoU (A)	0.9255	0.5462	0.6185	0.7210	0.8290	0.6183	0.6711	0.1275	0.7335	0.5426
IoU (B)	0.9359	0.5256	0.6486	0.7245	0.8378	0.6222	0.6861	0.0525	0.7295	0.6386

The results of the experiments with focal loss as the cost function, indicated mixed outcomes in terms of segmentation performance. While both A and B configurations showed marginal decreases in mIoU, the DICE coefficient experienced a significant decline, especially in configuration A. Despite this, configuration B showed a notable higher Mean DICE score compared to A. Overall neither of the variations effectively addressed the class imbalance problem, as evidenced by the minor improvements in the individual IoU score for each of the classes in Table 4.

We were expecting the focal loss to perform significantly better than cross entropy based on the article published by Nayak 2022 [49], however, the performance in improving segmentation accuracy fell short of expectation compared to the baseline categorical cross entropy results. One potential explanation for this could be the failure to identify the optimal alpha and gamma values suitable for

our dataset. The absence of fine tuning may have limited its efficacy in solving the class imbalance issue. Another plausible explanation could be the manual implementation of the focal loss function. Given the complexity of its implementation, it’s possible that TensorFlow was not able to accurately differentiate our custom cost function which would lead to suboptimal training outcomes.

3.2.3 Using Weighted Categorical Cross Entropy

We then experimented with weighted cross-entropy cost functions; we explored two configurations. Configuration A utilized the pre-calculated class weights based on the frequency of the classes derived from earlier analysis. While the results showed a slight improvement compared to the baseline, the enhancement was not significant. Recognizing the potential for further improvement, we proceeded to develop custom weights through a trial-and-error approach for configuration B. This iterative approach helped us to develop the cost function more efficiently and specifically for our dataset. While the improvements for A were modest, the refinement in configuration B gave us more significant improvements, proving the importance of customization in optimizing segmentation performance.

Table 5 Weighted Cross Entropy Loss Results

Model	mIoU	Mean DICE	Pixel Accuracy
Weighted Cross Entropy (A)	0.6249	0.6815	0.9103
Weighted Cross Entropy (B)	0.6865	0.7232	0.9440

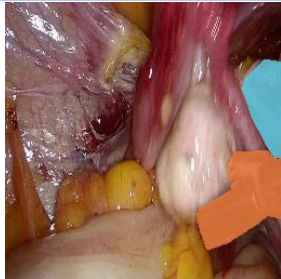







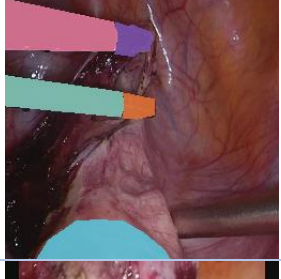
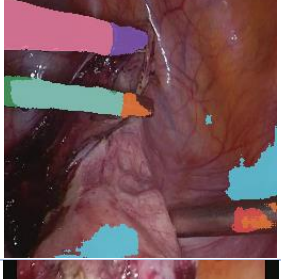
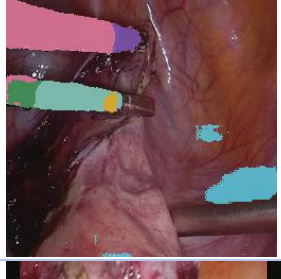
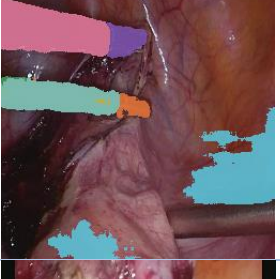


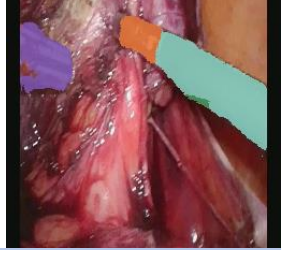
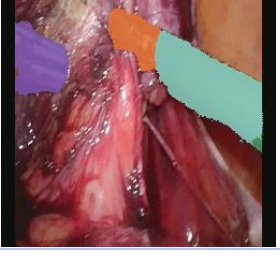
Category	Background	Grasping-M	Grasping-S	LigaSure-M	LigaSure-S	Dissecting & Grasping -M	Dissecting & Grasping -S	Hook-M	Hook-S	Uterus
Weights (A)	0.1923	3.6352	2.2406	1.8258	1.5094	3.48438	2.19621	29.4462	1.1526	0.8164
IoU (A)	0.8998	0.5681	0.7137	0.6579	0.7943	0.5519	0.6684	0.1354	0.6715	0.5880
Weights (B)	0.1	0.5	0.5	0.25	0.25	0.5	0.5	0.8	0.3	0.25
IoU (B)	0.9413	0.5950	0.6731	0.7366	0.8165	0.6759	0.6448	0.2861	0.7310	0.7244

Upon analysing the results presented in Table 5, it becomes clear that Configuration B for the weighted cross entropy has successfully enhanced the issue with category imbalance. Notable in performance improvements for the “Hook-M” category. With the IoU score improving from a mere 3% originating from the baseline cross entropy cost function to over 28%. This configuration has shown a remarkable enhancement in properly delineating the minor classes. Furthermore, the model’s performance for overall segmentation as indicated by mIoU and Mean DICE, has not only been maintained, but it has also seen a considerable improvement of approximately 2% for each of the metrics. This improvement signifies that the configuration B of the weighted cross entropy is the most optimal iteration of our model by properly addressing the class imbalance issue while enhancing the overall performance.

3.3 Results Visualization

After visually evaluating the test set using three different models, the baseline cross entropy cost function, the best performing focal loss model(Configuration B) and the optimally weighted cross entropy cost function(Configuration B), our findings matched with our earlier metric-based conclusions. After comparing all 150 test images against ground truth annotations, the results firmly confirmed that the weighted cross entropy (Configuration B) model outperformed the rest. A few of the results are shown along with the ground truth in Table 6 confirming our conclusion.

Table 6 Segmentation Predictions for Different Models

Ground Truth	Loss Function Experiments		
	Normal Cross Entropy	Focal Loss	Weighted Cross Entropy
			
			
			
			

3.4 Final Prototype

3.4.1 Prototype Evaluation

With Weighted Cross Entropy B, being our best model, we decided to use it for our final prototype. In Table 7 we have shown the evaluation results of the final prototype vs the base model on training, validation and testing dataset to show the improvements which we were able to achieve through iterative experiments.

Model	Training Dataset			Validation Dataset			Testing Dataset		
	mIoU	mDICE	Accuracy	mIoU	mDICE	Accuracy	mIoU	mDICE	Accuracy
Base Model	0.7645	0.7610	0.9722	0.5939	0.6566	0.9397	0.5897	0.6478	0.9351
Final Prototype	0.7613	0.7659	0.9551	0.6729	0.7204	0.9438	0.6865	0.7232	0.9440

Table 7 Final Prototype Evaluation

The results achieved from the final prototype shows significant improvements compared to the base model across validation and testing datasets. In terms of Mean Intersection over Union, the prototype achieved a considerable increase from 0.5897 to 0.6856, an improvement of over 15%, indicating superior segmentation performance. This enhancement is further justified by the mDICE scores, which rose from 0.6478 to 0.7232 for the testing dataset. However, there was not much improvement for pixel-wise accuracy, from 0.9351 to 0.9440, but as explained earlier, pixel-wise accuracy is not the best metric for evaluating segmentation models as it does not account for spatial correctness.

Analysis of the data also reveals a significant issue of overfitting in the base model, as discussed earlier. In contrast while overfitting persists to some extent for the final prototype, improvements are observed compared to the base model. The final prototype still shows higher performance metrics for the training dataset, mIoU of 0.7613 compared to 0.6865, the performance gaps are significantly narrower.

In addition to the overall segmentation performance evaluation, we also conducted an analysis of Intersection over Union for individual classes, considering the class imbalance existing in the dataset. Based on the results in Figure 14, it is evident that the final prototype model shows improvements across all categories. This improvements in performance are particularly more significant for categories that exhibited poor segmentation IoU in the base model.

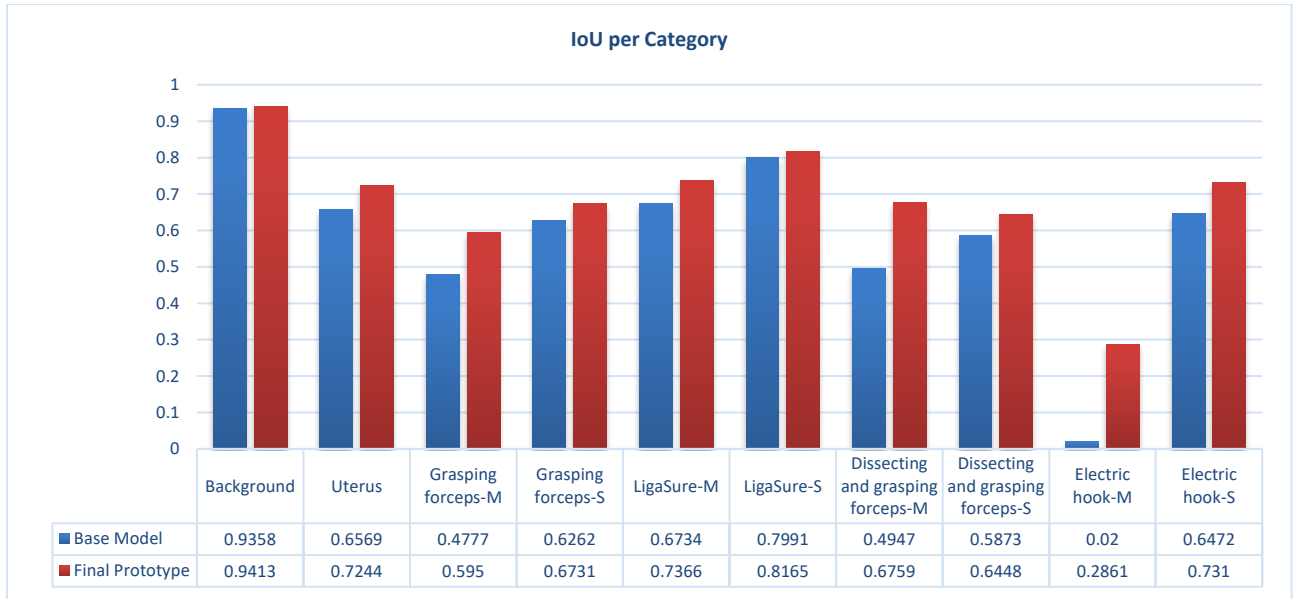


Figure 14 IoU per Category

Although the class imbalance issue is improved significantly, the performance is still unacceptable for some of the categories as the IoU score for 3 of the classes is still below 65%. These classes will not have as good of a segmentation result compared to the rest. We were not able to improve this class imbalance any further by customizing the weights for the model indicating the only way to improve their performance, is by adding more augmented images to the training database which includes these underrepresented categories.

3.4.2 Prototype Limitations

Underrepresented Classes: One limitation of the prototype is with the segmentation of the minor classes. For these predictions the boundaries appear to lack clarity. These classes may exhibit significantly more inconsistencies and irregularities compared to the more dominant classes.

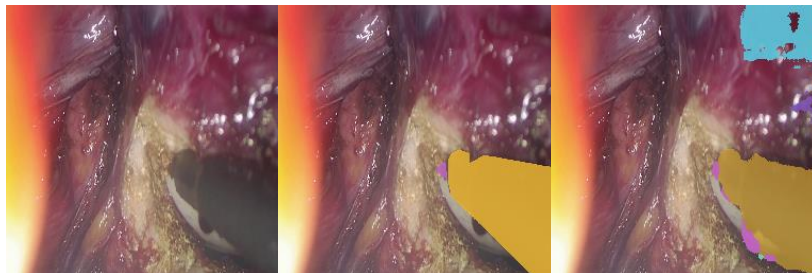


Figure 15 Underrepresented Category Prediction Example

Smoke and Low Brightness: Scenarios involving smoke or low light conditions are very common in laparoscopic hysterectomy surgical videos. Unfortunately, our model seems to struggle in cases with the presence of smoke or low illumination, which leads to ambiguity and uncertainty in the segmentation results.

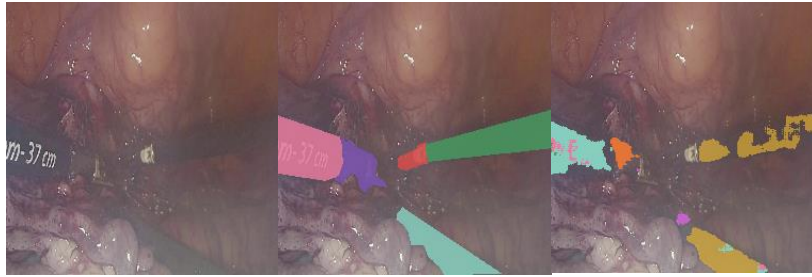


Figure 16 Prediction for Low Light and Smoke

Proximity and Partial Visibility: When the surgical tools come into close proximity to the camera or are partially visible, the segmentation model may face difficulties in properly segmenting the tools' boundaries or categories. The proximity to the camera can obscure feature of the surgical tools, resulting in challenges for our segmentation model.

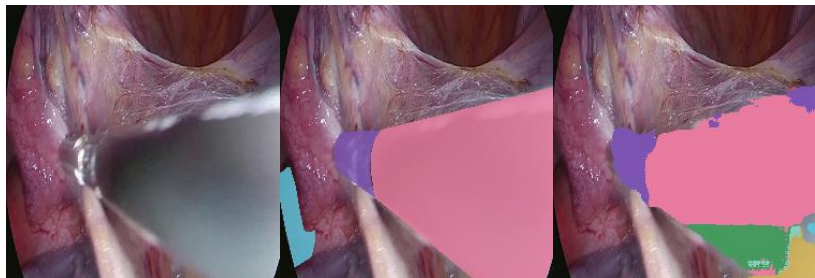


Figure 17 Prediction for Tools Close to The Camera or at The Edge of the Screen

3.5 Segmentation on Real-Time Video

In an experiment to extend our prototype segmentation model beyond segmenting static images, we created a python script to segment video footage, despite the design of the model being primarily for image segmentation. We used the videos from task 1 in AutoLaparo dataset. The annotated dataset which we used to train our model was also used by selecting frames from these videos. It is important to note that video segmentation was not the primary aim for this project, making this experiment an additional endeavour.

This endeavour yielded interesting insights into both limitations and performance of our prototype. With an average processing time of 28 milliseconds for each frame, our model achieved an acceptable video feed rate of 36 frames per second on a moderate GPU (NVIDIA GTX 1050 Ti). This shows significant potential for better latency performance on more powerful hardware configurations.

However, this experiment also highlighted significant limitations. The lack of temporal context within the model, caused a noticeable jittering in the segmentation output, indicative of the model's lack of "memory" for consecutive frames. The use of Recurrent Neural Networks (RNNs) [50] could potentially address this limitation by introducing smoother segmentation transitions between frames. Moreover, the relatively small size of our dataset comprising of only 1800 images, acted as a limiting factor, indicating the need for larger and more diverse datasets to improve the performance of our model for video segmentation.

Chapter 4 – Discussion

4.1 Conclusion

Our project has demonstrated the possibility of developing a segmentation model using a basic architecture inspired by U-Net for surgical scene understanding. Through a series of experiments focusing on optimizing the model, including data augmentation and explanation of different cost/loss functions, we successfully came up with a refined prototype. The final iteration was able to achieve acceptable segmentation performance, with a mIoU of 69% and a DICE coefficient score of 72%. These results show the potential of leveraging deep learning models for improving surgical scene understanding and laying the groundwork for future enhancements in the field of minimally invasive surgeries.

While our project contributes valuable insights into surgical segmentation for laparoscopic hysterectomy, it's important to mention that these results are not as great as the leading results within the broader landscape of surgical tools and anatomy segmentation research field [51]. We also must mention that we were not able to find any published segmentation results specific to the AutoLaparo dataset, hence we have no actual metrics to compare our results achieved with. Although the mIoU of 69% seems respectable, there remains ample room for improvement.

4.2 Challenges

4.2.1 Custom Function Implementations

One of the challenges we encountered was the manual implementation of advanced functions such as Focal Loss, weighted categorical cross entropy and Dice coefficient. These metrics were not available in the version of the Keras that we used for this project, requiring extra efforts to implement them manually. This process was time-consuming and complex due to the fact that we were not able to find any published implementation of these functions and had to rely on our understanding of the mathematical formula.

4.2.2 Configuring TensorFlow with GPU

Another challenge that we encountered was configuring TensorFlow to utilize the graphics card on our device. We faced issues as the later versions of TensorFlow GPU were not officially supported on Windows 11 [52], adding complexity to the setup process. This issue required a considerable amount of time and effort, ultimately leading to the decision to downgrade TensorFlow. This decision came at the cost of losing access to many new features available in the later versions.

4.2.3 Computational Time and Resources

One significant obstacle faced throughout the project was the considerable time and computational resources required for experimenting and training the segmentation model. With well over 85 hours dedicated to training various models, the process proved extremely time intensive. The limited computational power available, reduced the ability to conduct a broader range of experiments and iterations, making it difficult to effectively optimize our hyper parameters. Access to additional computational resources could have improved the experimentation process significantly, enabling more iterations and potentially facilitating further improvements to the performance.

4.3 Limitations

The main limitation of this project is the discrepancy between the intended goal of improving surgical scene understanding for laparoscopic hysterectomy and the actual performance of the image segmentation models. While we made a lot of efforts to enhance the accuracy and performance of our segmentation model and provided valuable insights for laparoscopic hysterectomy surgeries, the achieved model may not be reliable or robust enough to solely rely upon, especially in such an important and sensitive domain where human lives are at stake. The importance and effect of accurate and dependable surgical scene understanding tools cannot be overstated, as it has significant potential to directly impact surgical decision making, safety of patients and outcome of surgeries. Hence, while the project represents a step forward in the direction of leveraging machine learning in the field of medical surgeries, further improvements and refinements of our segmentation model is necessary in order to ensure its effectiveness in real world surgical context.

4.4 Ideas for Future Work

4.4.1 Attention U-Net

An idea with great potential for future work involves exploring the use of attention in the U-Net architecture to improve the segmentation performance. While the base U-Net architecture is effective in various segmentation tasks, combining it with attention mechanisms could give us the chance to further improve the model's ability to capture relevant feature and focus on the more important regions within the input images. With attention allowing the model to pay more attention to specific parts of the input image that contain the most information, enhances the accuracy and quality of segmentation [53]. This approach could help the model better adapt to complex and varied structures present in surgical images, leading to better segmentation results.

4.4.2 Hyper Parameter Optimization

Another idea for future work involves using automated hyperparameter optimization tools such as Weights and Biases sweeps or similar frameworks to streamline this process. These tools use advanced algorithms to intelligently search the hyperparameter space, optimizing the model's performance. Adding such tools into the pipeline for development of segmentation models, could significantly improve and accelerate the optimization process, leading us to finding optimal parameters more efficiently and allocating more time toward further model experimentations.

4.4.3 Improving the Dataset

Expanding the dataset by combining images from multiple sources could significantly enhance the generalizability of our segmentation model. By doing so we could capture a wide range of anatomical variations and surgical instruments, therefore helping the model to better reflect the real-world surgical scenarios. Furthermore, adding extensive data augmentation to our pipeline can help better address the edge cases and improve the model's reliability towards variations in patient anatomy and lighting conditions.

4.4.4 Video Segmentation Using RNNs

Using Recurrent Neural Networks (RNNs) in the segmentation pipeline could offer a promising avenue for future research, specifically in the context of applying segmentation techniques to surgical videos. RNNs are suitable for modelling dependencies in sequential data, making them an excellent choice for capturing the dynamic nature of surgical tools and anatomies in video footage [50]. By leveraging RNNs, we can enable the model to maintain contextual understanding across frames and produce smoother and more consistent segmentation predictions. This method addresses the limitation of the temporal consistency and jittering visible in video segmentation and opens up opportunities for reliable real-time segmentation applications in surgeries.

4.4.5 Integration with Phase Detection Models

Integrating the segmentation model with a phase detection model can be a promising idea for improving the overall scene understanding in laparoscopic surgeries. Leveraging the phase information in the AutoLaparo task 2 dataset [31], which includes data on various phases and steps of surgery, the combined solution may be able to offer a comprehensive understanding of the surgical progression. By combining the results, we can facilitate real-time monitoring of surgical workflows which could offer significant assistance to surgeons.

References

- [1] Mascagni, P., Alapatt, D., Sestini, L. et al. Computer vision in surgery: from potential to clinical value. *npj Digit. Med.* 5, 163 (2022). <https://doi.org/10.1038/s41746-022-00707-5>
- [2] Einarsson, J. I., & Suzuki, Y. (2009). Total laparoscopic hysterectomy: 10 steps toward a successful procedure. *Reviews in obstetrics & gynecology*, 2(1), 57–64.
- [3] Mukhopadhaya, N., & Manyonda, I. T. (2013). The hysterectomy story in the United Kingdom. *Journal of mid-life health*, 4(1), 40–41. <https://doi.org/10.4103/0976-7800.109635>
- [4] Abdulateef, Salwa & Salman, Mohanad. (2021). A Comprehensive Review of Image Segmentation Techniques. *Iraqi Journal for Electrical and Electronic Engineering*. 17. 166-175. 10.37917/ijeee.17.2.18.
- [5] Hmrishav Bandyopadhyay. (2021). A Gentle Introduction to Image Segmentation for Machine Learning. [online] Available at: <https://www.v7labs.com/blog/image-segmentation-guide>.
- [6] Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N. and Terzopoulos, D. (2020). Image Segmentation Using Deep Learning: A Survey. arXiv:2001.05566 [cs]. [online] Available at: <https://arxiv.org/abs/2001.05566>.
- [7] Oct 13, M.W. and Read M. (2022). Difference Between Semantic and Instance Segmentation. [online] Roboflow Blog. Available at: <https://blog.roboflow.com/difference-semantic-segmentation-instance-segmentation/>.
- [8] Hassan, Z. (2023). Semantic Segmentation vs. Instance Segmentation: Explained. [online] Folio3AI Blog. Available at: <https://www.folio3.ai/blog/semantic-segmentation-vs-instance-segmentation/> [Accessed 30 Apr. 2024].
- [9] O’Shea, K. and Nash, R. (2015). An Introduction to Convolutional Neural Networks. arXiv:1511.08458 [cs]. [online] Available at: <https://arxiv.org/abs/1511.08458>.
- [10] www.guru99.com. (n.d.). TensorFlow Image Classification: CNN(Convolutional Neural Network). [online] Available at: <https://www.guru99.com/convnet-tensorflow-image-classification.html>.

- [11] Saha, S. (2018). A Comprehensive Guide to Convolutional Neural Networks—the ELI5 way. [online] Towards Data Science. Available at: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>.
- [12] Nik (2023). ReLU Activation Function for Deep Learning: A Complete Guide to the Rectified Linear Unit • datagy. [online] datagy. Available at: <https://datagy.io/relu-activation-function/>.
- [13] Koech, K.E. (2020). Softmax Activation Function — How It Actually Works. [online] Medium. Available at: <https://towardsdatascience.com/softmax-activation-function-how-it-actually-works-d292d335bd78>.
- [14] Badrinarayanan, V., Kendall, A. and Cipolla, R. (2017). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(12), pp.2481–2495.
- [15] Stathis Kamperis (2021). The encoder-decoder model as a dimensionality reduction technique. [online] Let's talk about science! Available at: <https://ekamperi.github.io/machine%20learning/2021/01/21/encoder-decoder-model.html>.
- [16] Ronneberger, O., Fischer, P. and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. [online] arXiv.org. Available at: <https://arxiv.org/abs/1505.04597>.
- [17] Ryan, M. (2019). How Neural Networks 'Learn'. [online] Medium. Available at: <https://towardsdatascience.com/how-neural-network-learn-3b56c175b5ca>.
- [18] Mao, A., Mohri, M. and Zhong, Y. (2023). Cross-Entropy Loss Functions: Theoretical Analysis and Applications. [online] arXiv.org. doi:<https://doi.org/10.48550/arXiv.2304.07288>.
- [19] 365 Data Science. (2021). What Is Cross-Entropy Loss? [online] Available at: <https://365datascience.com/tutorials/machine-learning-tutorials/cross-entropy-loss/>.
- [20] Lin, T.-Y., Goyal, P., Girshick, R., He, K. and Dollár, P. (2018). Focal Loss for Dense Object Detection. arXiv:1708.02002 [cs]. [online] Available at: <https://arxiv.org/abs/1708.02002>.
- [21] Gupta, L. (2021). Focal Loss — What, Why, and How? [online] The Startup. Available at: <https://medium.com/swlh/focal-loss-what-why-and-how-df6735f26616>.
- [22] Kingma, D.P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. [online] arXiv.org. Available at: <https://arxiv.org/abs/1412.6980>.

[23] Brownlee, J. (2016). Dropout Regularization in Deep Learning Models With Keras. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/>.

[24] Santos, C.F.G. dos and Papa, J.P. (2022). Avoiding Overfitting: A Survey on Regularization Methods for Convolutional Neural Networks. ACM Computing Surveys. doi:<https://doi.org/10.1145/3510413>.

[25] Mumuni, A. and Mumuni, F. (2022). Data augmentation: A comprehensive survey of modern approaches. Array, 16, p.100258. doi:<https://doi.org/10.1016/j.array.2022.100258>.

[26] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K. and Yuille, A.L. (2017). DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. arXiv:1606.00915 [cs]. [online] Available at: <https://arxiv.org/abs/1606.00915>.

[27] Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J.M. and Luo, P. (2021). SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers. arXiv:2105.15203 [cs]. [online] Available at: <https://arxiv.org/abs/2105.15203>.

[28] <https://keras.io/>

[29] <https://www.tensorflow.org/>

[30] <https://wandb.ai/site>

[31] Wang, Z., Lu, B., Long, Y., Zhong, F., Cheung, T.-H., Dou, Q. and Liu, Y. (2022). AutoLaparo: A New Dataset of Integrated Multi-tasks for Image-guided Surgical Automation in Laparoscopic Hysterectomy. [online] arXiv.org. doi:<https://doi.org/10.48550/arXiv.2208.02049>.

[32] Baruah, I.D. (2020). One Hot Encoding, Standardization, PCA : Data preparation steps for segmentation in python. [online] Medium. Available at: <https://towardsdatascience.com/one-hot-encoding-standardization-pca-data-preparation-steps-for-segmentation-in-python-24d07671cf0b>.

[33] Shah, T. (2017). About Train, Validation and Test Sets in Machine Learning. [online] Towards Data Science. Available at: <https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7>.

[34] GeeksforGeeks. (2017). `numpy.argmax()` in Python. [online] Available at: <https://www.geeksforgeeks.org/numpy-argmax-python/>.

[35] <https://matplotlib.org/>

[36] www.nvidia.com. (n.d.). GeForce GTX 1050 Ti | Specifications | GeForce. [online] Available at: <https://www.nvidia.com/en-gb/geforce/graphics-cards/geforce-gtx-1050-ti/specifications/>.

[37] <https://github.com/>

[39] Jensen, M.D., Ryan, D.H., Apovian, C.M., Ard, J.D., Comuzzie, A.G., Donato, K.A., Hu, F.B., Hubbard, V.S., Jakicic, J.M., Kushner, R.F., Loria, C.M., Millen, B.E., Nonas, C.A., Pi-Sunyer, F.X., Stevens, J., Stevens, V.J., Wadden, T.A., Wolfe, B.M. and Yanovski, S.Z. (2013). 2013 AHA/ACC/TOS Guideline for the Management of Overweight and Obesity in Adults. *Circulation*, [online] 129(25 suppl 2), pp.S102–S138. doi:<https://doi.org/10.1161/01.cir.0000437739.71477.ee>.

[40] Jason Brownlee (2019). Understand the Impact of Learning Rate on Neural Network Performance. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/>.

[41] Goswami, D.S. (2020). Introduction to Early Stopping: an effective tool to regularize neural nets. [online] Medium. Available at: <https://towardsdatascience.com/early-stopping-a-cool-strategy-to-regularize-neural-networks-bfdeca6d722e>.

[42] Jordan, J. (2018). Evaluating image segmentation models. [online] Jeremy Jordan. Available at: <https://www.jeremyjordan.me/evaluating-image-segmentation-models/>.

[43] Rosebrock, A. (2016). Intersection over Union (IoU) for object detection. [online] PyImageSearch. Available at: <https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>.

[44] [kaggle.com](https://www.kaggle.com). (n.d.). Understanding DICE COEFFICIENT. [online] Available at: <https://www.kaggle.com/code/yerramvarun/understanding-dice-coefficient>.

[45] <https://numpy.org/>

[46] <https://opencv.org/>

[47] Kolbinger, F.R., Rinner, F.M., Jenke, A., Carstens, M., Krell, S., Leger, S., Distler, M., Weitz, J., Speidel, S. and Bodenstedt, S. (2023). Anatomy segmentation in laparoscopic surgery: comparison of machine learning and human expertise – an experimental study. *International Journal of Surgery*, [online] 109(10), pp.2962–2974. doi:<https://doi.org/10.1097/js9.0000000000000595>.

- [48] Wang, Lu & Wang, Chaoli & Sun, Zhanquan & Chen, Sheng. (2020). An Improved Dice Loss for Pneumothorax Segmentation by Mining the Information of Negative Areas. IEEE Access. PP. 1-1. 10.1109/ACCESS.2020.3020475.
- [49] Nayak, R. (2022). Focal Loss : A better alternative for Cross-Entropy. [online] Medium. Available at: <https://towardsdatascience.com/focal-loss-a-better-alternative-for-cross-entropy-1d073d92d075>.
- [50] Schmidt, R.M. (2019). Recurrent Neural Networks (RNNs): A gentle Introduction and Overview. arXiv:1912.05911 [cs, stat]. [online] Available at: <https://arxiv.org/abs/1912.05911>.
- [51] Lei Yang, Yuge Gu, Guibin Bian, Yanhong Liu, An attention-guided network for surgical instrument segmentation from endoscopic images, Computers in Biology and Medicine, Volume 151, Part A, 2022, 106216, ISSN 0010-4825, <https://doi.org/10.1016/j.combiomed.2022.106216>.
- [52] TensorFlow. (n.d.). Install TensorFlow with pip. [online] Available at: <https://www.tensorflow.org/install/pip#windows-native>.
- [53] Oktay, O., Schlemper, J., Folgoc, Loic Le, Lee, M., Heinrich, M., Misawa, K., Mori, K., McDonagh, S., Hammerla, Nils Y, Kainz, B., Glocker, B. and Rueckert, D. (2018). Attention U-Net: Learning Where to Look for the Pancreas. [online] arXiv.org. Available at: <https://arxiv.org/abs/1804.03999>.

Appendix A – Self Appraisal

A.1 Critical Self-Evaluation

For this laparoscopic hysterectomy scene understanding project, it's clear that while the evaluation results achieved may not have met the desired level of excellence, they undeniably demonstrate significant potential. The aim of developing a segmentation model for segmenting surgical tools and anatomical structures was successfully realized. Furthermore, efforts were made towards optimizing the model and creating a prototype which is capable of displaying masks overlaying surgical images, a goal that was effectively reached. Despite the potential for improvements in the achieved results, this project lays a solid foundation for future improvements in this field, showing promising avenues for enhancing segmentation models in laparoscopic procedures.

In reflecting on this project, it is important to acknowledge that even though certain milestones were reached, some initial aims remained unfulfilled. Notably, the aim of significantly optimizing the model's performance was limited due to the constraints stemming from limited time and computation resources. The lack of experimentation opportunities limited the through exploration of different optimization methods, resulting in lower model efficacy. Moreover, the aim of comparing the achieved prototype with state-of-the-art models was not achieved due to the absence of published results for the AutoLaparo dataset. Attempts for solving this issue were made by exploring alternative dataset with published results however, these endeavours were met with challenges, including unavailability and lack of response from dataset authors. Despite these issues, the project's progression highlights the importance of resource allocation and strategic planning for unforeseen challenges.

Reflecting on the project's trajectory, it is clear that a more proactive approach for evaluating our aims and potential limitation and risks could have given a more favourable outcome. For example, anticipating the challenges related to time and computational resources, an earlier consideration for renting a cloud GPU for training might have solved these issues, enabling more opportunities for extensive experimentation and optimization efforts. Furthermore, instead of solely focusing on incremental improvements through experimentation with loss functions, a shift towards exploring and experimentation towards exploring different variations of U-Net models might have led to a more substantial improvement in performance. This insight underscores the importance of highlighting the value of proactive risk assessment in guiding research endeavours towards a more favourable outcome.

A.2 Personal Reflection and Lessons Learned

Throughout my academic journey, I have always had a longstanding fascination with machine learning particularly in the field of computer vision. Working on this project provided a great opportunity for me to get a better understanding of the intricacies of state-of-the-art deep learning models used for computer vision projects. The hands-on experience gained, not only helped with my technical skills but also fostered a deeper appreciation for the complex mechanisms used in designing and implementing neural network architectures.

For this project, I developed a keen aptitude for background research and crafting well-written technical reports. I refined my efficiency in sourcing relevant research papers and leveraging online resources to address challenges in an effective manner. This experience not only improved my skills in synthesizing information from diverse papers but also gave me a sense of resourcefulness and agility in navigating the ever-expanding landscape of computer vision research. In future, these skills will undoubtedly help me in my pursuit of academic and professional endeavours.

Finally, as the project unfolded, I gained an appreciation for the importance of allocating time and resources to various tasks and features. I learned to navigate problems by breaking them down into manageable tasks and allocating some extra time for unforeseen emergencies and setbacks. By embracing proper time management principles, I was able to maintain a good momentum and productivity for this project, resulting in its successful completion. Moving forward, lessons learned from this project will undoubtedly serve as a framework for navigating future challenges with confidence.

A.3 Legal, Social, Ethical and Professional Issues

A.3.1 Legal Issues

The AutoLaparo dataset, one of the main components of this project, was acquired through proper channels, ensuring adherence to guidelines and copyright regulations. References to the dataset were meticulously included, acknowledging the contributions of the original authors. Moreover, the tools employed through this project, including TensorFlow and Keras were chosen for their open-source nature, maintained by Google and are distributed under the Apache License 2.0. This licence allows users the freedom to use the software for any purpose, including commercial applications and research. It is important to comply with the terms of the license which includes providing a copy of license and clearly indicating changes applied to the software.

A.3.2 Social Issues

One potential social issue with this project is in case of applying this technology for minimally invasive surgeries, it might give a false sense of security to surgeons using automated segmentation

models. While these technologies aim to assist surgeons, there is a concern that over-reliance on such tools may lead to reduced vigilance among surgeons. If they become overly reliant on such automated tools, they may overlook critical details or fail to exercise necessary caution. Additionally, some of the patients may feel not comfortable to undergo minimally invasive surgery if they are aware that the procedures will be recorded by computer vision systems. Concerns about privacy may deter patients from the willingness to undergo surgeries that involve the use of automated segmentation models.

A.3.3 Ethical Issues

Privacy emerges as a paramount concern when addressing the ethical considerations for this project. This issue revolves around the patients whose data is utilized, particularly in the context of the AutoLaparo dataset employed in this project. This dataset is consisted of videos of patients undergoing surgical procedures and it important to ensure that all the individuals have provided consent for the use of their footage by researchers. Moreover, more ethical issues may raise if the prototype developed in this project is used in any actual surgical settings. While the automated segmentation models aim to improve surgical outcomes, patients' privacy must be prioritized. Many patients may object to the recording of their surgical footage citing concerns about privacy invasion and data security. Therefore, it is important that healthcare professional and researchers engage in transparent communication with patients ensuing them of the implication of using such technologies.

A.3.4 Professional Issues

The segmentation model for laparoscopic hysterectomy has been developed in compliance with the British Computer Society code of conduct. Weights and Biases was employed for iterative analysis of model changes. Git and GitHub were utilized for version control enabling tracking of project modifications. Valuable feedback from my supervisor and assessor was carefully used to ensure the delivery of a high-quality segmentation model.

Appendix B – External Materials

The AutoLaparo dataset used for this application which contained 1800 images and masks of laparoscopic hysterectomy surgery.

<https://autolaparo.github.io/>

@InProceedings{wang2022autolaparo, title = {AutoLaparo: A New Dataset of Integrated Multi-tasks for Image-guided Surgical Automation in Laparoscopic Hysterectomy}, author = {Wang, Ziyi and Lu, Bo and Long, Yonghao and Zhong, Fangxun and Cheung, Tak-Hong and Dou, Qi and Liu, Yunhui}, booktitle = {International Conference on Medical Image Computing and Computer-Assisted Intervention}, pages = {486--496}, year = {2022}, organization = {Springer} }

Machine Learning tools for creating the segmentation model: Keras and TensorFlow

<https://keras.io/>

<https://www.tensorflow.org/>

Drawing tool used for creating diagrams

<https://www.drawio.com/>

Analytic tool for evaluation and live tracking different experiments

<https://wandb.ai/site>

Libraries used for data visualization

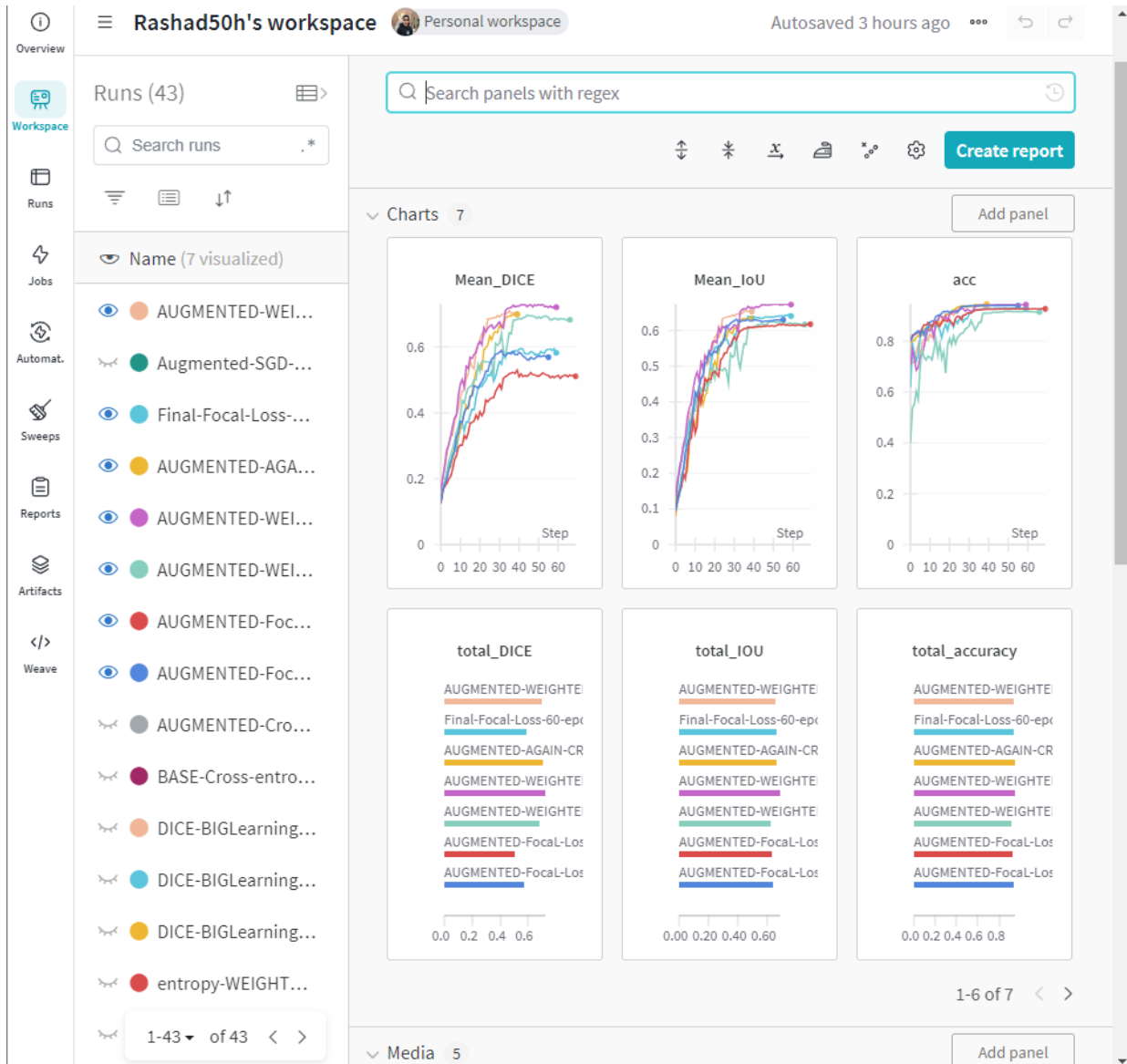
<https://opencv.org/>

<https://matplotlib.org/>

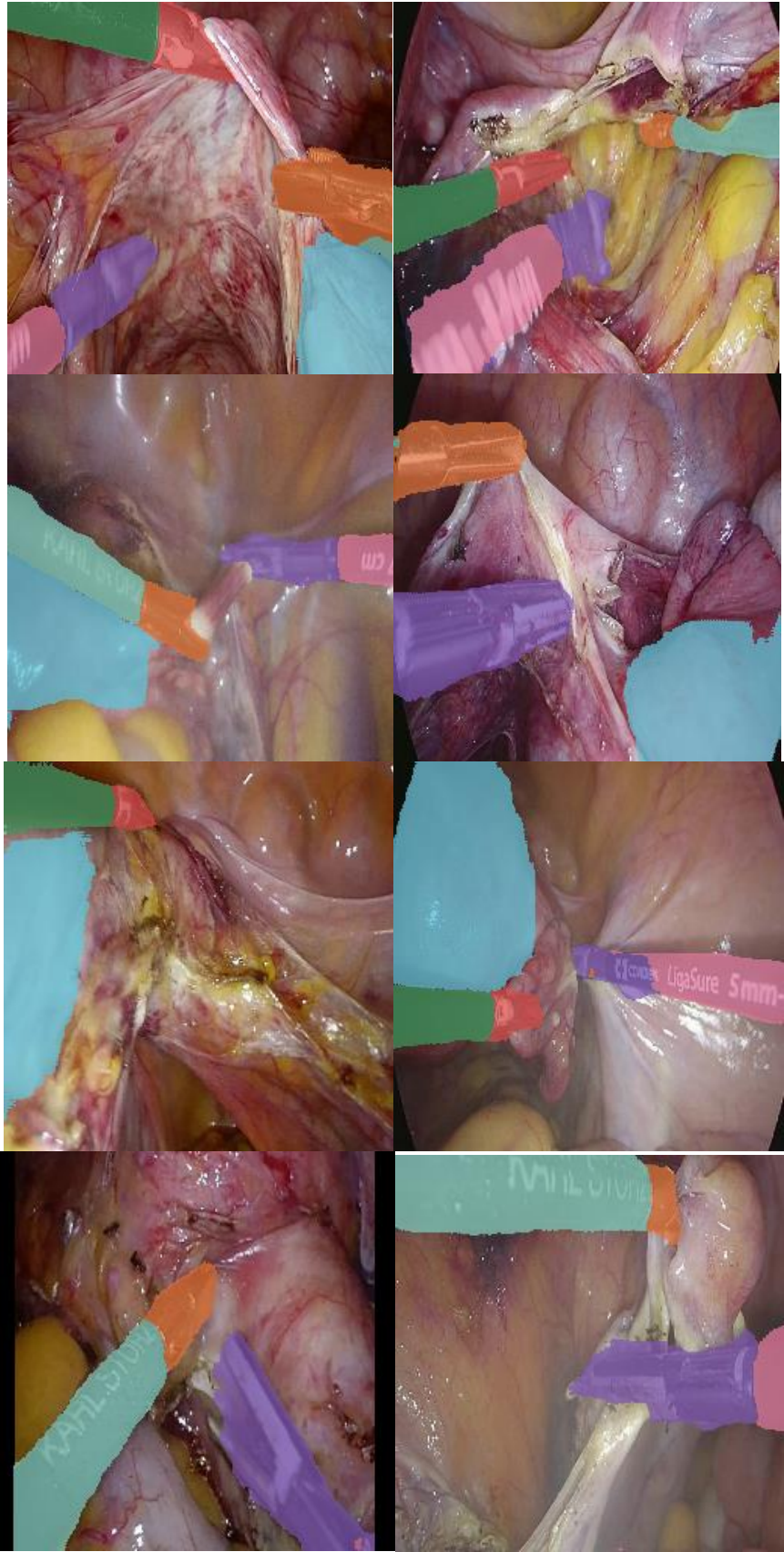
<https://numpy.org/>

Appendix C – Additional Material

Custom Dashboard on Weights and Biases:



More Predictions from the Prototype:



Custom DICE Function:

```
def dice_coef(y_true, y_pred, smooth=100):
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)
    intersection = K.sum(y_true_f * y_pred_f)
    dice = (2. * intersection + smooth) / (K.sum(y_true_f) + K.sum(y_pred_f) +
    smooth)
    return dice

def dice_coef_multilabel(y_true, y_pred, i=None, M=10, smooth=1e-5):

    if i:
        result = dice_coef(y_true[:, :, :, i], y_pred[:, :, :, i], smooth)
        return result
    else:
        dice = 0
        for index in range(M):
            result = dice_coef(y_true[:, :, :, index], y_pred[:, :, :, index], smooth)
            dice += result
        return dice / M
```

Custom Focal Loss Function:

```
def categorical_focal_loss(alpha, gamma=2.):

    alpha = np.array(alpha, dtype=np.float32)

    def categorical_focal_loss_(y_true, y_pred):

        epsilon = K.epsilon()
        y_pred = K.clip(y_pred, epsilon, 1. - epsilon)

        cross_entropy = -y_true * K.log(y_pred)

        loss = alpha * K.pow(1 - y_pred, gamma) * cross_entropy

        return K.mean(K.sum(loss, axis=-1))

    return categorical_focal_loss_
```

Custom Weighted Categorical Cross Entropy Function:

```
def weighted_categorical_crossentropy(weights):  
  
    weights = K.variable(weights)  
  
    def loss(y_true, y_pred):  
  
        y_pred /= K.sum(y_pred, axis=-1, keepdims=True)  
        y_pred = K.clip(y_pred, K.epsilon(), 1 - K.epsilon())  
  
        loss = y_true * K.log(y_pred) * weights  
        loss = -K.sum(loss, -1)  
        return loss  
  
    return loss
```

Appendix D – Links to Deliverables

Demo Video:

- <https://www.youtube.com/watch?v=PnGtaq9bOVY>

Code Repository:

- <https://github.com/rashad-h/Surgical-Scene-Understanding-Laparoscopic>