Real Time Analysis of Cataract Surgery Videos Using Deep Learning

A thesis

submitted in fulfilment

of the requirements for the Degree

of

Master of Science (Research) in Computer Science

at

The University of Waikato

by

Jesse Whitten

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

2022

Abstract

A deep learning framework is proposed that classifies surgical instruments within cataract surgery videos and generates clinical summary reports, proving objective reporting and analysis. This framework is run in real time on an embedded system, potentially allowing real time feedback directly in a clinical environment. To evaluate this framework, a novel dataset of 39 cataract surgery videos is sampled at different frame rates and labeled with 10 surgical instrument classes. This dataset is used to train and evaluate several image classification deep learning model architectures: ResNet-50, ResNet-152 and InceptionV3. An Nvidia Jetson Nano 4GB embedded system is used to operate an optimised variation of the framework, evaluating the accuracy and representation of clinical summary reports. It was found that the deep learning models provided a maximum mean predictive AUC score of 0.982 and a sensitivity of 0.968 on hold out test videos. The Jetson Nano was demonstrated to provide accurate and representative clinical summary reports with a video frame rate of 6 frames per second, completing in real time. These results show that the deep learning framework is viable for integration into the clinical environment, automatically providing clinical summary reports containing objective assessment statistics, video annotations and graphical timeline visualisations of cataract removal surgeries.

Acknowledgements

I would like to thank and acknowledge Dr Michael Mayo for his invaluable help and mentorship, I very much appreciate all the time and effort he has put into guiding me and providing valuable feedback throughout this long work. I would also like to thank Dr James McKelvie for his contributions, direction and support. He has taught me enough about cataract surgery to be nearly confident enough to perform operations myself, and has inspired a great appreciation for the medical sector. Finally, I thank my family and friends for their continued excitement about my progress and milestones, a great encouragement.

Contents

Chapter 1

Introduction

An aging global population poses a significant challenge, not just due to the expected slowing of economic growth [1], but as a number of health risks increase with age, the global quality of life may decrease. A cataract, an opacification of the natural crystalline lens within the eye, has an increasing risk of developing for every decade of age [2]. The development of a cataract is associated with visual impairment and can dramatically reduce quality of life for those affected. Cataracts are currently the leading cause of blindness around the globe [3], with cataract surgery being the only option for restoring vision. Continuing to improve cataract surgery health outcomes and decrease complication rates is paramount for a high quality of life within an aging society.

Machine learning methods, primarily deep learning techniques, are now being implemented into the medical sector to improve health outcomes and reduce complication rates [4], ushering in a new era of medical technologies. Although cataract surgery is the most common surgical procedure worldwide [5], there is limited research conducted on combining deep learning and cataract surgery. This is likely due to a lack of publicly available datasets, which, requiring patient consent, are difficult to publish. This restricts the ability to evaluate deep learning models.

The primary objective of previous research in this area has been split between two main focuses: surgical phase and step classification, and surgical instrument classifica-

tion. A modern cataract surgery typically consists of ten to twelve surgical steps, which are divided across four main phases. Cataract surgeries will also involve between ten and twenty individual surgical instruments, each used for specific tasks throughout the duration of the surgery. A number of deep learning models and techniques have been proposed and evaluated for classification of both these categories, with substantial predictive results. However, very little research has been conducted on using these models within a clinical environment, or integrating them into the surgery workflow. Cataract surgery has yet to benefit from the revolutionary field of deep learning.

The research conducted within this thesis involves classifying surgical instruments within cataract surgery videos, providing a detailed representation of the surgery workflow. Clinical summary reports are then generated from the classified videos, providing visual representations of the surgery timeline and objective statistics showing the exact usages of instruments throughout the surgery. These reports are useful for automatically documenting surgeries, providing objective assessments for surgeons in training, quality improvement and a number of administrative purposes. After developing and training a deep learning prototype on a server, it is deployed to an embedded device, allowing the classification of videos directly in a clinical environment, in real time.

There are three primary research questions proposed within this thesis: can deep learning provide high predictive performance for detecting various instruments when they appear in individual frames of cataract eye surgeries? Can these predictions be combined over the length of the video to produce relevant clinical summary reports that represent cataract surgery videos? Can that framework be accurately operated on embedded systems in real time? To evaluate these research questions, a deep learning framework is proposed and evaluated. Figure 1.1 shows a simplified flowchart of the proposed framework at prediction time, which takes a single cataract surgery video and generates a corresponding clinical summary report. First, in step 1, frames are extracted from a cataract surgery video at a particular frame rate. Then, in step 2, frames are preprocessed and in step 3 instrument classifications are made by deep

learning models. Step 4 involves creating Instrument Activity Sequences, sequences of positive predictions, which are used in step 5 to create a clinical summary report for the original video. To evaluate this framework, a novel dataset is created and a range of experiments are conducted, evaluating classification accuracy, the representation and relevance of the generated reports and the operability of the framework on embedded systems.

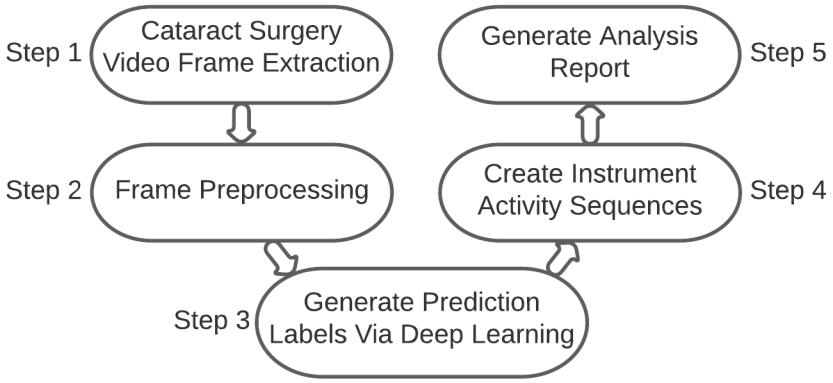

Figure 1.1: Flowchart of the deep learning framework.

The layout of this thesis is as follows. Chapter 2 examines the cataract surgery techniques and the relevant background literature of cataract surgery classification. Chapter 3 discusses the novel cataract surgery datasets developed for the evaluation of the research in this thesis. Chapter 4 proposes the deep learning framework that analyses cataract surgery videos and generates clinical summary reports. It also discusses the selection method, training and evaluation of deep learning models. Chapter 5 discusses the clinical summary reports, analysis of videos and the clinical relevance of the deep learning framework. Chapter 6 explores the use of embedded systems for operating the deep learning framework in a clinical setting, in real time. Finally, Chapter 7 discusses the findings and results of the research conducted throughout this thesis.

Chapter 2

Background

2.1 History of the Cataract Surgery

The opacification of the natural crystalline lens often occurs with age, leading to loss of vision and decreased quality of life for those affected. This clouding of the crystalline lens is referred to as a cataract, and currently is the cause of 51% of blindness worldwide [6]. While age is a primary risk factor for developing a cataractous lens, other factors, including certain diseases such as diabetes, lifestyle choices such as smoking and alcohol use, and environmental causes such as prolonged exposure to sunlight or occupation each contribute to increasing risk [7], [8]. There are approximately ten million cataract surgeries performed every year [5], and a large volume of research has contributed to the development of many surgical techniques to clear, remove or replace the cataractous lens.

Figure 2.1 shows a diagram of the anatomy of a human eye. As labeled in the diagram, the crystalline lens sits freely within the capsular bag (also often referred to as the lens capsule), which is suspended in place by zonuler fibres. The pupil offers a convenient access point to the capsular bag and lens. In general terms, a cataract or cataractous lens refers to the natural lens which has become clouded or opaque, so a cataract could also be equivalently depicted in Figure 2.1 as the lens. Modern cataract removal techniques involve removing the cataractous lens from within the capsular bag.

This is done by operating instruments through incisions in the cornea and the dilated pupil to access the capsular bag and the housed crystalline lens. After removal, an artificial intraocular lens is then implanted within the capsular bag, restoring vision.

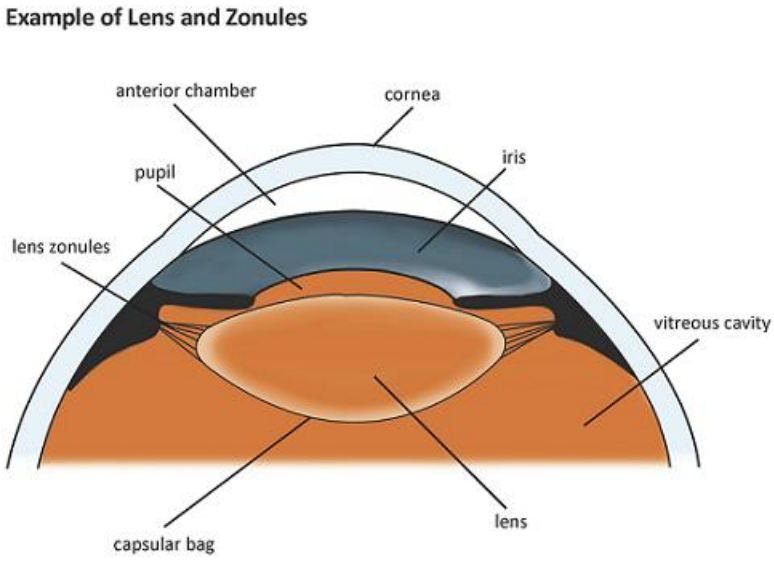

Figure 2.1: Eye Anatomy [9].

Throughout the last five or six centuries the technique of cataract removal has evolved with numerous refinements to improve saftey and outcomes. Two main strategies rotated in and out of favor as surgical methods developed. These two methods are extracapsular cataract extraction (ECCE) and intracapsular cataract extraction (ICCE), where the former makes an incision into the capsular bag and removes the cataractous lens, leaving the capsular bag in place, while the latter method removes the capsular bag and cataractous lens from the eye, breaking the lens zonules [10]. Couching, one of the earliest documented cataract removal techniques, differs from both ICCE and ECCE techniques, as the cataractous lens remains in the eye. Zonules are disrupted to allow dislocation of the crystalline lens. The lens is then manipulated using a needle to push and rotate the cataractous sections of the lens out of the axis of vision of the patient. While considered a success by early surgeons, couching often led to blindness shortly after surgery due to the lack of aseptic technique and the remain-

ing cataractous lens continuing to develop and worsen [10]. For reference, Figure 2.2 shows a timeline of the development of cataract surgery methods over the last several centuries.

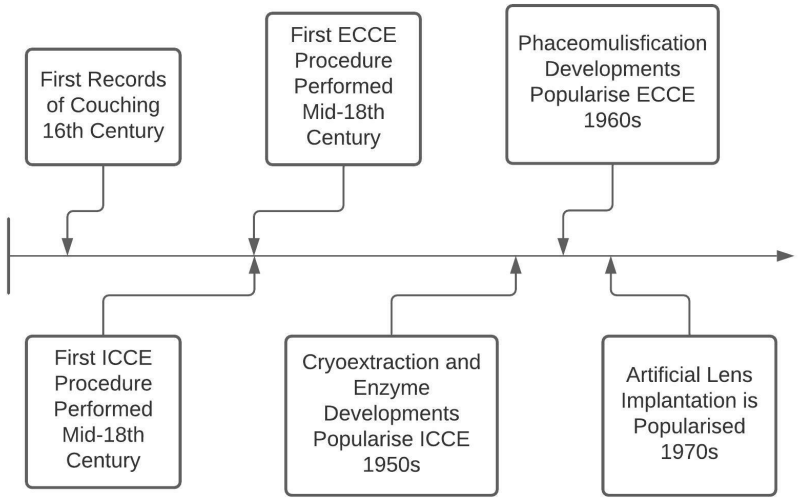

Figure 2.2: Cataract Surgery History Timeline.

In the mid 18th century the first ECCE and ICCE procedures were performed respectively by Jacques Daviel and Samuel Sharp. Daviel's ECCE technique involved cutting into the lens capsule, exposing the lens and then extracting the cataractous lens by curettage. This method had a reported 50% success rate, a considerably better alternative to couching. It did however require a 10mm incision which led to slow healing rates, and the technique often left cataract remnants inside the lens capsule, leaving patients prone to infection [10]. Sharp's ICCE method removed the entire lens capsule from the eye; he used his thumb to forcibly expel the capsule and to break the zonular fibres that held it in place. Without the natural lens patients would often require strong spectacles. This method of extraction was not received well, and did not gain popularity until the mid 20th century when technology advances allowed enzymes or cryoextraction to gracefully break the zonular fibres holding the capsule in place. These ICCE techniques quickly fell out of use as ECCE methods advanced, as large incisions require sutures that provide a high risk of surgically induced astigmatism [10],

where the front of the eye gains an irregular shape and can cause blurry vision.

Modern techniques involve phacoemulsification, a process that uses ultrasound waves to liquefy and remove cataractous lenses. The introduction of intraocular lenses allowed the insertion of artificial lenses into the lens capsule to replace cataractous lenses. Early artificial lenses were rigid and required large insertions to implant. Both ICCE and ECCE methods began to use artificial lenses, but the ECCE method, which implanted the lens into the capsular bag, rather than the anterior chamber (as in the ICCE technique), was associated with a lower complication rate. Modern ECCE procedures use both phacoemulsification and modern flexible lenses in conjunction [10]. Alongside incorporating several other advances in technique and technology, a 2012 research paper shows that modern ECCE procedures have a success rate of 90%-95% [11], and other studies show they typically take between 20 to 30 minutes to complete [12].

2.2 Phacoemulsification

Phacoemulsification has been the most common type of cataract removal surgery since the 1960's [13], and is the primary surgical technique examined within this thesis. This ECCE procedure typically consists of four key phases and involves eight to twelve steps, each requiring different instruments and techniques. The four overarching phases of this procedure are the incision phase, the capsulorhexis phase, the nuclear disassembly phase and the IOL implantation phase. The individual steps involved are described in [14]: "(1) anesthesia, (2) prepping, draping, and the microscope's setup, (3) the corneal incisions, (4) capsulorhexis, (5) hydrodissection and nuclear disassembly, (6) cortical cleanup, (7) implantation of the IOL, and (8) closure of the incision." This routine linear progression of steps make up the conventional surgery procedure, but variations in pupil size, patient characteristics, and surgeon preference may result in the use of additional steps like using certain medications, other devices or a variety of surgical techniques. Other surgical steps are optionally included, such as the use of a malyugan ring or iris hooks to dilate the pupil or the insertion of dye to assist in visualising the capsule in

particular cataract subtypes. These steps are depicted in Figure 2.3. A large number of instruments are used throughout these phases, including forceps, keratome blades, paracentesis blades, rycroft cannulas, choppers, phacoemulsification probes, irrigation and aspiration probes, rhexis forceps, lens cartridge injectors, malyugan ring injectors and toric axis markers. The majority of these instruments can be seen in Figure 2.4. Additionally, a binocular stereo microscope is used to aid surgeons. The microscope provides an optimal view of the eye, records a video stream and can accommodate the operating surgeon and an assistant. This all results in an intricate, but standardised, surgical technique, that can be varied for individual patients and surgeons.

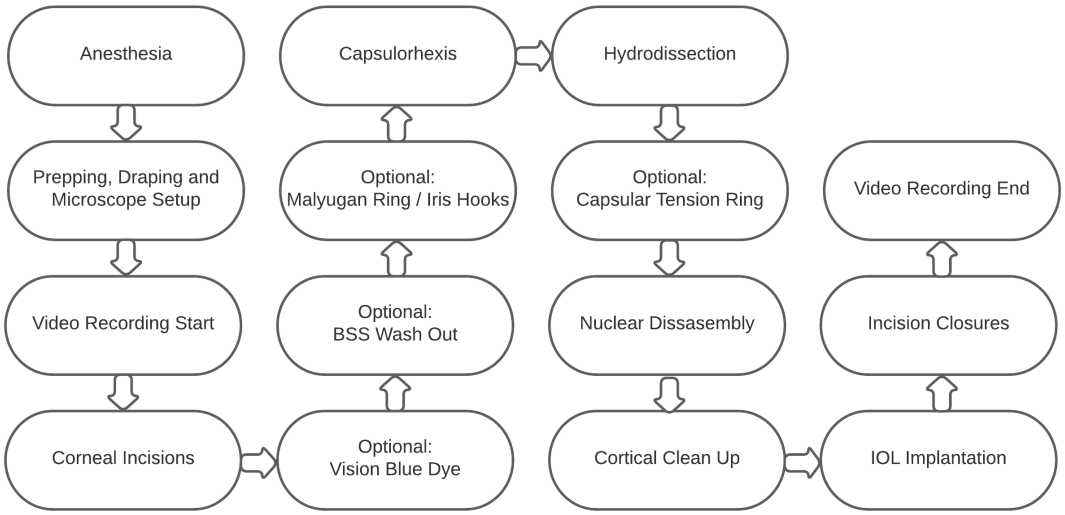

Figure 2.3: Phacoemulsification Cataract Surgery Process.

Anesthesia is the initial step of the procedure, and is commonly applied in the form of an injection behind the eye or an eye drop of topical anesthesia. Alternative forms of anesthesia can be used. In certain conditions the surgeon may elect to use intravenous sedation or general anaesthetic if indicated. It's mentioned in [14] that even the simple use of tape to strap the patients forehead to the headrest can often permit safe surgery.

Prepping, draping and the microscope's setup is an important pre-surgical step, where efforts to minimise bacterial contamination from the surrounding skin, eyelashes and meibomian glands are made. This includes using a half strength povidone iodine

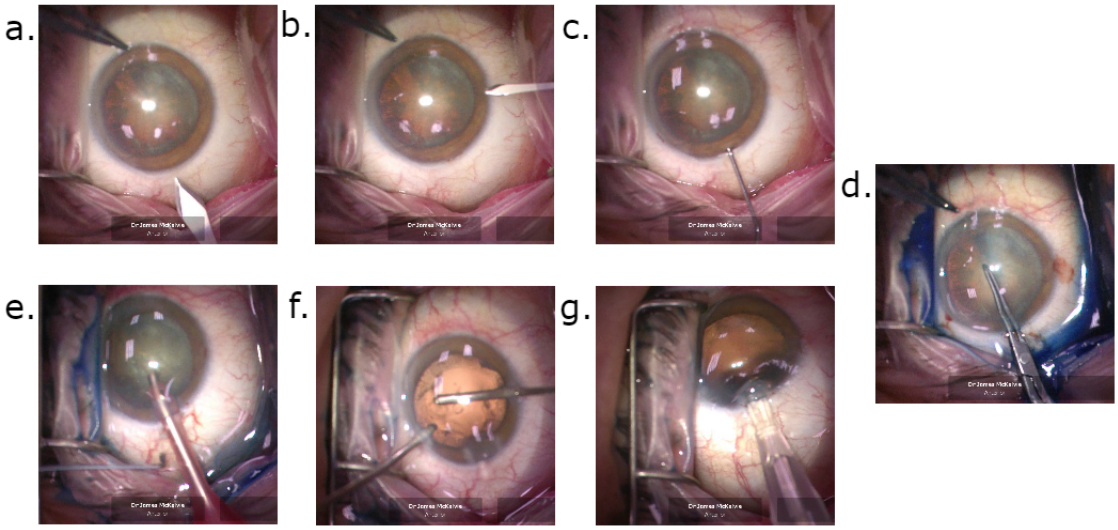

Figure 2.4: Common Instrument Examples.

Key: a. (top) Forceps, (bottom) 2.4mm Keratome Blade; **b.** (top) Forceps, (right) 1.1mm Paracentesis Blade; **c.** Rycroft Cannula; **d.** Rhexis Forceps; **e.** (bottom left) Chopper, (center) Phacoemulsification Probe; **f.** Irrigation/Aspiration Probes; **g.** Lens Cartridge Injector.

solution over the nearby skin and the ocular surface, and draping to isolate the eyelashes and meibomian glands. This step also includes the insertion of a speculum to hold open the patients eyelids. The microscope light must be configured to produce a sufficient red reflex within the eye.

The incision phase must then be performed prior to the subsequent surgical steps, and involves the corneal incision step. Often the non-dominant hand will use a stabilising instrument, such as a forceps grip, shown in Figure 2.4(a)(b), which can be used to grasp the sclera for stabilisation or to hold near the incision point to prevent eye rotation. Other instruments are also occasionally used for stabilisation, such as a Thornton fixation ring, which requires constant downward pressure to maintain traction, or cotton tipped applicators [15]. There are three incisions often required to operate the instruments used throughout the procedure: two short paracentesis incisions and a sin-

gle large main incision made using a keratome blade. The size of these incisions (and hence blade sizes also) is dependent on the individual surgeon and other instrument sizes [15]. The forcep grips and two blades are shown in Figure 2.4(a)(b). The widths of these blades differ with individual surgeon preferences, as they need to be large enough to insert the intraocular lens and instruments through, while narrow enough to restrict fluid escape during operation. The paracentesis incisions are made with consideration to the main incision, where close placement may result in "oar locking", making the use of instruments difficult. Viscoelastic fluid is often injected into the eye to maintain intraocular pressure, which provides stability while creating the main incision. A cannula instrument is often used for this injection, shown in 2.4(c). The main incision is then made using a keratome blade, and is typically placed for maximum accessibility and freedom of movement for the use of instruments and the surgeon [15].

Between the incision phase and the capsulorhexis phase there are several optional steps that are determinate on several factors. The first of these steps is the vision blue dye step, which involves the injection of blue dye into the eye, to improve visualisation of the cataract for particular cataract subtypes. This can be seen most clearly in Figure 2.4(d), where the cataractous lens in the center of the eye has a blue tinge, as opposed to its normal colour seen in Figure 2.4(f). The next optional step is a wash out with Balanced Salt Solution (BSS). BSS is an isotonic salt solution, and consists of a salt concentration balanced to match the composition of intraocular fluid. It can be used to hydrate the incisions and wash out unwanted vision blue dye. Finally, if the pupil has not dilated to a satisfactory radius allowing the manipulation of instruments, a Malyugan ring or iris hooks can be used to artificially dilate the pupil. These devices are removed in the final incision closures step.

The capsulorhexis phase is where the anterior surface of the capsular bag is pierced, folded back and removed in a circular extraction, allowing the ECCE technique of removing the cataractous lens from within the capsular bag. The precision of the capsulorhexis is highly important for IOL stability and position after insertion [15]. This

surgical maneuver requires very precise movements and is remarked to be a commonly difficult part of the operation for inexperienced surgeons [14]. Significant complications can occur from a capsulorhecis that is to small, to large, or runs out to the equator of the lens. Ensuring appropriate filling of viscoelastic within the anterior chamber using the cannula instrument is critical to maintaining optimal tearing characteristics of the capsule [15]. Often a Cystotome needle is then used to initially puncture the anterior surface of the capsular bag, but often the Rhexis forceps, shown in 2.4(d), is sufficient for both initial puncture and extraction. Once the resulting flap is punctured and elevated, Rhexis forceps are then used to pull the capsular flap in a circular motion, shearing a circular section and extracting it from the anterior face of the capsular bag. Great care is taken; a perfectly circular and appropriately sized extraction allows for minimal complications, such as further tearing and radial extension during the phacoemulsification phase, or difficulty removing the cataract.

The nuclear disassembly phase consists of two primary steps, hydrodissection and phacoemulsification. As seen in Figure 2.1, the (cataractous) lens roughly resembles a three dimensional elliptical shape, and sits within the capsular bag. The hydrodissection step removes the adhesion keeping the lens stationary within the surrounding capsule, and allows it to rotate freely. More specifically, a cannula instrument is used to inject BSS or local anaesthetic into the capsular bag, around and behind the lens, breaking the adhesion between the capsule interior and the lens. A successful hydrodissection is indicated by a wave pattern of solution showing from the posterior of the lens. This enables the free rotation of the lens, allowing it to be removed from the capsule without unnecessary complications [14], [15]. After hydrodissection the optional step of implementing a Capsular Tension Ring is available for patients who may have weak zonular fibres. This C-shaped ring device reinforces the capsular bag and allows for safer manipulation of the cataractous lens in the following step [16].

Following hydrodissection, the “phaco” machine, which uses ultrasound waves to crack dense objects and is shown in Fig. 2.4(e), is used in the phacoemulsification step.

This is where the cataractous lens is emulsified to facilitate removal via aspiration. There are several techniques available for the use of the phaco machine to disassemble the crystalline lens, and these may differ in difficulty to learn, the time required to perform and complication rates. One of these techniques is the divide and conquer strategy [15]. The divide step of this strategy involves grooving and cracking the crystalline lens, breaking it into several parts. The phaco machine has a foot pedal which allows the operator fine control over the power of the instrument. Greater power is required nearer the center of the lens, where the crystalline lens is thicker. The initial groove must be wide enough to fit the instruments, while also deep enough to ensure cracking the nucleus of the lens. A visible red reflex often indicates a correct depth [14]. After this groove is created, a second instrument is inserted, often called a "chopper", and is shown in 2.4(e). The two instruments are placed centrally into the groove and the chopper is twirled in an "up and out" rotational movement, creating a crack within the lens. This is repeated, creating further fractures and dividing the lens into smaller mobile pieces [15]. The conquer stage consists of the removal of these fractured pieces, using the vacuum function of the phaco machine. Larger pieces can be cracked or broken again, and further viscoelastic can be reinjected using a cannula instrument to float pieces up from the pupillary plane. Great care must be taken throughout the entire step to ensure the phaco machine needle tip only intercepts pieces of the lens, as it can easily cut through other sections of the eye, causing complications [15].

Cortical clean up is a critical step, where the remaining lens cortex is removed via precise instruments: the Irrigation and Aspiration probes, shown in Fig 2.4(f). If this step is not conducted thoroughly, postoperative inflammation is risked, alongside difficulty achieving consistent concentration of the intraocular lens [14]. Additionally, inattentive operation can lead to rupturing the lens capsular bag, a significant complication [15].

The IOL Implantation phase involves an artificial Intraocular Lens (IOL), which is folded into a cartridge and injected through the main incision using the lens cartridge

injector instrument, shown in 2.4(g). The IOL consists of an optic lens and two haptic hooks, and can often be injected directly into the capsular bag that housed the cataractous lens [14]. The IOL can then be unfolded and may be manipulated further inside the lens capsule by using a mushroom manipulator, Sinskey hook or the Irrigation and Aspiration probes [15]. Before IOL injection, viscoelastic is introduced via a cannula instrument to appropriately fill the capsule and anterior chamber, ensuring the delicate capsule remains inflated to enable IOL implantation within the capsular bag [15].

The concluding incision closures step involves ensuring that the paracentesis and main wound incisions are not leaking and that appropriate surgical fluids are removed from the eye. All viscoelastic must be aspirated from the capsular bag and anterior chamber. This can be performed using Irrigation and Aspiration probes, taking care to avoid contact with the iris, capsule or cornea [15]. Afterwards, the incisions are closed by injecting BSS into the corneal stroma. If the resulting swelling does not create a watertight seal on the incisions, sutures can be used to prevent bacteria ingress or infection [14]. The operation is then concluded, having safely replaced the cataractous natural lens with an artificial intraocular lens!

2.3 Statistical Instrument and Phase Recognition in Cataract Surgeries

Understanding the surgical process of the phacoemulsification operation is key to grasping the difficulty of detecting the individual steps and phases involved using computer vision. Clues, such as identifying instrument presence in a frame or knowing the sequence of previous phases is insufficient, as phases often share instruments and could be repeated during a surgery. This makes phase detection from a single frame, or even a sequence of frames, unreliable. Surgical instrument identification is a much simpler object detection problem, but most applications involving image recognition in cataract surgeries require the sequence and timing of phases as this is how surgeons

typically refer to the operation components. Later in this thesis instrument activity is presented in a manner allowing the surgeon to draw their own conclusion as to which individual phases or steps of the operation are occurring, but over the last decade many considerable attempts have been made to automatically classify surgical phases. This section considers the statistical methods employed for phase and instrument classification, which perform arguably better for phase classification than recent deep learning methods. These statistical methods require "handcrafted" features and visual cues, where researchers manually engineer predictive patterns and features from cataract surgery videos for the classification models. This approach phased out as deep learning was introduced, as it was found that using computer algorithms to automatically determine these predictive features of the data was more successful for biomedical image recognition tasks [4], although the statistical models maintain very high predictive performances.

Many types of handcrafted features were extracted across various studies. The research paper published by Lalys, et al. is one of the earliest studies using statistical methods to characterise cataract removal surgery phases, published in 2012, and uses five distinct image cues. To extract visual cues, shape-oriented cues and texture-oriented cues, they respectively used a simple histogram intersection, a Haar classifier and a bag-of-words approach using local descriptors. The remaining two image cues were extracted using a conventional image classification approach including a feature extraction process, a feature selection process and a supervised classification with a Support-Vector Machine (SVM), providing image features representative of image cues. They then used these five image cues as features in a Hidden Markov Model (HMM) to classify across 12 different surgical steps, and found an accuracy of 91.4%. After creating feature sequences, they used Dynamic Time Warping (DTW) to measure the distance of an image to the center of a cluster of pre-labeled images of the same step. This measures how similar the image is to the average image of each step, and predicts the step that is the most similar. The DTW model found an accuracy of 94.4% across

the 12 surgical steps [17].

Then, in 2014, Charriere, et al. published a paper describing the effects of using motion and gesture features as handcrafted features in the statistical predictive models. They used two separate feature vector types: Bag of Words (BoWs) and Motion Histograms (MHs) feature vectors. They used Space-Time Interest Points (STIPs) to generate the BoW feature vectors. To craft the motion histogram feature vectors they extracted features across a sequence of optical flows, tracking strong corners across frames. Motion and gestures are then described across a time sequence, and to compare these sequences they used a Bhattacharyya distance formula. They then apply a nearest neighbours algorithm to both feature vectors to predict the surgical phase and to test the predictive performance of each feature vector type. They found that using motion features greatly helped predictive performance over several surgical phases, but also showed no improvement on several surgical steps where little movement was employed by the surgeon [18].

Later that year Quellic, et al. published a follow up paper proposing the use of a novel Conditional Random Field (CRF) model in place of HMM's for phase classification using motion and instrument observations over sub-sequences of frames [19]. This approach would identify idle and active phases, and then try and classify the active phases. They were able to design a specialised CRF model for their problem, the first framework designed to run in real-time during a cataract removal operation. They found that this had some improvement, and several years later in 2017 Charriere, et al. published a paper analysing a phase classification model consisting of a Bayesian network (BN) and two CRFs [20]. The BN accounted for the general order of phases over several frame sequences, and the CRFs were used for classifying visual cues within the frame sequences. They compared this model against two other models, a BN + HMM model and a Hierarchical Hidden Markov Model (HHMM) (improved versions of the model proposed initially in [19]). They found that the BN + CRF model outperformed the other two models, both when presented with either Motion Histograms or

instrument presence feature vectors. In fact, they found that the BN + CRF model, using only manually provided instrument presence features, presented an extremely high mean AUC of 0.980 for phase classification in real-time [20].

It becomes evident that features like motion and gestures, or the initial visual cues, shape-oriented cues or texture-oriented cues do not give as high surgical phase classification accuracy as simply providing instrument annotations of the surgery. This instrument presence feature vector, at the time these papers were conducted, had to be manually labeled by experts, reducing its practicality in real applications. Statistical methods using only visual cues and motion features gained a maximum AUC of 0.856 for phase classification [18], and in comparison the manually provided instrument annotation features gained a maximum AUC of 0.980 for phase classification [20]. This high result encouraged researchers to build models to attempt to identify instrument presence with object detection models to automatically label instrument annotation feature vectors for phase detection and move focus away from the classical handcrafted features.

2.4 An Overview of Deep Learning Architectures

This thesis relies heavily on the use of deep learning architectures and theories that have been developed over the last several decades. This section contains an overview of deep learning, an explanation of the models involved throughout the rest of this chapter, and a deeper analysis of several architectures that are implemented later in this thesis.

The original Multilayer Perceptron (MLP) deep learning model, shown in Figure 2.5, consists of an input layer, one or more hidden layers, and an output "classifying" layer. Each layer is considered a "fully connected" layer, and maintains a list of weights for each neuron it consists of. Figure 2.5 depicts these weights as arrows, traveling from neurons in one layer to the neurons in the next. Each neuron contains a single value commonly referred to as an activation value, these are derived from the activations and

weightings of the previous layers, and flow forward through to the next layers. In the context of computer vision, an input image is entered into the first layer, where each input neuron would have its activation value set equal to a corresponding pixel value from the image. Next, the activation values for each neuron in the following layers are calculated using equation 2.1, where $a_{i,j}$ represents the j^{th} neuron activation value in the i^{th} layer, $w_{i-1,j}$ represents the weight value from the j^{th} neuron in the previous layer leading to $a_{i,j}$, and $a_{i-1,j}$ represents the activation value of the j^{th} neuron in the previous layer. Put simply, each activation value is equal to a weighted sum of the activation values in the previous layer. The activation values are normalised using a typically non-linear "activation function", to ensure they sit on the range $[-1, 1] \in \mathbb{R}$.

$$a_{i,j} = \sum^{j} [w_{i-1,j} * a_{i-1,j}] \tag{2.1}$$

Each layer now contains an array of neuron activations that represent patterns and features within the input image; each set of activations can be thought of as a feature vector. As the activation values flow through the model, each layer draws higher level features from the previous feature vector. The final layer's activations, which represent certain classes, say "cat" or "dog", indicate the probabilities that each class is present in the input image, based on which highest level features were found by the last hidden layer.

During training, the weight values in each layer are adjusted after each batch of training instances (each batch of images, in this example). This is commonly performed by calculating the gradient of descent over the multidimensional space of neuron weights to minimise a loss function. This results in gradually stepping the neuron weights toward an optimal point, such that each layer learns to extract the most predictive features as possible from the previous layer. Other deep learning architectures largely expand upon this MLP model, with some differences.

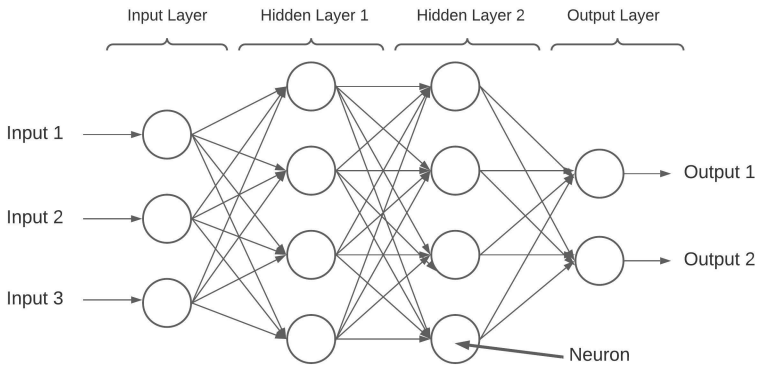

Figure 2.5: Multilayer Perceptron (MLP).

The Convolutional Neural Network (CNN) family of models use convolution layers to extract relevant features, using specific kernel matrices in a sliding window approach to reduce the number of weights within a layer. For example, in the equivalent fully connected layer, for two layers with an n number of neurons, the first layer will contain n^2 weight values, as each neuron is connected to each neuron in the following layer. In comparison a 3x3 kernel matrix contains a constant nine weight values. This matrix will be applied using a sliding window to an entire layer of neurons to gain the activation values of the following layer. The smaller number of weights allows for faster training and the calculation of activation values, allowing for much more efficient models [21].

Several architectures within the CNN family are the ResNet-50, ResNet-152 [22] and InceptionV3 networks [23]. This thesis implements these three deep learning architectures in later chapters, and so it is worthwhile to analyse the structure of these models. In a similar fashion to the MLP model, shown in Figure 2.5, these three models consist of an input layer, an output layer and a large number of hidden layers that pass feature vectors through the model, gradually extracting and refining higher level features. These hidden layers are grouped into "blocks", where each block is a closed system and has a single feature vector input and output. The ResNet and Inception models are made up of a certain number of these identical blocks, streamlining the

architecture design. Figure 2.6 shows simplified diagrams of the ResNet block and the Inception block.

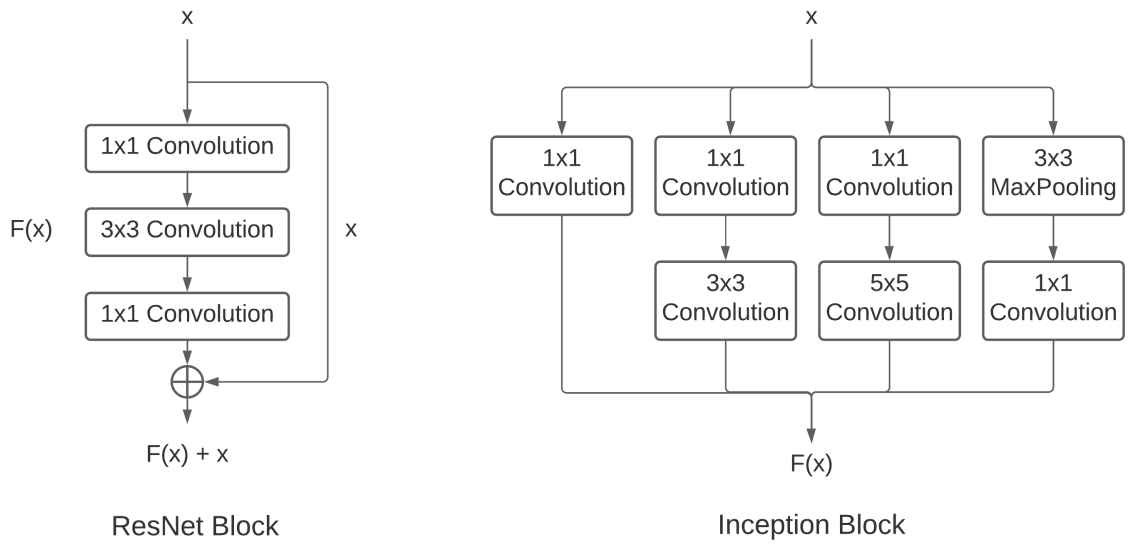

Figure 2.6: Simplified ResNet and Inception Blocks.

The ResNet block is similar to the classical MLP architecture, although it uses convolutional layers rather than fully connected layers, and contains an additional "skip connection", depicted as the arrow passing around the $F(x)$ function of convolutional layers. The feature vector input, x, is passed both into the $F(x)$ function and also through the skip connection into the $F(x) + x$ operation, representing the final output feature vector. This allows the $F(x)$ function to learn the weight values needed to set the function $F(x)+x = x$, resulting in $F(x)+x$ becoming an identity function. Training deep learning models primarily involves calculating the gradient of descent for each layer by using the calculus chain rule. Each layer adds another function to differentiate, growing the chain rule expression and further diminishing the gradient. This results in the common "vanishing gradient" problem, where the gradient quickly diminishes to zero, and the earliest layers do not have their weights updated. Now, as $F(x)$ is (or is close to, while still extracting useful features) the identity function, the chain rule expression can contain a much greater number of functions without diminishing the

gradient, and, hence, the model can contain many more blocks of layers [22]. ResNet-50 uses 50 ResNet blocks, while ResNet-152 uses 152 of these blocks. ResNet-50 tends to run more efficiently than its counterpart, due to having less parameters and complexity, while the predictive performance between these models tends to depend on the specific task.

The Inception block uses multiple convolutional layers within each block. A convolutional layer consists of a process where a kernel matrix is stepped across an input feature matrix, multiplying element-wise and creating an output matrix of reduced size. This results in reduced complexity and trainable parameters, while still maintaining useful features and detail. Kernels with manually defined elements can be used, but usually deep CNN models learn the optimal kernel values during training. The dimensions of the kernel is important, and the convolution layers are often classified by these dimensions, for example in Figure 2.6 the differing sizes of the layers within the Inception block can be seen (i.e. 1x1, 3x3) [21]. It is not obvious which size of kernel matrix should be used, so the Inception block combines a number of differing kernels to maximise accuracy, including a Max Pooling layer. These are then combined to create the output function $F(x)$. In the InceptionV3 architecture there are several variations of this block throughout the model. During training, the model trains the convolution layers within each block by adjusting the kernel values, and hence the blocks work together to produce the highest-level features for the output classifying layer, similar to the MLP model [23].

Both ResNet and Inception architectures tend to perform best with a large number of predictive classes. Both models performed exceptionally well when trained and tested on the ImageNet dataset, published in [24]. ImageNet contains around 1000 classes, and is a staple benchmark for new deep learning architectures. When implementing models for new applications, a large amount of training data is needed to achieve the maximum predictive power of these models. Often a technique called transfer learning or pre-training is used, where models are first trained on ImageNet, or another dataset,

and then trained on a more specific dataset for the application. This helps the model learn lower level features effectively for free, and saves the expensive specific dataset for the more significant higher level features.

The standard Recurrent Neural Network (RNN) uses previous instances within a data sequence as additional features within its layers. This allows time sequence features emerging across sequential instances to be taken into account. For example, in the context of cataract surgery videos, the RNN model will consider the fact that if one frame contains a particular instrument it is likely it will also appear in the following frame. Thus the following frames' prediction is reliant on the prediction or features of the previous frame. The Long-Short Term Memory (LSTM) model [25] is an improved version of the standard RNN model, that maintains instances both long term, over a long sequence of data, and short term, over one or two instances. This allows overarching patterns to be accounted for while also addressing smaller predictive factors between consecutive instances. The Gated Recurrent Unit model [26] is a counterpart to the LSTM model, working very similarly, although it attempts to overcome the vanishing gradient problem that LSTM models suffer from. GRU does have less complexity and runs faster than the LSTM model, although which of the two models achieves the higher predictive performance tends to depend on the specific task.

2.5 Deep Learning Instrument Recognition in Cataract Surgeries

The holy grail of instrument and phase classification is the real-time application: automatically working alongside the surgeon in the operating theatre offering instrument and phase information, or detecting complications early. They contain the potential to provide the surgeon with real time data so they can adjust and improve their technique on the fly, reducing complication rates and improving outcomes. The excitement around artificial intelligence and autonomous systems has deeply impacted the medical

imaging field [4]. This excitement, alongside the historical success of deep learning over statistical methods, has started a trend of classifying cataract surgery phases with deep learning. Unfortunately, statistical predictive methods do not appear to have been revisited since the promising paper [20] was published in 2016. Deep learning models tend to struggle to compete with the results of statistical models for phase classification, and only by reducing the number of classes by combining surgical phases can they achieve competitive performances. Instrument classification with deep learning has, however, seen great success with high performing models, due to deep learning being very well suited to object detection, its original usecase. This section will discuss the recent research into cataract surgery instrument classification using deep learning models.

The authors of the promising statistical methods paper [20], continued their research later that year by publishing the first paper involving the use of deep learning models to classify cataract surgical instruments [27]. They mention the M2CAI laparoscopic instrument identification challenge of 2016 (while still in the medical sector, this challenge is not related to cataract surgery) and use the top performing models, all based on deep learning techniques, as justification for building a CNN surgical instrument identifier. They note the absence of any motion features used within the challenge papers [28], [29], [30], [31], leading to the authors designing a multi-image fusion process within their CNN architecture that fuses a sequence of 16 frames (0.64 seconds of video) into a single image, accounting for motion features across the sequence. They found this process produced a significantly higher sensitivity and specificity than simply predicting instruments from single frames using a similar CNN. Classifying ten instrument classes found an AUC range of ($0.953 \leqq A_z \leqq 0.987$), a promising result for the first CNN instrument classifier.

The problem of instrument classification was then extended out to the wider research community with the CATARACTS challenge, issued later in 2017 [32]. This challenge provided a dataset containing over nine hours worth of cataract surgery video recordings, annotated by a panel of two expert surgeons, and is multi-labeled with 21

individual surgical instrument classes. The dataset contains surgeries performed on 50 patients and all videos have a resolution of 1920x1080 pixels, with a frame rate of 30 frames per second. This expertly labeled and diverse dataset has become the staple benchmark for instrument classification verification, and many other papers have since used this dataset for further research. Fourteen teams and the challenge organisers each contributed one or more architectures to best classify the dataset, competing to maximise the mean AUC over each instrument class. The results were published later in 2019, and they provide a thorough breakdown of the presented solutions, along with emergent patterns of success or failure. This challenge pushed the thorough research of instrument classification in a competitive and strictly structured environment, and provided the foundation *and* pinnacle of instrument classification in the field. The top ranked teams were, in order: DResSys, LaTIM, CUMV and TROLIS, and respectively provided a predictive testing mean AUC over all instrument classes of: 0.9971, 0.9931, 0.9897 and 0.9812. No teams attempted to build surgical phase classification models.

DResSys was developed at D-Wave, and consists of an ensemble of CNN models for individual frame classification and several temporal smoothing techniques for post-processing classifications. They jointly used ResNet-50, Inception-v4 and NasNet-A at the frame level, using pre-trained weights on the ImageNet database. They extracted a sub-dataset from the provided dataset, sampling in a fashion that maximised balance across classes and used this, in conjunction with image augmentation, for training the ensemble models. It appears that the success of this team is largely dependent on the smoothing techniques they employed, explicitly the use of a Markov random field to capture the dependence of frames on the time series. This ensemble and smoothing technique resulted in the highest mean AUC found by any team in the challenge [32].

The LaTIM architecture was developed by the challenge organisers, and consisted of a frame level CNN ensemble with an additional RNN ensemble to process the frame classifications. For training they used a novel boosting technique [33] for all CNN and RNN networks. In a first edition of this architecture they used an Inception-v4, an

Inception-ResNet-v2 and an o_O network for the frame classification, and an LSTM and GRU networks for the RNN ensemble. For the second edition they trained a single NasNET-A network for the CNN ensemble and they used three LSTM models in the RNN ensemble. The advantage of this second edition being that the architecture becomes unidirectional, allowing for online, real-time applications [32].

CUMV consists of an ensemble of two CNN's, pretrained on ImageNet. These two models, ResNet-101 and DenseNet-169, are trained independently, and a gate function [34] is used to compute the inner product of normalised prediction confidences for each instrument, inferring a prediction at the frame level [32]. TROLIS differs from other architectures as it develops targeted models for instrument groups, dividing classes into common, rare, and very rare. The common instruments were classified with an ensemble of ResNet-50 models, trained on differing size images. The rare instruments had an ensemble of three feed-forward ResNet-50 models, and the rarest instrument category, containing only the Biomarker instrument, used a statistical method which pulled out representative visual cues from the images. This strategy did not result in high predictive performance for the rare and rarest instruments, with the top three teams considerably outperforming TROLIS on these classes. Thus, the four top performing architectures submitted to the challenge differ substantially but provide similar testing performances.

The organisers of the challenge also drew trends from the teams architectures that contributed to success or failure. They had five "steps to success", the first being that successful teams tended to keep full videos aside for validation. Then the use of data augmentation in training: random horizontal flipping, random cropping, random scaling, random rotation and random shifting were used throughout the top teams, with the random horizontal flipping being consistently applied by all. They also noted the trend of using the latest CNN models, specifically their deepest versions, and the use of multiple models within the architecture. Finally they noted that the top teams also tended to use temporal smoothing techniques. Additionally, they noticed that

several factors did not seem to influence the success of an architecture: the number of selected training frames, the types of data augmentation used, the CNN's input image size or the use of test-time data augmentation. They also found that more sophisticated architectures did not necessarily do well unless the five "steps to success" were followed [32].

This challenge involves the generally field leading instrument identification architectures to date, with few other papers attempting to develop competing models for this context. After the papers publication, research focus returned to the subject of phase classification, as described in the next section, researching the use of instrument annotations and alternatively the use of video context as feature vector inputs.

2.6 Deep Learning Phase Recognition in Cataract Surgeries

In wake of the CATARACT challenge, the authors of [35] took the instrument classification findings and designed an instrument classifier and two experimental phase classifiers for surgical phase detection. They relabeled the CATARACT public dataset by surgical phase for training and verification. Echoing the ideas of the statistical paper [20] that found manually provided instrument annotations gave high results for phase classification, they designed their architecture to have an instrument classifier leading into a phase classifier, making phase predictions from time sequences and automatically provided instrument annotations. Their instrument identification model consisted of a ResNet-152 model pretrained on the ImageNet dataset, feeding into a fully connected output layer with 21 instrument class outputs. This multi-label predictive network followed three of the five "steps to success" outlined in the challenge paper: applying data augmentation, keeping full videos aside for validation, and using the deepest versions of off-the-shelf predictive models. They did not, however, follow the remaining two rules, due to only using a single model and applying no temporal smoothing techniques to

the multi-label predictions. They divided the CATARACT public dataset into train, validation and holdout sets (80-10-10%), for both instrument and phase labels. They report a mean AUC of 0.9959 for instrument classification on their holdout set, and on the CATARACT test dataset, a private test set withheld by the challenge organisers for evaluating the team performances, they scored a mean AUC of 0.9769 for instrument classification, landing among the top team results.

The main contributions of [35], however, are the two phase classification deep learning models. They developed two Recurrent Neural Network (RNN) models, which took the binary multi-label instrument annotations and the last instrument feature pooling layer provided by the ResNet-152 model as separate inputs. The authors argue that using these feature vectors as inputs provides visual features such as motion and orientation, and visual cues, such as lighting and colour, that could potentially enhance phase detection. They specifically use RNN networks due to the sequential nature of phases; RNN's consider a time sequence of instances for predictions, allowing adjacent surgical phases within the surgery video to influence the classification of a phase. The first RNN model consisted of a Long Short-Term Model (LSTM) and the second a Gated Recurrent Unit (GRU) model. Both models were trained independently on both binary instrument annotations and feature inputs, with sequence lengths of around 33 seconds, argued as a reasonable choice due to phases tending to continue for several minutes. The LSTM model, trained on the CATARACT public dataset and tested on the CATARACT test dataset, achieved accuracy's of 68.75% when given binary inputs and 78.28% when given the feature vectors. In comparison, the GRU model achieved accuracy's of 71.62% when given binary inputs and 68.96% when given feature vectors. Thus, GRU had a higher predictive performance on binary instrument annotations, while the LSTM had a higher predictive performance on the instrument feature vectors.

In early 2018, Primus, et al. compared three CNN models for phase classification, notably without the use of explicit instrument annotations as input features [36]. The

three CNNs were based on the GoogLeNet architecture, and were all trained from scratch (no pretrained weightings) on a cataract surgery dataset that was also introduced within the paper. The dataset had 175,488 frames in the training set, kept four full videos aside for testing purposes, and contained class labels for ten individual surgical steps. The first model was trained simply, applying only data augmentation to the training set, and was used as a benchmark performance. The second model was trained on a purified and subsequently balanced dataset, and the third model added a fourth colour channel to the input images which contained a current frame ratio timestamp, allowing temporal information. All models were trained over 50 epochs, keeping the model with highest predictive performance on the training set. It was found that the three models sequentially outperformed each other on the testing dataset: the benchmark model scored a sensitivity of 0.53, the second model scored a sensitivity of 0.67 and the third scored a sensitivity of 0.72. While it is difficult to compare the phase classification models proposed in both [35] and [36], due to differing performance metrics and datasets, they both provided arguably low results for phase classification.

The following year in 2019 the critical question was posed by the review paper [37], can cataract surgical phases be sufficiently detected through deep learning techniques? To answer this, the paper experiments with five algorithms with differing input data: (1) a support vector machine input with cross-sectional instrument annotation data; (2) a recurrent neural network (RNN) input with a time series of instrument annotations; (3) a convolutional neural network (CNN) input with cross-sectional image data; (4) a CNN-RNN input with a time series of images; and (5) a CNN-RNN input with time series of images and instrument annotations. Each algorithm was trained and evaluated with 5-fold cross-validation on a dataset gathered from 100 surgery videos and labeled by both phase labels and instrument annotations. They considered ten surgical phases and steps for classification. They found that the use of instrument annotations as input features performed better than video images alone, as seen in Table 2.1, algorithms (1), (2), (4) and (5) outperformed algorithm (3) with regard to AUC (although (3) can be

seen in Table 2.1 to have a high unweighted mean accuracy). They also found that temporal modeling of instrument annotations appeared to give higher performances than analyzing cross-sectional snapshots of instruments when comparing the results of algorithms (2) and (1), where the recurrent neural network (RNN) within (2) allows time sequence features to be considered alongside the instrument annotations. These findings reinforce the excellent statistical algorithm proposed in [20], which also relied upon manually provided instrument annotation data and implemented a Bayesian network for temporal smoothing. The paper concluded by arguing that sufficient phase classification performance could be achieved through deep learning methods, specifically by using instrument annotations feature inputs and considering time sequences of frames.

Metric	**Algorithm (1)**	**Algorithm (2)**	**Algorithm (3)**	**Algorithm (4)**	**Algorithm (5)**
Accuracy	0.938	**0.959**	0.956	0.921	0.915
AUC	0.737	**0.773**	0.712	0.752	0.737

Table 2.1: Summary Measures of Algorithm Performance for Phase Classification (partial table results), [37].

Key: Algorithm 1: SVM, Instrument Annotations; **Algorithm 2:** RNN, Instrument Annotations; **Algorithm 3:** CNN, Images; **Algorithm 4:** CNN-RNN, Images; **Algorithm 5:** CNN-RNN, Images and Instrument Annotations.

Another approach, conducted by [38], reduced the number of classifiable phases in an effort to only identify the major aspects of the surgical process. They trained an InceptionV3 model to distinguish frames between three phase categories: continuous curvilinear capsulorhexis (CCC), nuclear extraction and "other" (a collection of the remaining steps within the surgery). To train and validate the model they formed a novel dataset consisting of 303 individual cataract surgical videos from Tsukazaki hospital with frames sampled at rate of 1 FPS. The videos were annotated by an ophthalmologist with the start and end times of phases, and subsequently the frames were individually labeled. Similar to the accuracy of 95.6% produced by algorithm 3 from [37], which likewise consisted of a single CNN trained purely on image data, they achieved an accuracy of 90.7% for CCC, 94.5% for nuclear extraction, and 97.9% for other, with a mean

accuracy of 96.5%. A main consequence of this study was the ability to identify phases within five seconds of video, allowing real-time processing of surgeries. While limiting the number of classifiable classes to increase accuracy reduces the surgeon reference and annotation uses of this architecture, the authors argue that most surgical complications occur within the CCC and nuclear extraction phases, making the architecture more useful for real-time guidance and anomaly detection applications.

In addition to the real-time applications of cataract surgery phase classification, a recent paper [39] has proposed a full, end-to-end framework providing an online library resource that allows the submission and relevance-based retrieval of cataract surgery videos. They have a process allowing the processing of cataract surgery videos and then the dividing and categorising of relevant sections. This allows a query system to retrieve video sections based on instrument presence, phase classification and even a similarity search. An interesting part of this proposed framework is the processing of the videos prior to classification with deep learning models. Here several steps are taken from other relevant papers, including multi-scale deconvolutional neural networks for parsing and deblurring [40], instrument and eye detection with R-CNN masking [41], and active and inactive phase segmentation using ResNet-50. This process raises the effectiveness of using deep learning models to classify phases, instruments and action gestures to allow the indexing of particular sections of surgery videos.

2.7 Open Research Questions

Research into instrument and phase detection has greatly progressed, and now frameworks and predictive models exist that provide satisfactorily high results. Many tools and products can now be explored with these research developments to address the exciting possibilities and applications within the ophthalmic sector. Further research areas can be looked into, building on the groundwork of deep learning and statistical methods. These methods still need further research, especially in the area of generalisation. The authors of [42] focus on this issue in regards to instrument classification with

deep learning, and they found that their ResNet-50, InceptionV3 and NasNet Mobile models predict poorly when generalised across datasets. The three architectures scored recalls above 0.70 when trained and tested on the CATARACT public dataset [32], and when trained and tested on the Cataract-101 dataset [43]. However, when models trained on CATARACT were cross-tested on Cataract-101, and vice-versa, the recall dropped to no more than 0.28 for any model. While this paper offers an insight into the substantial problem of generalisation, many questions remain, including investigation of the generalisation of phase classification models, or the generalisation of higher performing recurrent deep learning models. Further datasets could also be developed and made public for comparison, offering more information to this area.

One of the limitations of this field of research is the lack of publicly available datasets. Currently there are a number of datasets that have been developed, including the CATARACTS dataset [32], the Cataract-101 dataset [43], the Brest cataract dataset [18] and unique datasets developed and used within the papers [17], [19], [20], [27], [36], [37] and [38]. However, only the CATARACTS dataset and the Cataract-101 dataset are publicly available, reducing the ability of researchers to conduct experiments in this field. Several of these unpublished datasets are incredibly diverse and robust, containing hundreds of videos performed by multiple surgeons, and could potentially train models far better at generalisation than the currently available public datasets.

An unresearched application of classification models is the real-time detection of clinically significant complications. The identification of posterior capsule ruptures, vitreous loss or other complications are often difficult to detect by the operating surgeon, especially inexperienced surgeons. Having a model running in real-time that is trained to detect problematic motion gestures, unexpected objects or other inferred patterns could be used to alert the operating surgeon and prevent complications. Another potential application for classifiers would be to deliver surgical annotations in real-time for observers. This may help to train new surgeons or provide information for assistants and causal observers. Finally, the use of a phase or instrument timeline could

be used to detect outlier or anomaly surgeries, which may indicate a higher chance of a complication having occurred.

Chapter 3

Constructing a Novel Cataract Surgery Instrument Dataset

3.1 The Need for a Novel Dataset

A critical step within the deep learning framework proposed in this thesis is to accurately predict the presence of instruments throughout cataract surgery videos. To generate instrument annotations, deep learning models must be selected and trained on an appropriate dataset; training data must be plentiful, correctly labeled and representative of the videos that the framework will be encountering in the real world clinical applications. As discussed previously in the background chapter, there are only two publicly available datasets, the CATARACTS dataset [32] and the Cataracts-101 [43] dataset. While the Cataracts-101 dataset contains a considerable 101 complete videos of cataract surgeries, the dataset is labeled by the current surgical phase, rather than by the presence of surgical instruments. It would need to be relabeled in order to be useful for training the instrument detection models proposed in this thesis. Although the CATARACTS dataset has been used to successfully train and test previous instrument detection models, difficulty in acquiring access and a preference for locally performed surgery videos led to the decision to sample and label a novel dataset for this thesis. A total of 39 cataract surgery videos were selected for the dataset, all performed by a

single expert surgeon. These videos well represent the range of videos expected to be encountered in the real world application of this framework, allowing the deep learning models to achieve an accurate and reliable predictive performance.

3.2 The Preprocessing Method

Deep learning methods operate by recognising features within data that signify particular results. In computer vision techniques, where an image or set of images comprises the data, often images are masked to isolate the most significant areas of the image. This removes pixels that have no relevance within the data, ideally allowing the deep learning models to more accurately and efficiently learn the identifying patterns and features within the images. A convenient attribute of Convolutional Neural Networks (CNN) is that the pixel location of a feature within an image is fairly insignificant, as the convolution kernel is iterated over the entire image with the same weights. This is opposed to classical fully connected layers, where features must be roughly in the same location to be recognised, as weights are trained for each individual pixel. Thus, maintaining the pixel locations of features is of much less importance with CNN models. The challenge of preprocessing, then, is to remove irrelevant pixels while keeping potential features as uniform as possible throughout the set of images.

Throughout the research into developing surgical phase and instrument detection models for cataract surgery videos, a number of different approaches have been used in preprocessing. However, many papers appear to use minimal or no preprocessing, and still see accurate results. An example of this are the findings published in the instrument detection challenge paper [32], which does not note the use of any customised image preprocessing in any of the fourteen competing models, other than image resizing and model specific preprocessing functions. This challenge does, however, achieve high predictive performance. It should be noted that data augmentation, where training data has some level of randomness injected in the form of shearing, zoom, rotation or other effects to synthesise additional training images, is not considered image preprocessing

in this thesis.

Similarly to predictive models themselves, image preprocessing can be achieved through handcrafted algorithmic methods or through deep learning methods. For both methods, the objective in the context of cataract surgery videos is to create a mask for each individual frame which best isolates the relevant areas of the image, placing black pixels in place of irrelevant pixels. Thus the deep learning models can ignore areas surrounding the eye of the patient and learn more significant features.

Mask R-CNN [41] is a deep learning framework designed to perform both object detection and pixel segmentation in parallel. This allows the framework to quickly find Regions of Interest (RoI) within an image, and both classify and plot bounding boxes around these pixel areas. These bounding boxes can be applied as image masks, and consequently Mask R-CNN can be used as a preprocessing step to isolate RoIs within individual frames. The paper [39] uses this method to segment the cataract surgery frames, filtering out irrelevant data from the images. The effect of this method can be seen in Figure 3.1, where the iris and instrument have been successfully detected and the image masked accordingly. This paper worked on classifying four primary surgical phases using a ResNet-50 model. A similar paper, [38], also classifies two of the same phases using InceptionV3, but does not use any preprocessing. While it is not conclusive to compare such different papers, due to differing models, datasets and performance metrics, [39] appears to have higher predictive performance for predicting the phases Capsulorhexis and Phacoemulsification. They scored precision scores of 99% and 99% respectively, while [38] scored accuracy scores of 90.7% and 94.5% respectively. This indicates that the Mask R-CNN preprocessing step performs better than without.

There are also several non-deep learning algorithmic preprocessing methods that have been applied within the relevant cataract surgery literature. Several early papers use Motion Histograms as features when making predictions, and so a common trend is the registering of all pupil centers to the same location using simple coordinate system changes. This results in the normalisation of the location of the pupil throughout

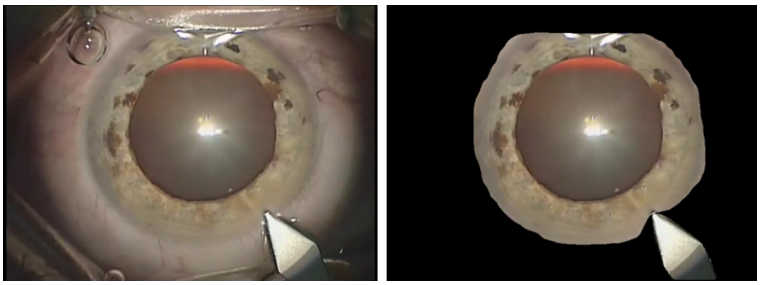

Figure 3.1: Mask R-CNN applied to cataract surgery frame, source: [39].

a video. This is done to minimise the movement of the patient or camera during the duration of the video, generating better features from instrument gestures and movements [20], [44], [18]. Zoom level is also estimated for each frame based on the illumination reflection in the center of the pupil. The frames are resized so that each pupil has the same diameter. Due the pupil dilation potentially changing throughout the surgery, each individual frame must be processed in this way [20], [44], [18]. Another preprocessing method is the application of a circular mask around the pupil, with some preset diameter size. This helps remove irrelevant data from the outer edges of the frame, and is very similar to the Mask R-CNN mask, although it can cut larger proportions of the instrument out of the frame [20], [44], [18].

The preprocessing method used in this thesis is based upon the non-deep learning methods discussed above. While Mask R-CNN is likely to provide greater results than the algorithmic methods, bounding boxes would be required to be drawn around the irises and instruments within thousands of training images, a large investment of time and infeasible within the scope of this thesis. Instead, the preprocessing takes the form of image resizing, pupil detection, masking, and shifting, following the procedure described in [18]. Zoom level estimation and subsequent image resizing is not included within this process. Although the shifting and centering step is intended for frameworks using Motion Histograms, it is cheap on processing resources and helps to normalise the frames for the deep learning models. Figure 3.2 shows the process followed within this preprocessing method. Figure 3.2a shows the initial raw frame, Figure 3.2b shows

the frame after pupil detection, Figure 3.2c shows the frame after masking, and Figure 3.2d shows the frame after centering, in its final form.

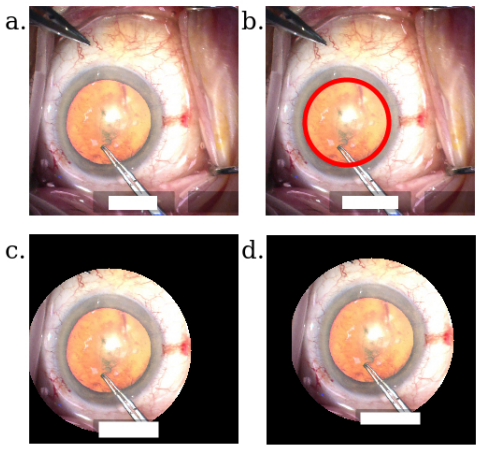

Figure 3.2: Preprocessed frames with identifying information removed.

Each frame begins the preprocessing method by being cropped along the larger horizontal direction to produce square image dimensions. They are then each resized to a 224x224 resolution or a 299x299 resolution, depending on the requirements of the deep learning model being used. This is done first to reduce the size of the images throughout the rest of the preprocessing steps, saving CPU processing resources. To detect the pupil location within each frame, the Hough Circle Transform algorithm [45] is used. This algorithm identifies circular shapes within imperfect images, presenting several candidate circle locations for each frame. These candidates are then filtered by restricting the minimum and maximum radius sizes, the threshold values used by the Canny Edge detector (a critical step of the Hough Transform algorithm) when finding the candidates, and the accumulator value, which can be thought of as a confidence value of the candidate circle. The optimal values of these parameters were found through experimentation. Additionally, due to the irises outer edge, the limbus, often being mistaken for the pupil's outer edge, the outside face of candidate circle regions must be darker than the inside face. After these constraints were applied, it was found that candidate circles were still occasionally identified outside of the pupil, often resulting

from circular patterns of visible blood veins or round areas of the eye lid. To restrict these misidentified candidate circles further, a sliding queue window was implemented and was run over the frames. This queue maintained a set of candidate circle coordinates drawn from the previous several frames in the video. When a new frame was processed, a candidate circle was found by the Hough Circle Transform algorithm and added to the queue. Then, if the candidate circle was centered more than 30 pixels away from the average center of the circles within the queue, the average circle was used in place of the candidate circle for the new frame. Through experimentation it was found that this dramatically reduced the number of candidate circles that were centered far from the pupil center, but also responded slowly when the patient abruptly moved their eye. It was also found that a queue size of three, consisting of the current frame's candidate circle and the two previous frame's circle locations, gave optimal results. Figure 3.2b shows a representation of a finalised candidate circle found by the Hough Circle Transform and passed through the filtering constraints.

Figure 3.3 shows a manufactured demonstration of the queue system. Three frames are shown, they are in consecutive order and are extracted from a larger sequence of frames. The three red circles represent the candidate circles found by the Hough Transform Algorithm for each frame, and are members of the sliding queue (which is at the maximum size of three). In 3.3c, it can be seen that the red circle is incorrect, and is more than 30 pixels away from the average candidate circle of the previous two frames. The blue circle, seen in 3.3c, is the corrected candidate circle, which is calculated by taking the average location of these three red circles. As can be seen, the blue circle is also not fully aligned on the pupil, but is much closer than the red circle. The blue corrected circle then replaces the incorrect red circle in the sliding queue, and the process continues for the next frame in the sequence.

Masking the image after pupil detection is straightforward; a two dimensional array is created with the same size as the frame, and from the center point of the candidate circle, with a given radius, array elements are selected to form a filled circle. This

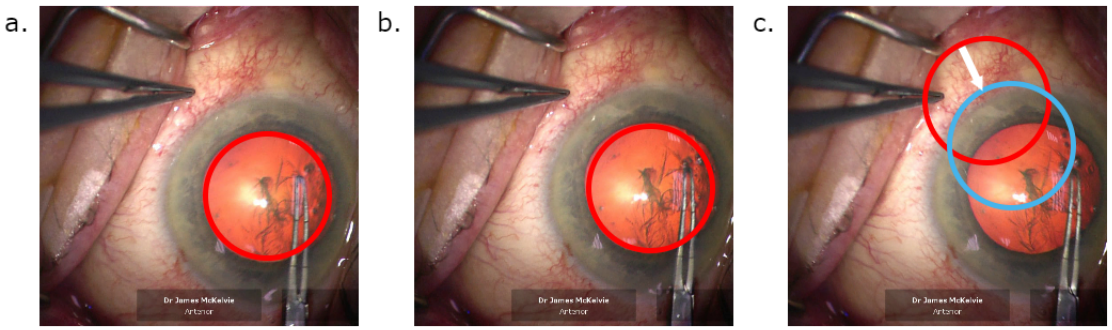

Figure 3.3: Sliding queue example.

mask is then applied to the original frame, and pixels not in the mask are set to black, while those pixels in the mask area are left untouched. The mask radius is set to twice the size of the pupil radius, chosen during experimentation as an optimal value. This effect is shown in Figure 3.2c. Next the entire image is shifted via a coordinate system change, to set the center of the candidate circle to the center of the frame. This effect is shown in Figure 3.2d. Thus, the frame has been correctly preprocessed and is ready to be classified by deep learning models.

A programming error found in the preprocessing script resulted in an estimated 2-3% of frames failing to be preprocessed. This occurred when the Hough Transform Algorithm could not find any candidate circles, and so could not continue the following steps. This was fixed by using the sliding average circle as the new candidate circle in these cases. However, throughout the later experiments in this thesis, frames are processed in batches of 50; occasionally the first frame in the batch cannot find any candidate circles and there is not yet an average circle to use. These frames also fail. This was tested to affect 1.16% of frames. These 2-3% of frames were dropped from the training and testing datasets of the first set of 29 videos and do not count towards model predictive performances. Likewise, the second set of 10 videos was affected by the 1.16% loss, and the experiments using this dataset exclude these dropped frames.

As a side note, while the radius of the pupil is known, there has been no effort made to normalise the image zoom across all frames. This is because patients may

have varying pupil sizes, and pupil sizes can change during the course of a video. Other papers have estimated zoom by measuring the width of the camera light mirrored in the pupil [18], but due to time constraints this has not been replicated within this thesis. The effect of frames not having a normalised zoom level is mitigated during training of the deep learning models, where additional frames are generated with varying zoom levels.

3.3 Video Labeling Process

The dataset originally consisted of 29 cataract surgery videos, which were used in the training of the deep learning models. Later an additional 10 videos were also labeled and used for testing purposes, bringing the total number of videos to 39. This large set of sampled videos represent the wider range of videos expected to be encountered in the practical application of the deep learning framework. The large dataset allows for greater generalisation of the deep learning models as well as ensuring the many situations and any variability within the surgeries are encountered during both training and testing. The deep learning framework developed in this thesis is designed to predict instrument presence within video frames, as such nine instrument classes and one idle class were defined. These nine instrument classes are shown in Figure 3.4, and consist of forceps, 2.4mm keratome blades, 1.1mm paracentesis blades, rycroft cannulas, choppers, phacoemulsification probes, irrigation and aspiration probes, rhexis forceps and lens cartridge injectors. Also present in several videos are malyugan ring injectors and toric axis markers, which are not considered in this thesis and are not included in the training and testing datasets due to an insufficient amount of video footage.

The first set of 29 videos was labeled separately from the later additional 10 videos, and will be discussed first in this section. Frames were sampled from each video at a rate of six frames per second, and if two sequential frames were considered perceptually identical then the second frame was dropped from the video. This has no effect on training the deep learning models, and allowed for more concise datasets. This dataset

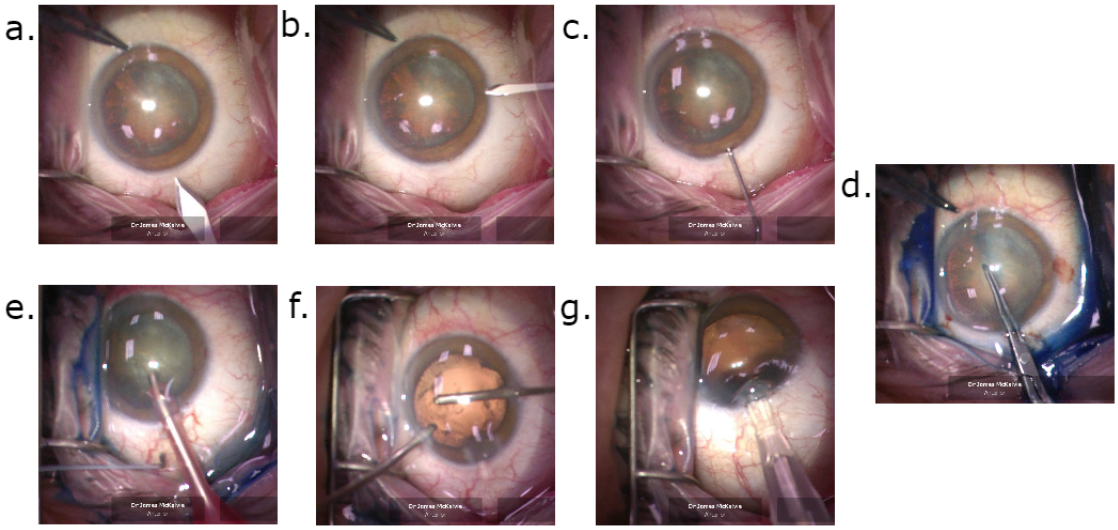

Figure 3.4: Common Instrument Examples.

Key: a. (top) Forceps, (bottom) 2.4mm Keratome Blade; **b.** (top) Forceps, (right) 1.1mm Paracentesis Blade; **c.** Rycroft Cannula; **d.** Rhexis Forceps; **e.** (bottom left) Chopper, (center) Phacoemulsification Probe; **f.** Irrigation/Aspiration Probes; **g.** Lens Cartridge Injector.

of sampled frames was then preprocessed following the complete method described in the previous section. From this preprocessed dataset an initial 10 videos were labeled by hand into binary active and idle classes, where if an instrument was engaged with the eye (touching or within the cornea) the frame was considered active, and if not it was considered idle. Notably, even if an instrument is in frame and visible, if it is not directly engaging the eye it is considered idle. This labeled subset from the 29 preprocessed videos was used to train a small ResNet-50 deep learning model, which was then used to label the remaining 19 videos of the dataset. These 19 videos were then corrected by hand, providing an accurately labeled dataset of 29 videos of active and idle classes. Testing of the ResNet-50 model on the 19 corrected videos (not used in training) provided a 95% accuracy rate for the binary active and idle classification task. The majority of incorrect classifications occurred in the transition between active and

idle phases, particularly when an instrument hovered over the eye before engaging it. In a similar study, the authors of [39] divided up the Cataract-101 dataset into active and idle phases using a ResNet-50 model, finding an accuracy of around 93%.

The main purpose of labeling the dataset into binary classes was to firstly test the effectiveness of the ResNet-50 model on the preprocessed cataract surgery videos, but also to allow for faster labeling of the active phases into the nine instrument classes. Using the correct active and idle phase transitions provided by the binary labeled dataset allowed greater sections of each video to be labeled at once, reducing the labeling time required. The 29 videos were then labeled by hand into the final ten classes, including the idle phase, providing 57,422 accurate multi-labeled frames. Unfortunately, the frames of these 29 labeled videos were stored in PNG format *after* preprocessing, making the ability to retrieve labeled copies of the original raw images very difficult. The preprocessing bug that was found to drop 2-3% of frames when no candidate circles were found does affect the first set of 29 videos. This was fixed after the creation of the dataset. This is not a significant issue, as the majority of these failed frames from the first set of videos show the eye at the start and end of videos, or the rare case where severe blur is disrupting the image.

The second set of 10 cataract surgery videos were sampled at the same frame rate as the first set of 29 videos, at a rate of six frames per second. They were labeled by deep learning models built upon the InceptionV3 architecture and trained on the first set of 29 videos, and had all ten class labels predicted at once. These deep learning models are discussed in depth in the following chapter. The frame labels were then corrected by hand, providing 32,865 multi-labeled frames. Unlike the first set of 29 videos, this second set was stored in PNG format *without* preprocessing applied, allowing additional testing and experimental ability.

3.4 Data Analysis

The 39 cataract surgery videos were recorded using a Zeiss Lumera 700 surgical microscope, at a rate of 25 frames per second (FPS). The videos had a resolution of 1280 x 720 pixels. Table 3.1 shows general statistics of the 39 videos, notably the average video duration of 10.79 minutes and the standard deviation of 4.65 minutes. This large standard deviation is due to several videos starting midway through the operation, and several other videos including optional surgical steps which minimally extend the operation duration. These optional steps are described in the background chapter and are the Vision Blue Dye step and the Mulyugan Ring or Iris Hook insertion step. These optional steps involve the Mulyugan ring injector and the Toric axis marker instruments, and frames containing these instruments have been excluded from the dataset due to only appearing briefly in several videos.

Average Video Duration	**Standard Deviation**	**Minimum Duration**	**Maximum Duration**
10.79 minutes	4.65 minutes	5.70 minutes	21.57 minutes

Table 3.1: Statistics of the combined set of 39 videos.

The frames of the first set of 29 videos were extracted using the Python library OpenCV, while the second set of 10 videos had frames extracted by the Python library MoviePy. As the videos were stored in MPG format, which, as part of its compression algorithm, does not include consecutive duplicate frames, there is a disparity between the number of frames within each set. A test revealed that when provided with the ten videos of the second set OpenCV retrieved 19,956 frames in total (ignoring duplicate frames), while MoviePy retrieved 33,270 total frames (keeping duplicate frames). On average, OpenCV retrieved 40.23% less frames than MoviePy. So while Set 2 contains a of third the number of videos as Set 1, by retrieving duplicate frames Set 2 consists of over half as many frames as Set 1. The effect of this on testing the deep learning models is minimal, the presence of or lack of duplicate frames is not expected to make any significant impact on predictive performance. It should be noted that if there is a higher

proportion of duplicate frames for a particular class this could place a higher weighting on this class during training, introducing bias. However, the second set of 10 videos is not used for training within this thesis. The deep learning framework, described later in this thesis, requires precise frame sequence and timing information and so Set 1 cannot be used for accuracy testing purposes; this was the primary motivation to create a second set of labeled videos.

Table 3.2 shows the frequency distributions of each class, noting that each frame may be labeled with one or more classes. These sets are considerably imbalanced, with the smallest combined class (2.4mm Keratome blade) consisting of 378 frames while the largest combined class (Idle class) consists of 26,107 frames. The blade classes have low frequency due to each incision lasting roughly two and a half seconds per video. It should be noted that there are likely more frames of the blade in frame but not engaged with the eye than in frame and engaged. In future work, more data instances could be gained if all frames containing the blade were labeled, rather than only those where it is engaged with the eye. The Forceps class has a similar frequency between the two video sets, implying that a larger number of duplicate frames exist in the episodes where this instrument is used during the operation.

This imbalance is a common problem in the research of Cataract Surgery videos, and the authors of [36] address it by augmenting their own dataset and training deep learning models for phase identification. They try two methods of balancing the dataset: purification and automatic balancing. Purification involves reducing the variability within each phase, and they do this by manually pruning frames within each phase that do not contain instruments. For automatic balancing they reduced class counts by applying random sampling, and increased class counts by applying random data augmentation techniques, such as duplication, rotation and scaling. They compared two CNN's, where one was trained with the unbalanced control dataset, and the other on the balanced and purified dataset. They found an average recall of 0.53 and 0.67 respectively, showing that the augmented dataset provides a higher predictive perfor-

mance overall. Most importantly, for the incision phase the recalls were 0.06 and 0.80, showing an impressive increase in accuracy for frames containing the blade classes. No automatic balancing or purification was applied to the datasets used within this thesis. The predictive performance found by the deep learning models was satisfactory without dataset balancing and there were some time constraints. The authors of [36], however, demonstrate the ability of data balancing to increase the predictive performance of deep learning models for cataract surgery phase classification.

Instrument Name	**Set 1 (29 Videos) Frequency**	**Set 2 (10 Videos) Frequency**	**Sets 1 & 2 Frequency**
Idle (No Instrument)	16,826	9281	26,107
Forceps	2750	2486	5626
2.4mm Keratome Blade	234	144	378
1.1mm Paracentesis Blade	270	220	490
Rycroft Cannula	8355	3638	12,038
Chopper	14,899	9860	24,759
Phacoemulsification Probes	15,219	9124	24,343
Irrigation and Aspiration Probe	12,157	6595	18,752
Rhexis Forceps	3342	2526	5868
Lens Cartridge Injector	750	492	1242
Total Frequency	75,063	44,801	119,603

Table 3.2: Dataset label frequency distributions. Set 1 refers to the first 29 cataract surgery videos, Set 2 refers to the second 10 cataract surgery videos.

Chapter 4

The Deep Learning Framework and Model Selection

4.1 The Deep Learning Framework

The primary contribution of this thesis is the novel deep learning framework. This is a system that takes as input raw cataract surgery videos, extracts and preprocesses frames, and analyses them using deep learning models to create clinical surgery reports. Figure 4.1 shows a flowchart of this process. First, in step one, video frames are extracted at some frame rate, typically six frames per second (FPS). These frames are then preprocessed in step two, using the preprocessing method described in the previous chapter. This prepares the frames for step three, where deep learning models provide a multivariate prediction vector which represents the likelihood of each surgical instrument being present in each frame. In step four this prediction vector is used to create Instrument Activity Sequences. These sequences mark consecutive frames where the same instrument is active in the cataract surgery video. Finally, in the fifth step, these Instrument Activity Sequences are used to create a clinical summary report, which provides a visual representation of the surgery process as well as statistical information and video annotations. The first four steps will be discussed in further detail in this section, while the fifth step will be discussed in the following chapter.

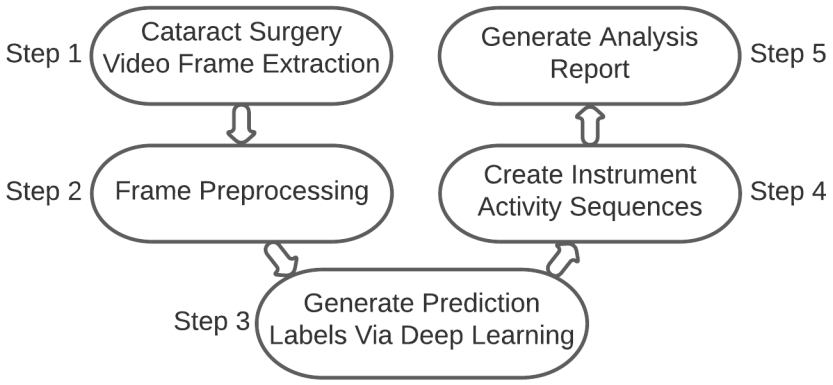

Figure 4.1: Flowchart of the deep learning framework.

There are several Python libraries for extracting frames from videos, including OpenCV and MoviePy. The former is a widely popular video and image editing package, supporting Python, Java, C++ and many other programming languages. Many OpenCV operations are also under active development for CUDA and OpenCL high-speed GPU interfaces, providing a promising future for automatic image editing. In the meantime, many of the operations in the Python version of OpenCV are simply wrappers for C or C++ based operations, which, as Python code runs slowly, provides an effective balance of speed and ease of programming. Unfortunately, when extracting frames from certain video formats, OpenCV fails to extract duplicate frames. This is due to some video formats compressing the video content by dropping consecutive near-duplicate frames from the video. This would normally be remedied by using the retained duplicate frames in the place of the removed frames during decompression, but OpenCV does not implement this function in its frame extraction operations. This creates an issue, as to match frames to time locations within the video an accurate frame count is required. One solution would be to use alternative video formats that do not act in this way, but to allow a more general video input to the framework MoviePy was opted as the frame extraction library. MoviePy is an open source library programmed in Python and runs slower than OpenCV, but retains duplicate frames removed by the

video compression algorithm. All 39 videos used in the evaluation of the deep learning framework were recorded at 25 FPS. Step 1, depicted in Figure 4.1, was programmed to allow an FPS extraction rate on the range $[1, 25] \in \mathbb{N}$. The frame extraction rate does not effect predictive performance, but does contribute to the analysis accuracy in steps 4 and 5. In a later chapter several extraction frame rates are compared to find an optimal rate, maximising accuracy and minimising run time. A default extraction rate of 6 FPS is used throughout the majority of experiments in this thesis, and extracts an average of 3890 frames per video.

The second step, depicted in Figure 4.1, involves running the extracted frames through the preprocessing process described in the previous chapter. This preprocessing script uses OpenCV and Numpy vectorisation operations where possible, avoiding resource consuming "loops" and lengthy Python computations. This tends to be the most resource consuming step of the framework, as large numbers of images are edited multiple times using CPU intensive operations. To optimise this process, the extracted frames are broken up into smaller sets of images. These sets are then processed with multiple threads, utilising the entire CPU. Memory locations for processed images is set apart before multi-threading to ensure the sequence order of images is maintained, as steps four and five require the frames to be in the exact order they were extracted in. Once all frames have been preprocessed and ordered appropriately they are moved to step three.

This step of the framework involves using deep learning models to generate multivariate predictions for each frame. Specifically, each frame receives ten confidence values, one for each instrument class (including the idle class). This confidence value falls on the range $[0, 1] \in \mathbb{R}$, and values above the conventional confidence threshold of 0.5 indicate that the class is positively predicted while values below indicate the class is negatively predicted. Due to the imbalanced nature of the dataset, the optimal threshold values differ for each class. By manually examining the prediction values of frames and through trial and error on test videos from the first dataset, a rough optimisation

was estimated and the values {0.5, 0.5, 0.3, 0.2, 0.7, 0.5, 0.5, 0.5, 0.5, 0.5} were chosen. The blade classes, under represented in the dataset, tended to need a lower threshold, while the Rycroft Cannula, often miss classified as other instruments, tended to need a higher threshold. It should be noted that the majority of predictions were either below 0.1 or higher than 0.9, and the threshold tuning only affected a small proportion of frames, generally at times when instrument activity sequences began and ended.

For each frame the deep learning models generate a multivariate prediction label consisting of a set of ten values, each determining the classification probability for the respective class. The Idle class is included as one of the classes, with its own probability. This setup allows for the contradictory situation where a frame can be both positively predicted idle and have a positive instrument prediction. The predictive values fall on the range $[0, 1] \in \mathbb{R}$, with a higher value representing a higher confidence in that class. The ensemble thus generates a multivariate prediction vector when provided all of the extracted, preprocessed frames, which is used in step four. This step also includes a small level of postprocessing. If a frame has no positive class predictions, the previous frame's prediction(s) are used. In the case that the there are no previous positive frame predictions the next positive frame's prediction(s) are used. The multivariate prediction vector is now rounded to multivariate binary labels, zero or one, using the class thresholds to determine positive or negative labels.

Instrument Activity Sequences are defined as a continuous sequence of frames containing the presence of a single particular instrument or the idle class. These are calculated by running through the multivariate predictions for each frame in time order, keeping track of consecutive sequences of positive predictions. Each sequence has the start frame and end frame IDs, the surgical instrument that is being recorded (or the idle class), and the number of frames within the sequence.

A grace window is applied when creating sequences: if there is a sequence of positive frame predictions, and a negative frame prediction is found, then the grace window is applied and if another positive frame prediction is found within the time period the

sequence continues as normal. A default two second grace window is applied. Many of the frames within Instrument Activity Sequences may not actually have a positive prediction of the class, because the grace window covers false negative class predictions within the surgery. The last frame of the sequence must have a positive class prediction as the sequence ends when no frames within the grace window time contain positive predictions. Figure 4.2 provides an example diagram for the grace window. Window 1 finds a negative prediction, but because a following positive prediction is found within the window, the sequence continues. Window 2 finds a negative prediction and no positive predictions within the time allowed, so ends the sequence on the final positive prediction.

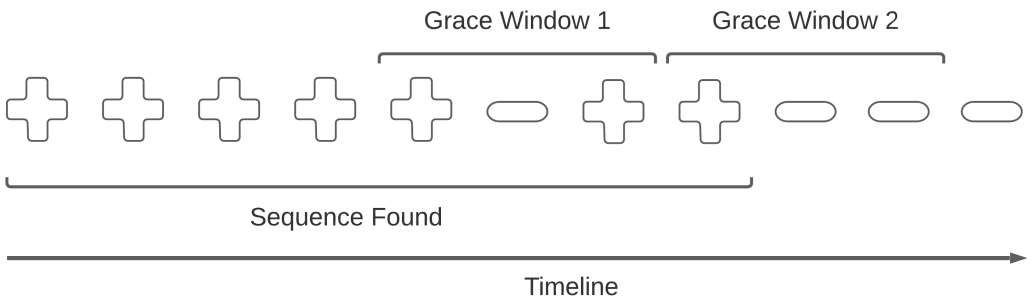

Figure 4.2: Example of grace window: Window 1 found a positive prediction, so continues the sequence. Window 2 did not find a positive prediction, so ends the sequence.

As each sequence only records one class, one or more sequences may run concurrently. This raises a slight issue in regards to the idle class, as some instruments are active for less than the grace window length, allowing the idle sequences to run through these sequences. For example, the idle sequences can overlap the blade sequences, as often a single incision takes less than a second to perform, not triggering an end to the idle sequences. This could be solved by implementing differing grace window lengths for each predictive class, reducing the grace window length, or forcing the idle sequences to "give way" to surgical instrument sequences.

Overall, the deep learning framework provides a system for extracting frames from

videos, preprocessing the extracted frames, detecting the instruments or idle state within those frames, and building Instrument Activity Sequences, tracking the presence of instruments throughout the entire video. Finally, the fifth step takes these sequences and generates clinical summary reports, visualising the surgery process and providing detailed statistical information and video annotations. This replaces the need for surgeons to manually review video footage, instead the system automatically summarises surgical videos and presents concise and representative clinical summary reports. This framework depends heavily on deep learning, leveraging the powerful predictive ability of modern machine learning architectures and methods to accurately analyse the cataract surgery videos. The selection, training and evaluation of these deep learning models is critical to the success of the proposed deep learning framework.

4.2 The Selection and Training of Deep Learning Models

The second chapter of this thesis, the background chapter, explores two primary areas of research into cataract surgery videos, instrument classification and surgical phase and step classification. Researchers have developed numerous statistical and deep learning models to gain the maximum achievable predictive performance for these two problems. Instrument classification peaked in the CATARACT challenge paper, as the highest performing team provided a near perfect AUC score of 0.9971 with an ensemble of CNN models. A number of other deep learning models provided similar results, including models entered by other teams to the challenge paper and the later research paper. Phase classification has not had the same consistent success, but did reach a maximum AUC score of 0.980 using statistical models, given manually provided instrument annotations. A deep learning approach, also using manually provided instrument annotations, achieved a lower AUC score of 0.773. Finally, another deep learning approach reduced phases from the typical ten to twelve steps into three categories, capsulorhexis,

nuclear disassembly and other. They scored an average accuracy of 96.5%.

Selecting between surgical phase or instrument classification for the deep learning framework required several attributes to be considered. While phase classification tends to be more often used by surgeons for the surgical categorisation of cataract videos, the lack of predictive accuracy and a lower number of classes makes this a less attractive option. Due to the high predictive performance and the detailed surgical analysis provided by instrument classification models, it was decided to build the deep learning framework upon instrument classification techniques.

Selecting an optimal deep learning architecture for instrument classification offered some variation of choice. The challenge paper offered 14 potential designs and architectures to choose from [32], and the paper [35] offers an additional approach. These architectures and their results are well discussed in the background chapter. The significant deep learning models tested throughout these approaches are listed: ResNet-50, ResNet-101, ResNet-152, DenseNet-169, InceptionV4, NasNet-A, Inception-ResNet-v2 and o_O. Using the proven predictive performance of these models for instrument classification as a guideline, we chose three CNN models to train and evaluate: ResNet-50, ResNet-152 and InceptionV3. The three chosen model architectures are described in the background chapter, with the two ResNet models consisting of sequences of ResNet blocks, and the InceptionV3 model consisting of a sequence of Inception blocks.

ResNet-50 is ideal due to its low complexity, it consists of five ResNet blocks of convolution layers and skip connections, resulting in a 50 layered model. Compare this to the ResNet-152 model, which is 152 layers deep, and so is much deeper and computationally expensive. The shallower network allows faster predictions and training, while still maintaining high predictive performance. The second model selected, ResNet-152, was chosen due to the authors of [35] using the model alone to perform instrument classification and gaining the high predictive accuracy of 99.07% over 21 instrument classes. The winning team of the challenge paper, DResSys, used an ensemble of ResNet-50, InceptionV4 and NasNet-A. As such, the Inception architecture is included among our

experiments. Keras, the Python package used to build the models in this experiment, does not offer InceptionV4 as default and it would need to be imported via open source builds. As InceptionV3 only achieves slightly less predictive accuracy than InceptionV4 on the ImageNet dataset [46], and is available through Keras, InceptionV3 is selected as the final model in the experiment.

Following the example of [35], all three models are pre-trained on ImageNet, where weights found by extensive training on ImageNet are loaded into the models prior to training on our datasets. ImageNet contains around 1000 individual classes of random objects: cars, cats, dogs, ect. The large amount of training data in ImageNet helps to train the layers of the models to identify predictive features. The low levels of the models are general, picking out lower level features, like straight lines or colour patterns. The higher layers of the models are more specific to the classes in ImageNet that are being classified, and in theory find features that may not be useful for the instrument classes that the models are intended to classify. As such, when training on our specific surgical instrument datasets, a number of the lower model layers are frozen, to stop the weights from being changed during this training. The higher level layers, however, are changed and adjusted during training on our dataset to better draw features predicting instrument classes. This technique allows the use of a much larger number of training images to be used, refining lower level features while maintaining specifically trained higher level features. Selecting the optimal number of lower level layers to freeze, however, requires some experimentation.

These pre-trained models are designed to output the predictions across 1000 classes of ImageNet. To redesign these for our instrument dataset, the final classifying layer is removed and several custom layers are added to bring forth the predictions for the ten instrument classes in our dataset. Figure 4.3 depicts the final model architecture. Images are inputted at either resolutions of 224x224 pixels for the ResNet based architectures, or 299x299 pixels for the InceptionV3 based architecture. Then one of the three chosen model architectures are implemented and the final highest feature vector

is extracted. This flows into a max pooling 2D layer to reduce the number of output activations, and is followed by a flatten layer to reduce the activations to a one dimensional array. This array then flows into the fully connected dense layer consisting of 10 neurons, one for each class. A sigmoid activation function simultaneously converts these activation values to the range $[0, 1] \in \mathbb{R}$, providing a probability that each class is present in the input frame.

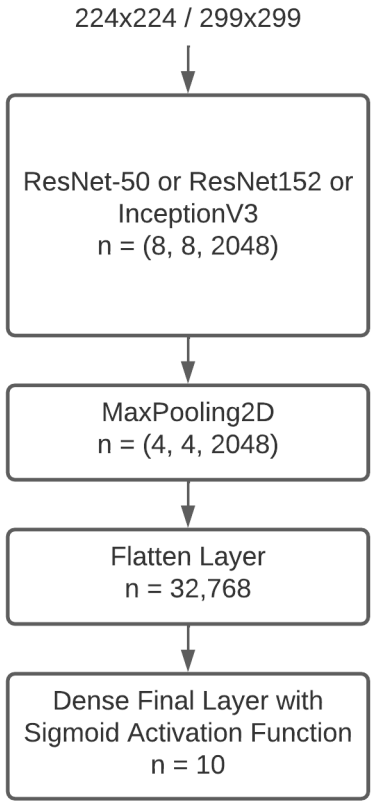

Figure 4.3: Model Architecture, with n output nodes for reference. Note: the example n output node sizes in this figure are referenced from the InceptionV3 based architecture and differ from the two ResNet based architectures, with the exception of the final $n = 10$ node size.

During training several hyperparameters can have a strong impact on the final predictive performance of the model. These include the batch size, which is the number of training instances used between each round of optimisation, where the weight values of the model are updated to gradually step towards optimal values. A higher batch

size gives a better idea of which direction to adjust weights, and usually results in less rounds of optimisation per training epoch. The learning rate is another hyperparameter, which dictates how much weights are adjusted during each round of optimisation. A higher learning rate may overshoot optimal values, while a lower rate may not reach optimal values and can get stuck in local optima points, halting training. Another hyperparameter is the number of training epochs that are conducted. Each epoch consists of a full run of all available training data. Training over multiple epochs allows for more training data, but also runs the risk of overfitting the model to the training data, where it essentially memorises images rather than learns patterns and features. To avoid overfitting, the "model checkpoint" callback was used within the training process. This callback saves a copy of the model at the end of each training epoch, and after all epochs have completed returns whichever model scored the top validation performance. Overfitting can be detected by observing where validation performance begins to drop while training performance increases, so this callback gives confidence that the last model version before overfitting occurs is returned as the final model. Unfortunately, during training it was unclear which epoch was used by the callback to provide the optimal model. Two factors remain, the choice of loss function and the choice of optimiser. The loss function is repeatedly calculated throughout optimisation, and there are a number of function choices available. For multi-label deep learning models, it is suggested to use the Binary Crossentropy function as the loss function. This is opposed to the common Categorical Crossentropy function, which is typically used for multi-class problems, where the model must be exclusive in selecting only one class per instance. The choice of optimiser is important as finding the optimal values of the model weights is essential. Commonly gradient descent will be used as the optimiser, but other methods, including using momentum based techniques or techniques that adjust the learning rate, can be more efficient or useful in a range of situations.

Rather than experimenting to find the best combination of hyperparameters, we have again followed the example of the authors of [35] and used the values they suggest

for their ResNet-152 based instrument classification architecture. A batch size of 8 was used, a learning rate of 1e-4 was used and the Adam optimiser was used to train all models. In surgical instrument classification, multiple instruments could be active at once, resulting in a multi-label classification problem. Hence, Binary Crossentropy is selected as the loss function. For training, 20 epochs were used. Each of the three model architectures were trained four times, with each variation testing a different number of frozen layers. The four variations included 0% frozen layers, 50% frozen layers, 75% frozen layers and 100% frozen layers. Each percentage freezes in the direction of the first shallow layers towards the later deeper layers, for example, 50% frozen layers implies that the shallow half of the model (closest to the input layer) will not have weights adjusted while training on our surgical instrument dataset. Thus we attempt to find the optimal balance between training ImageNet's robust low level features with our own dataset's higher level features. Table 4.1 shows the 12 total models that were trained across two dimensions, model architecture and percentage of frozen layers. Note that a particular identifying label is given to each model (e.g. model-0).

Model	**0% Frozen**	**50% Frozen**	**75% Frozen**	**100% Frozen**
ResNet-50	model-0	model-1	model-2	model-3
ResNet-152	model-4	model-5	model-6	model-7
InceptionV3	model-8	model-9	model-10	model-11

Table 4.1: Model Label Key.

These 12 models were all trained on the first set of 29 videos, which were hand labeled and preprocessed prior to training. The dataset was broken into four equal subsets, each subset consisting of either 7 or 8 videos randomly selected from the dataset. All instruments were present in all four subsets. Each of these subsets were then broken into a further two subsets: a training subset consisting of 90% of the frames within the subset, and a validation subset consisting of the remaining 10% of frames. The frames were assigned randomly. This follows the training, validation and testing technique known as 4-Fold Cross-Validation. This technique involves evaluating a single model

(e.g. model-0) four times, each time training and validating the model on a unique combination of three of the four subsets and then testing the model on the remaining fourth subset. This all four subsets are used for both training and testing with each individual model (e.g. model-0), rather than the classical approach of withholding a section of the dataset for validation or testing and not using it during training. This process adds an extra dimension to the experiment, as each model in Table 4.1 now has four variations, each trained on a different combination of three of the four subsets. These variations are labeled 'a', 'b', 'c', and 'd'. Table 4.2 shows the subsets used for training each variation and provides an example of the labelling convention.

	model-#a	**model-#b**	**model-#c**	**model-#d**	**model-#e**
Training	Subsets 1,2&3	Subsets 1,2&4	Subsets 1,3&4	Subsets 2,3&4	Subsets 1,2,3,&4
Testing	Subset 4	Subset 3	Subset 2	Subset 1	N/A

Table 4.2: Dataset Subsets and Model Label Key (e.g. the four variations of model-0 are model-0a, model-0b, model-0c and model-0d). As a side note, #e refers to a model trained on the complete dataset, evaluated in later chapters.

Additionally, during training several data augmentation techniques are employed to maximise the usefulness of our surgical instrument dataset. Training images are rotated up to $\pm 30°$, have a width and height shift range of 20%, a shear range of 20% and a zoom range of 20%. Images could also be flipped horizontally. This added variety and generalisation to the training set and was shown to increase accuracy.

The challenge paper [32], offered "five steps to success" to build a top performing instrument classification model for cataract surgery videos. These steps include keeping full videos aside for testing, the use of data augmentation during training, the use of the latest off-the-shelf deep learning models, the use of multiple different architectures working together in an ensemble, and finally the use of temporal smoothing techniques for post-processing. The deep learning architectures proposed in this thesis meet the first three of those five steps. A later experiment in this thesis tests the impact of applying the fourth step, using all three of our architectures in an ensemble, but tem-

poral smoothing techniques are only used to a limited degree when creating Instrument Activity Sequences. The evaluations of the ResNet-50, ResNet-152 and InceptionV3 based architectures selected and trained for the deep learning framework are discussed in the following section.

4.3 Evaluation Via 4-Fold Cross Validation

Each multi-label model was trained once on each of the four subsets shown in Table 4.2, providing four variations of each of the 12 models. These variations had validation tests performed at the end of each training epoch to track overfitting and accuracy throughout the training process. The data for these validation tests came from the 10% of frames set aside within each subset, with the remaining 90% of frames used in training. Once all four variations of a model were trained, they were each tested upon the remaining subsets and the scores were combined across all four variations. Thus, the performance metrics used to evaluate the models have been calculated for all four model variations at once, and every frame within the entire dataset has contributed to the final performance score for each model. The dataset was divided into four folds because initially the Malyugan ring injector instrument was included as a classifiable instrument and only appeared in four videos. Four folds allowed each fold to contain at least one video with the instrument present. However, the instrument was removed later due to not having enough frames and due to time constraints the dataset remained divided into four folds.

Area Under the ROC Curve (AUC) has been chosen as the primary performance metric for evaluating the instrument classification models. The Receiver Operating Characteristic (ROC) curve is a curve that is plotted across two dimensions: the true positive rate and the false positive rate of a model. An example of a ROC curve is depicted in Figure 4.4. The true positive rate is the percentage of positive predictions that were correct (an instrument was predicted to be present, and was present), and the false positive rate is the percentage of positive predictions that were incorrect (an

instrument was predicted to be present, but was not present). As the models provide a probability on the range $[0, 1] \in \mathbb{R}$ that an instrument is present, different threshold values can be used to determine if the prediction is considered positive or negative. As both rates depend upon the threshold value they can be plotted for all possible values of the threshold, creating a positive non-linear curve. As the threshold value increases, the confidence required to classify a positive prediction also increases, and as such there are fewer false positive predictions. However, there is also a greater number of missed true positive predictions, resulting in higher rates of false negatives. Thus, the threshold value can be optimised to maximise both the true positive and true negative rates. The dotted line, depicted in Figure 4.4, represents the 0.5 line. A curve following this line indicates that predictions are no better than random, and curves below this line indicate a negative confidence. To gain a single value evaluation of the ROC curve, the area under the curve is calculated in a manner similar to a classical integral operation. This AUC value falls on the range $[0, 1] \in \mathbb{R}$, with a higher value indicating that lower threshold values give higher true positive rates.

Sensitivity, also referred to as Recall, is the true positive rate of the model. This is useful for evaluating how many of the positive predictions are actually correct. Specificity is the true negative rate of the model, which is useful for evaluating how many negative predictions are correct. In the case of instrument classification, this rate will be much higher than sensitivity, as most frames only include one or two of the instruments at once. Randomly predicting that an instrument is not present is much more likely to be true than predicting that it is present, as there are considerably more frames in which it is not present. If both sensitivity and specificity are near 1.00 for a model, then there are few false positive and false negative predictions.

Tables 4.3, 4.4 and 4.5 show the results of testing each model through 4-Fold Cross Validation. It was expected that the optimal number of frozen layers lay somewhere between 50% and 90%, but surprisingly the results show that a lower number of frozen layers provided higher predictive results. The ResNet-50 models 0, 1, 2 and 3 each

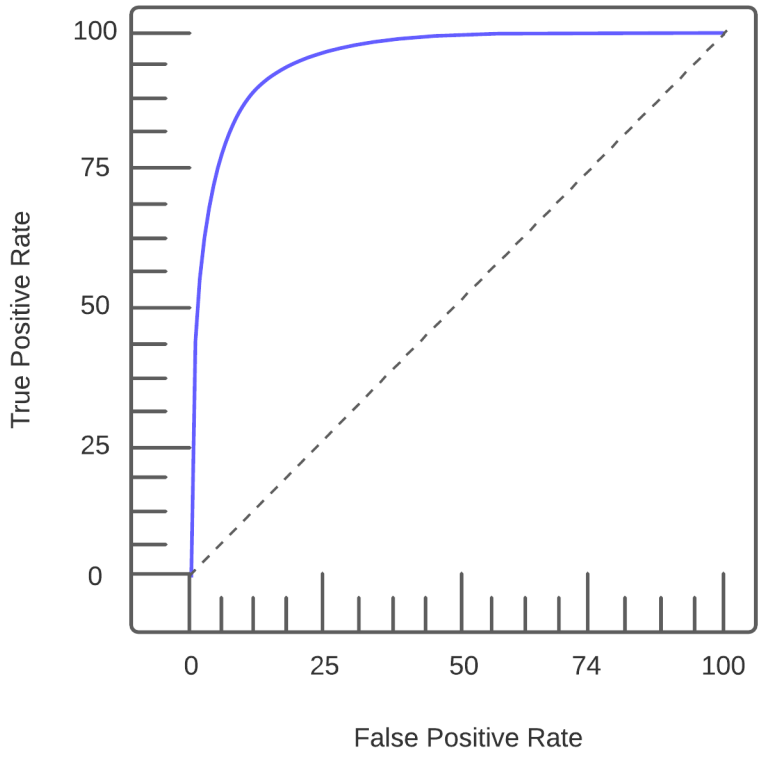

Figure 4.4: Receiver Operating Characteristic (ROC) Curve Example.

gradually descended in mean AUC and sensitivity, likewise with the InceptionV3 models 8, 9, 10 and 11. The ResNet-152 models, however, had model-5 slightly outperforming model-4, and both outperforming the remaining models 6 and 7. This pattern indicates that a lower percentage of frozen layers increases performance. Both model-0, with 0% frozen layers, and model-8, with 0% frozen layers, outperformed their counterparts. Regarding the high model-5 result, which had 50% frozen layers and outperformed its 0% counterpart, it would appear that an overall potential optimal number of frozen layers may lie around 25%. Due to time constraints this percentage value was not investigated further in this experiment.

Table 4.6 shows the results of the top performing models from each of the three architectures. Interestingly, model-0 gains the highest sensitivity and specificity, while model-8 gains the highest mean AUC. Model-8 performs very well on the lower fre-

Metric	Instrument	model-0	model-1	model-2	model-3
Class AUC	Idle (No Instrument)	**0.974**	0.967	0.972	0.950
Class AUC	Forceps	0.979	0.965	**0.982**	0.952
Class AUC	2.4mm Keratome Blade	**0.947**	0.874	0.911	0.910
Class AUC	1.1mm Paracentesis Blade	**0.931**	0.860	0.887	0.735
Class AUC	Rycroft Cannula	**0.950**	0.939	0.916	0.816
Class AUC	Chopper	**0.995**	**0.995**	0.994	0.986
Class AUC	Phacoemulsification Probe	**0.997**	0.994	0.990	0.992
Class AUC	Irrigation/Aspiration Probe	**0.990**	0.988	0.964	0.962
Class AUC	Rhexis Forceps	0.993	**0.994**	0.992	0.975
Class AUC	Lens Cartridge Injector	0.935	**0.969**	0.947	0.932
Mean AUC	All	**0.969**	0.954	0.955	0.921
Sensitivity	All	**0.966**	0.947	0.942	0.934
Specificity	All	**0.9978**	0.9975	0.9973	0.9960

Table 4.3: ResNet-50 predictive models: testing AUC, sensitivity and specificity.

Metric	Instrument Name	model-4	model-5	model-6	model-7
Class AUC	Idle (No Instrument)	0.970	**0.974**	0.971	0.956
Class AUC	Forceps	0.970	**0.980**	0.976	0.970
Class AUC	2.4mm Keratome Blade	0.918	**0.959**	0.950	0.941
Class AUC	1.1mm Paracentesis Blade	**0.931**	0.873	0.907	0.839
Class AUC	Rycroft Cannula	0.914	**0.925**	0.909	0.863
Class AUC	Chopper	0.994	**0.996**	0.991	0.990
Class AUC	Phacoemulsification Probe	**0.997**	**0.997**	0.994	0.992
Class AUC	Irrigation/Aspiration Probe	**0.989**	**0.989**	0.977	0.971
Class AUC	Rhexis Forceps	0.988	**0.990**	**0.990**	0.978
Class AUC	Lens Cartridge Injector	0.924	**0.975**	0.956	0.955
Mean AUC	All	0.960	**0.966**	0.962	0.945
Sensitivity	All	0.940	**0.947**	0.929	0.899
Specificity	All	0.9971	**0.9974**	0.9971	0.9965

Table 4.4: ResNet-152 predictive models: testing AUC, sensitivity and specificity.

quency classes, namely the Forceps, 2.4mm Blade and 1.1mm Blades instruments. These only have several hundred training instances, opposed to the many thousands of the Idle class. For this reason the InceptionV3 architecture has become the preferred deep learning model for the deep learning framework. There are very high AUC scores for all classes, implying that the models are returning a high level of confidence in their predictions. The very high sensitivity and specificity scores also show that the models

Metric	Instrument Name	model-8	model-9	model-10	model-11
Class AUC	Idle (No Instrument)	0.971	**0.972**	0.971	0.951
Class AUC	Forceps	**0.984**	0.979	0.963	0.941
Class AUC	2.4mm Keratome Blade	**0.988**	0.958	0.963	0.891
Class AUC	1.1mm Paracentesis Blade	0.937	**0.967**	0.922	0.855
Class AUC	Rycroft Cannula	0.944	**0.949**	0.929	0.829
Class AUC	Chopper	**0.995**	0.994	0.987	0.991
Class AUC	Phacoemulsification Probe	0.994	**0.995**	0.977	0.993
Class AUC	Irrigation/Aspiration Probe	**0.990**	0.983	0.967	0.950
Class AUC	Rhexis Forceps	0.990	**0.994**	0.975	0.961
Class AUC	Lens Cartridge Injector	**0.963**	0.952	0.939	0.934
Mean AUC	All	**0.976**	0.974	0.959	0.929
Sensitivity	All	**0.959**	0.949	0.923	0.897
Specificity	All	0.9965	0.9959	0.9907	**0.9971**

Table 4.5: InceptionV3 predictive models: testing AUC, sensitivity and specificity.

are very accurate, generating only a few false positives or false negatives.

Metric	Instrument Name	model-0 (ResNet-50)	model-5 (ResNet-152)	model-8 (InceptionV3)
Class AUC	Idle (No Instrument)	**0.974**	**0.974**	0.971
Class AUC	Forceps	0.979	0.980	**0.984**
Class AUC	2.4mm Keratome Blade	0.947	0.960	**0.988**
Class AUC	1.1mm Paracentesis Blade	0.931	0.873	**0.937**
Class AUC	Rycroft Cannula	**0.950**	0.925	0.944
Class AUC	Chopper	0.995	**0.996**	0.995
Class AUC	Phacoemulsification Probe	**0.997**	**0.997**	0.994
Class AUC	Irrigation/Aspiration Probe	**0.990**	0.989	**0.990**
Class AUC	Rhexis Forceps	**0.993**	**0.993**	0.990
Class AUC	Lens Cartridge Injector	0.935	**0.966**	0.963
Mean AUC	All	0.969	0.960	**0.976**
Sensitivity	All	**0.966**	0.947	0.959
Specificity	All	**0.9978**	0.9974	0.9965

Table 4.6: Top predictive models from each architecture: testing AUC, sensitivity and specificity.

While tested on the CATARACT test dataset and so a direct comparison is impossible, the instrument classification models from the challenge paper [32] gained mean AUC scores of 0.9971, 0.9931 and 0.9897. The top mean AUC score performed by our classification models in this experiment is 0.976, which would have placed sixth among

the results of the fourteen teams. Thus, there is high confidence in the reliability, robustness and sheer predictive power of the deep learning models presented in this thesis.

4.4 Further Experimentation

A number of smaller experiments to further improve predictive performance have been conducted in this section. As the models have all been trained on the complete first set of 29 videos, it is impossible to conduct several of the further planned experiments without testing on images used in training. Additionally, the 29 videos were labeled and saved in preprocessed form for training the models, which makes testing the efficacy of the preprocessing method very difficult. To combat these two issues, a second set of videos, described in the previous chapter, were labeled by model-8, hand corrected and saved as raw images with no preprocessing applied. All models used in this thesis are trained on the first set of videos, and this following set of experiments uses the second set of videos for testing.

Each model has four variations, a, b, c and d, each trained on a different subset, as seen in Table 4.2. In practice, the deep learning framework uses these four variations together in an ensemble to make predictions for frames coming through the system. Each variation gives a prediction for a particular frame and the four predictions are combined to generate a final prediction. This combining is done through two alternative methods: averaging the four predictions or choosing the maximum of the four predictions. Alternatively, a single model could be trained on all four subsets, removing the need for an ensemble. The first experiment in this section tests which method gains the highest predictive performance: the averaged ensemble, the max ensemble or the single model. All methods have preprocessing applied.

Table 4.7 shows the results found in this first experiment. The first notable result is the disparity between the accuracy provided by model-8 on the first set of 29 videos versus the second set of 10 videos. Model-8 gained a mean AUC of 0.976 on the first

set of videos, as seen in Table 4.6, but gained a mean AUC of 0.936 on the second set of videos. This is due to the preprocessing method; it appears that the second set of videos are zoomed closer to the eye than the first set, which results in less of the instruments being visible. This theory is reinforced in a later experiment which tests the efficacy of the preprocessing method. Given that all three prediction methods, the average ensemble, maximum ensemble, and single model predictions are presented with the exact same frames, reliable conclusions can still be drawn from the data. Table 4.7 shows that the averaged ensemble gains the highest mean AUC over all instruments, while the maximum ensemble gains the highest sensitivity score. The single model, trained on all the training data, struggles to compete with both ensembles. Again, AUC is prioritised, and the averaged ensemble is considered the better classifier for the deep learning framework.

	Averaged	**Maximum**	**Single**
Mean AUC	**0.936**	0.929	0.917
Sensitivity	0.919	**0.951**	0.874
Specificity	**0.9918**	0.9822	0.9906

Table 4.7: Predictive performance metrics of the model-8 variations ensemble, tested on the second set of 10 videos.

The challenge paper [32] shared five steps to success. The fourth step involved using multiple deep learning architectures together in a single ensemble. To test if this improves the accuracy of the deep learning framework, an ensemble consisting of the models model-0a, model-8b, model-5c and model-8d is tested on the second set of 10 videos. The predictions generated by each model are combined using the averaging method. Thus, each architecture has input into the final prediction value, with the higher performing model-8 effectively having a higher weighting on the outcome. Preprocessing is applied for this test. The results of the mixed ensemble are compared against the results of the model-8 averaged ensemble found in Table 4.7.

Table 4.8 shows the result of this experiment. The mixed ensemble provides a

slightly higher AUC performance than the averaged model-8 ensemble, but has a considerably lower sensitivity and specificity. While the mixed ensemble has a higher AUC, the considerably lower sensitivity and specificity lowers the value of using this ensemble as the primary classifier.

	model-0,8,5,8	**model-8**
Mean AUC	**0.939**	0.936
Sensitivity	0.892	**0.919**
Specificity	0.9882	**0.9918**

Table 4.8: Predictive performance metrics of the mixed ensemble vs the averaged model-8 ensemble, tested on the second set of 10 videos.

The final and most interesting experiment of this section involves evaluating the usefulness of the preprocessing method. Unfortunately, this cannot be conducted on the original set of videos used in training. Additionally, all models are trained using preprocessed frames, and so this test cannot evaluate models trained using non-preprocessed data. To conduct this experiment the averaged model-8 ensemble is tested using non-preprocessed frames from the second set of videos. Like Table 4.8, it is compared against the averaged model-8 ensemble results found in Table 4.7.

Table 4.9 shows the result of this experiment. The non-preprocessed column refers to the model-8 ensemble with no preprocessing applied to the testing frames. The preprocessed r=3x column refers to the model-8 ensemble with the radius of the mask circle set to 3 times the pupil radius size. The preprocessed r=2x column refers to the model-8 ensemble with the radius of the mask circle set to 2 times the pupil radius size; this is the default radius size used throughout the previous experiments. Counter to our expectations, the non-preprocessed method performed considerably better than its preprocessed counterparts. In fact, it achieves higher performance than the best results of model-8 on the first set of videos. This would suggest (but not conclude) that the preprocessing method lowers predictive performance for both the second and first sets of videos. A visual inspection of the preprocessed frames from the second set of videos

shows that the frames appear to be zoomed in at a greater rate than the first set of videos, cutting out a greater proportion of the instruments during the masking phase of the preprocessing method. This is reinforced by the result that the greater radius size of three times the pupil radius leads to performances similar to the performances found by the two times the pupil radius tested on the first video set (seen in Table 4.6). This indicates that the frames from the second set are zoomed in roughly 30% further than the first set.

	model-8 (no preprocessing)	**model-8 (preprocessed r=3x)**	**model-8 (preprocessed r=2x)**
Mean AUC	**0.982**	0.975	0.936
Sensitivity	**0.968**	0.957	0.919
Specificity	**0.9978**	0.9972	0.9918

Table 4.9: Predictive performance metrics of the averaged model-8 ensemble with differing preprocessing methods. Tested on the second set of 10 videos.

To conclude this chapter, the highest performing model configuration is proposed: an ensemble of the InceptionV3 model-8 variants with no preprocessing applied to testing frames. It may be true that training the models on non-preprocessed frames adds further performance, but due to the first set of videos being saved as preprocessed this cannot be evaluated. The InceptionV3 architecture with a low percentage of frozen layers gained a higher mean predictive AUC performance than both ResNet-50 and ResNet-152, notably outperforming when predicting the imbalanced classes. It was found that averaging the predictions of the four variants gave higher AUC performance than choosing the maximum prediction or training a single model on all the training data. An ensemble of the three separate architectures gave a slightly higher AUC performance than the InceptionV3 ensemble, but at the cost of lowered sensitivity and specificity scores. It was also found that the preprocessing method used in this thesis is counterproductive, lowering predictive performance. As seen in Table 4.9, the highest mean testing AUC performance gained within this thesis was 0.982, offering very high confidence in the reliability of the predictive models for the deep learning framework.

Chapter 5

The Deep Learning Framework for Clinical Practice

5.1 Generating Clinical Summary Reports and Instrument Activity Sequence Charts

The fifth step of the Deep Leaning Pipeline, as depicted in Figure 4.1, is the automatic generation of clinical summary reports. These reports include instrument use statistics, video annotations and graphical representations of the surgery timeline. This information helps to set a definitive and objective representation of the surgery, allowing accurate operative skill assessments of surgeons to be drawn. These help to track improvement, reveal areas of weakness and compare surgeon performances.

The deep learning models classify a cataract surgery video and provide a prediction vector. From this vector the Instrument Activity Sequences can be extracted. The prediction vector is time ordered, each frame occurs before the next during the surgery, allowing sequences of positive predictions to be found. As the prediction for each frame has ten probability values, one probability for each instrument, instruments can be concurrently active and sequences often overlap each other. Each sequence tracks a single class through a number of frames. A two second grace window is applied to the

sequences; if a gap between two sequences of the same class is less than two seconds they are combined and the between frames are considered to have positive predictions. This helps to catch false negatives, but loses precision by not recording the case where an instrument pauses for less than two seconds before continuing. There is opportunity for further improvement here, in the opposite case, where a sequence of negative predictions contains lone positive predictions, Instrument Activity Sequences will still be recorded. It is more likely that these are false positives, but currently this case is not addressed. This process is a form of temporal smoothing, where lone negative predictions within a sequence of positive predictions are assumed to be false negatives and are included as a positive prediction. This helps to increase the accuracy of the deep learning models post-prediction, and is the fifth step of the "five steps to success" proposed in [32].

Once the Instrument Activity Sequences are extracted from the prediction vector they can provide a large amount of valuable information about the surgery videos. Table 5.2 shows a list of the Instrument Activity Sequences provided by the deep learning models from a particular cataract surgery video. These sequences are grouped by the instrument tracked by the sequences and are arranged in time order. This allows the reader to see exactly at what time certain instruments were used, how long they were used for, and how many times they were used. This format can also be easily read by a computer program, with potential applications including creating digital video annotations, subtitling videos or comparing surgeries.

Figure 5.1 shows a graphical timeline summary of the Instrument Activity Sequences shown in Table 5.2. This Gnatt-like chart is a graphical representation of the instrument timeline during the cataract surgery. The surgical phases and steps described in the background chapter can be estimated by eye using the instrument activities presented in the chart. For example, in Figure 5.1, the following conclusions can be safely drawn: during the seconds 0 to 30, the Forceps grip, 2.4mm Keratome blade and 1.1mm Paracentesis blades are active, implying the incision phase is being performed; in seconds 30 to 50, the Rycroft Cannula is active, as viscoelastic is introduced to prevent capsular

rupture; in seconds 50 to 80, both the Forceps grip and Rhexis Forceps grip are active, implying the capsulorhexis phase is being performed; in seconds 90 to 120, the Rycroft Cannula is active, implying that hydrodissection is being performed; in seconds 150 to 350, both the Chopper and Phacoemulsification probe are active, implying the nuclear disassembly phase is being performed; in seconds 380 to 490 the Irrigation and Aspiration probes are active, implying that the cortical cleanup phase is being performed; in seconds 510 to 550, the Lens Cartridge Injector is active for a short time, followed by the Irrigation and Aspiration probes, implying that the lens has been inserted and manipulated into place. Finally, the Rycroft Cannula is used in seconds 550 to 560, implying that remaining viscoelastic fluid has been removed and the surgery is completed. So from the combination of instruments being used the phases of the operation can be discerned. Furthermore, it can be seen that the operating surgeon paused briefly during the operation of several instruments, and that the idle time between instrument transitions is recorded. Purely using instrument annotations to predict the timeline of surgical phases and steps, as has been roughly estimated just prior, is generally the same approach used by the top performing deep learning phase classifiers. As phase classification in the literature has not yet reached a satisfactory predictive performance, the report instead provides instrument activity summaries, giving a more detailed and accurate insight into the contents of the operation.

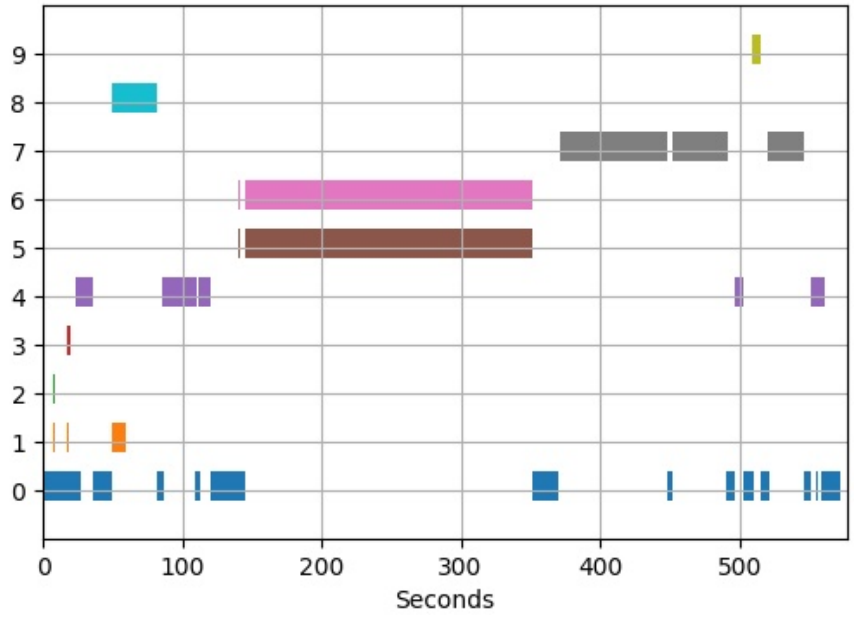

Figure 5.1: Instrument sequence chart representation of the cataract surgery video OS-2021-06-26_140517. Key: **(0)** Idle Class, **(1)** Forceps, **(2)** 2.4mm Keratome Blade, **(3)** 1.1mm Paracentesis Blade, **(4)** Rycroft Cannula, **(5)** Chopper, **(6)** Phacoemulsification Probe, **(7)** Irrigation/Aspiration Probe, **(8)** Rhexis Forceps, **(9)** Lens Cartridge Injector.

Table 5.1 includes further details about the accumulative instrument activity sequences for each instrument. The total number of uses is clearly recorded, alongside the proportion of the video the instrument is active for, and the total duration of time that the instrument is active for. These are found by summing the appropriate instrument activity sequences. The proportion of video values do not have to sum to 100%, as instruments can be active concurrently. Likewise with the total duration time. Of particular interest is the proportion of idle time, providing an indication of how efficiently the operating surgeon performed. These two tables and plot would now, having completed the deep learning framework's process, be combined into a single PDF report and emailed directly to the surgeon.

Instrument	**# of Uses**	**Proportion of Video:**	**Total Duration:**
Idle class	13	23.1%	02:13
.12 Forceps	3	2.00%	00:11
2.4mm Keratome Blade	1	0.26%	00:01
1.1mm Paracentesis Blade	1	0.36%	00:07
Rycroft Cannula	6	9.79%	00:56
Chopper	2	35.7%	3:26
Phacoemulsification Probe	2	35.7%	3:26
Irrigation/Aspiration	3	24.5%	02:21
Rhexis Forceps	1	5.52%	00:31
Lens Cartridge Injector	1	1.13%	00:06

Table 5.1: Instrument activity statistics for the cataract surgery video OS-2021-06-26_140517.

Instrument	Use:	Start Time:	End Time:	Duration:
Idle	Use 1	00:00	00:26	00:26
Idle	Use 2	00:36	00:50	00:13
Idle	Use 3	01:22	01:26	00:04
Idle	Use 4	01:49	01:52	00:03
Idle	Use 5	02:00	02:25	00:24
Idle	Use 6	05:51	06:09	00:18
Idle	Use 7	07:28	07:32	00:03
Idle	Use 8	08:11	08:17	00:05
Idle	Use 9	08:23	08:30	00:07
Idle	Use 10	08:36	08:42	00:06
Idle	Use 11	09:06	09:12	00:05
Idle	Use 12	09:15	09:16	0:01
Idle	Use 13	09:19	09:32	00:13
.12 Forceps	Use 1	00:08	00:09	00:00
.12 Forceps	Use 2	00:17	00:19	00:01
.12 Forceps	Use 3	00:50	01:00	00:09
2.4mm Keratome Blade	Use 1	00:07	00:09	00:01
1.1mm Paracentesis Blade	Use 1	00:17	00:19	00:02
Rycroft Cannula	Use 1	00:24	00:36	00:11
Rycroft Cannula	Use 2	01:26	01:49	00:23
Rycroft Cannula	Use 3	01:52	02:00	00:07
Rycroft Cannula	Use 4	08:17	08:23	00:06
Rycroft Cannula	Use 5	09:12	09:20	00:08
Rycroft Cannula	Use 6	09:32	09:32	00:00
Chopper	Use 1	02:20	02:21	00:00
Chopper	Use 2	02:25	05:51	03:25
Phacoemulsification Probe	Use 1	02:20	02:21	00:00
Phacoemulsification Probe	Use 2	02:25	05:51	03:25
Irrigation/Aspiration	Use 1	06:11	07:28	01:16
Irrigation/Aspiration	Use 2	07:32	08:11	00:39
Irrigation/Aspiration	Use 3	08:41	09:07	00:26
Rhexis Forceps	Use 1	00:50	01:22	00:31
Lens Cartridge Injector	Use 1	08:29	08:36	00:06

Table 5.2: A list of the Instrument Activity Sequences for the cataract surgery video OS-2021-06-26_140517. Time is in mm:ss format, with seconds rounded to nearest integer.

5.2 The Clinical Relevance of the deep learning framework

Accurately evaluating the operative skills of a surgeon is an important task, and a number of objective measurement tools and subjective self-assessments have been proposed. The objective measurement tools include the complication rate for a surgeon, which is indicative of performance. This metric, however, is based upon the proposition that all complications are preventable and may lead to surgeons avoiding high-risk cases. Alternatively, forced ranking can be used, competitively ranking surgeons into the categories: top 20%, middle 70%, or bottom 10%. While useful when applied constructively within a department, it is limited in its specificity for evaluation of operative skills [47]. There are a number of operative self-assessment tools available for cataract surgery in particular, including OASIS, GRASIS and ICO-OSCAR:phaco. These involve scoring the individual steps and aspects of each operation and recording errors and complications. These methods require training to identify errors, but claim to offer both subjective and objective measures of operative performance [48]. The authors of [48] analysed these models and found that they demonstrate coherent and complete portrayals of resident cataract surgical competency. However, the authors of [49] performed a study with 16 junior surgeons and 16 senior surgeons, comparing their ability to accurately self-assess their own operative competency. They found that the junior surgeons were better at assessing their competency with global indices such as tissue handling, iris protection and overall speed, while senior surgeons were better at assessing their competency with task-specific components of the surgery. Thus, self-assessment tools may not be an effective evaluator for junior surgeons, as they are poorer at objectively assessing their own operative competencies.

The authors of the recent paper [50] have researched building a deep learning based tool that objectively assesses intraoperative technical skill in the capsulorhexis phase of cataract surgery. Velocity vectors found by tracking the tips of surgical instruments and

optical flow fields are both used as input features to objectively measure the operative skill of surgeons. Their dataset for this experiment was built by ranking a number of cataract surgery capsulorhexis videos by operative skill along the range of 1-5, with 5 being expert skill. They had an unbalanced dataset, lacking videos scored with 2 or 3, leading to poorer results. They did, however, find an AUC value of 0.863 for predicting skill categories using instrument tip velocities and an AUC value of 0.803 using optical flow fields. The authors state that while this predictive performance is not high enough for operational use, it shows the process is a valid technique for objectively measuring intraoperative performance. This is the first and, to date, only attempt to objectively assess surgeon operative skill in cataract surgeries using deep learning techniques.

It is to this context that the deep learning framework gains its clinical relevance. While not providing a quantifiable measurement of operative skill, it does provide an automatic digital representation of a performed surgery. This allows conclusions and objective assessments to be drawn from detailed statistics, exact instrument activity records and reliable graphical representations of the surgery timeline. These have been previously unavailable, and until now self-assessments have been based upon the surgeon's recollection of the surgery or the time consuming task of reviewing of unedited surgery video recordings. This provides a more objective and definitive representation of the surgery to help draw assessments from.

For example, a surgeon can use the averaged proportions of idle time over several surgeries to assess how much time they are spending transitioning between instruments, or pausing during phases and steps. Minimising this idle time would suggest a more efficient and potentially more experienced surgeon. In the same vein, lowering the average duration of time spent with each instrument would likely indicate an experienced surgeon or a routine case with reduced complexity. Occasionally surgeons are required to repeat phases or steps in a surgery, often due to insufficient skill or surgical complications. The Instrument Sequence Chart helps to make obvious which sequence of instruments were used and when phases were repeated, helping to identify these cases.

Although not fully explored in this thesis, it is possible to use the Instrument Activity Sequences to create a normalised vector representation of the surgery. By combining a number of expertly conducted surgeries in this format, fingerprints that characterise the preferred technique of individual surgeons could be created. The application of distance metrics like Dynamic Time Warping or the Euclidean Distance Formula could be used to measure the deviation of surgeries from these fingerprints, which could then be used to measure the abnormality of individual surgeries.

Among the evaluation applications of this report, there are several additional administrative uses. Representing a typically ten minute long video with a single chart allows abnormal surgeries (where the typical phase and step process is not followed) to be easily located and identified. Additionally, the use of video annotations allows surgeons to quickly navigate video recordings to moments of interest, saving valuable time.

5.3 Distance, Blur and Other Findings

This section discusses several approaches applied to the deep learning framework to gain further insight into the nature of cataract surgery videos. The first of these is the measurement of eye movement throughout the video. This keys into the Hough Circle Transform stage of preprocessing, using the candidate circles and the Euclidean Distance formula to measure the number of pixels between the pupil center points of two consecutive frames. This distance is calculated throughout the set of video frames and an example dot-plot is depicted in Figure 5.2. A patients eye movement can be caused by a number of factors: the patient moving their head, the patient moving their eye, the surgeon moving the eye with an instrument, or the surgeon adjusting the microscope. A proposed application was to use the distance data to predict complications occurring within a surgery. It was hypothesised that if there is a lot of eye movement from the patient, then there is a higher chance of the surgeon making a mistake. To evaluate this a new dataset of numerous surgical videos would have been required, including several

videos that contain instances of complications. Due to time constraints this further experimentation was not included within the scope of this thesis.

From Figure 5.2, it can be observed that the rate of the patients eye movement is gradually declining throughout the surgery. This is likely some measurement of the anxiety of the patient, rather than due to the surgical process itself. The range of values that the distance values fall on is dependant on the video extraction frame rate. These Figures are calculated using a rate of 4 FPS, allowing a large amount of time between each frame. A higher FPS rate would leave less time for eye movement between frames and so the distance values would be scaled down. Figure 5.3 shows a histogram and density function representation of the distance data. It can be observed that there is a right-leaning skew, with the majority of distances close to zero. A more pronounced tail would indicate a larger degree of eye movement throughout the video. Due to these factors, including the dependant FPS rate, a measurement of the skewed density function is recommended to make an appropriate measurement of the degree of eye movement throughout the video.

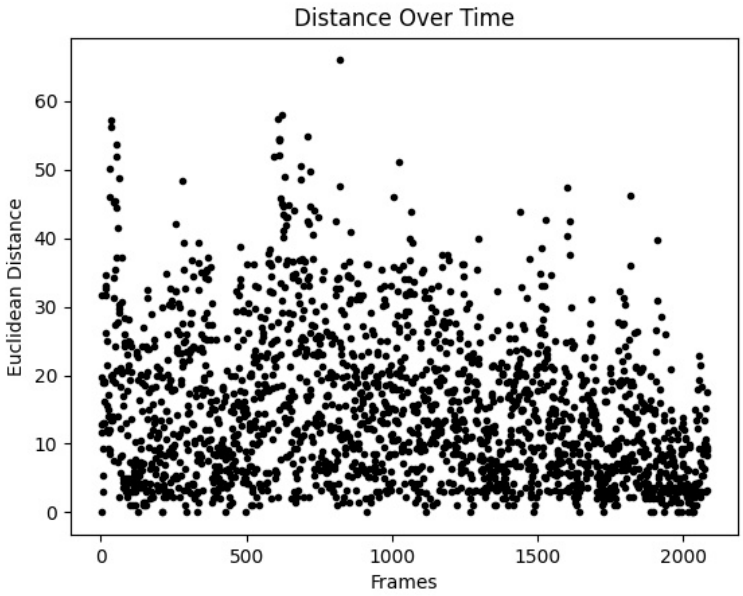

Figure 5.2: The euclidean distance (measured in pixels) moved by the pupil center point between two consecutive frames. Note: calculated at an FPS rate of 4; a higher FPS rate would scale the distance values down, as less time between frames is allowed.

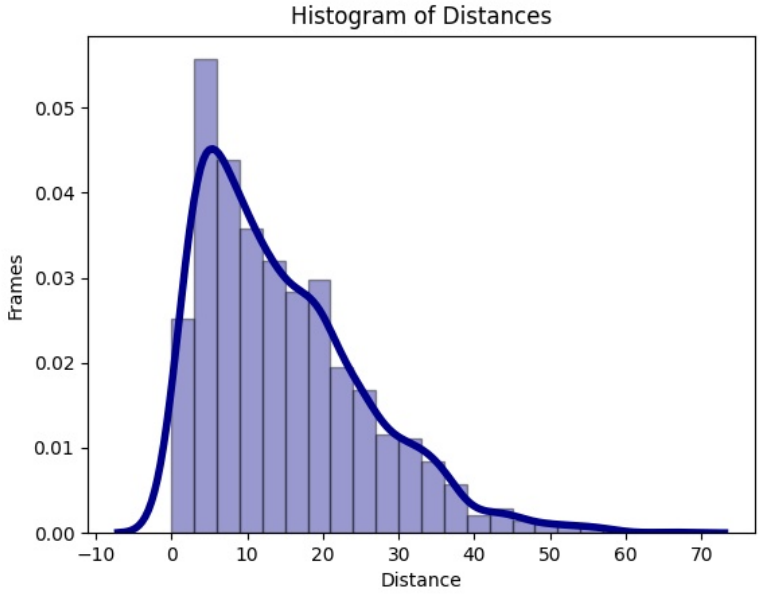

Figure 5.3: A histogram and density function plot of the pixel eye movement distances found within a cataract surgery video.

It should also be noted that these distance values are calculated prior to averaging

the candidate circle locations, unless no circle could be found for a particular frame, in which case the rolling average candidate circle is used. Thus, this process includes the distance between candidate circles which may not be accurate. The benefit of excluding averaged circles is that averaging limits the possible distance between two consecutive frames to less than 30 pixels and hence does not accurately track large eye movements.

The authors of the paper [51] note that a common quality impairment observed in cataract surgery videos is blur, typically caused by object motion or a defocused camera. The authors proceed to successfully use a deconvolutional neural network to remove blur from surgery videos. For the purposes of using deep learning to classify cataract surgery video frames, the impact of blurred frames on the accuracy of the predictive models is evaluated. First, the level of blur for each frame is detected using the variance of the laplacian technique, as described in the article [52]. The laplacian operator computes the second derivative of an input image and steps a kernel matrix over it (similar to a convolutional neural layer). If the variance between the convolution values are high then this indicates there are a large number of edges and non-edges present, representative of an in-focus image. If the variance is very low then there will be a large number of very soft edges present within the frame, typical of a blurry image. For reference, Figure 5.4 depicts several frames of varying blurriness and the calculated blur value for each frame. Counter-intuitively, a lower blur value implies that that there is more blur within that frame, as there is lower laplacian variance.

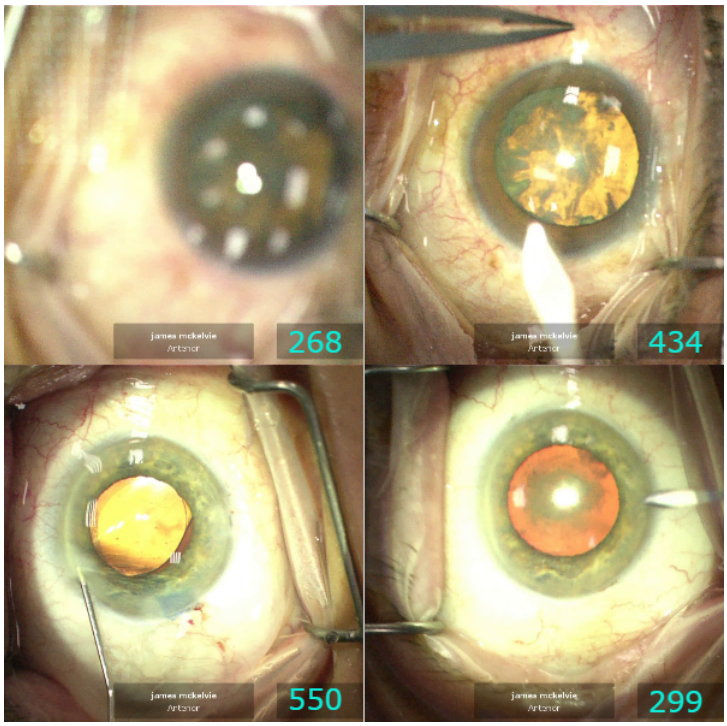

Figure 5.4: Four cropped but non-preprocessed frame samples, with calculated blur values for reference.

Figure 5.5 has been plotted by using the blur detector technique on each frame and plotting these blur values over the course of a particular cataract surgery video. It can be observed that the blur values follow a trend and do not appear to be random. The low point, observed around the 1800 frame mark, aligns with the cortical cleanup phase, where Irrigation and Aspiration probes are moving rapidly over and around the eye. It appears this is a more blurry section of this particular video.

It was noted that often during surgery the instruments would be physically raised above the eye and towards the camera, causing them to go out of focus. To try and identify these frames a segmentation technique was used. Each frame was divided into 4x4 sections and the blur value for each section was detected. An example of this is depicted in Figure 5.6. It was hypothesised that calculating the variance of these blur values may reveal frames where individual sections containing instruments were blurry but the majority of the image was not. However, it was observed that blurry instruments do not have a significant impact on the blurriness of their sections and

many other factors also influence the blurriness of each section. We were unable to reliably determine which of the sections contained blurred instruments (if any).

The blur variance detected for each frame of a cataract surgery video is shown in Figure 5.7. Unfortunately, in practice it appears that while the instruments are blurred in isolated sections, other sections of the image are also often blurred. This leads to the observation that there are likely at least three mechanisms applying blur: certain physical locations of the eye, a raised or quickly manipulated instrument, and some mechanism that applies blur evenly to the entire image. Dividing the image into multiple sections does not appear to help differentiate between these mechanisms.

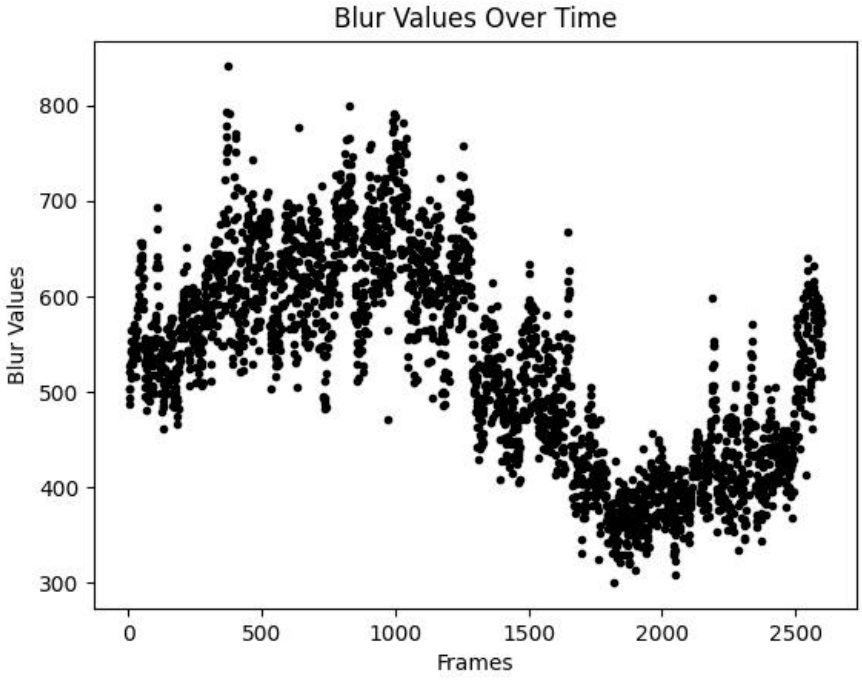

Figure 5.5: Normalised laplacian variable blur values plotted over the video run time. A lower value indicates more blur within a frame.

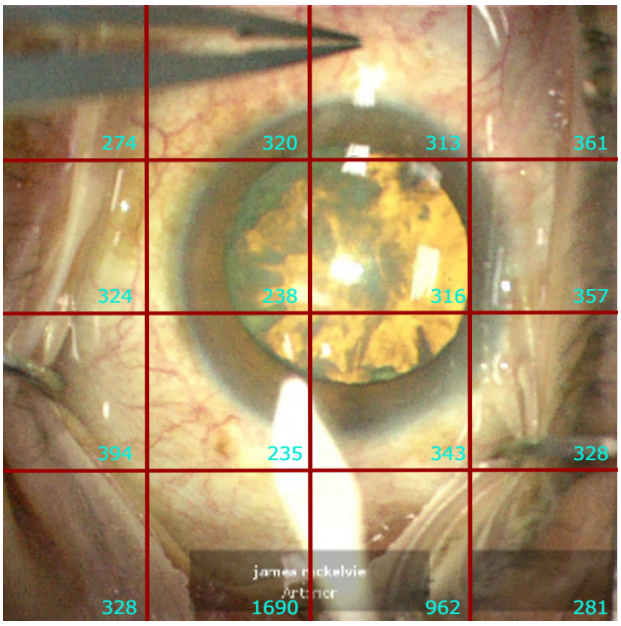

Figure 5.6: A cropped but non-preprocessed frame divided into 4x4 sections, with each section providing a blur value.

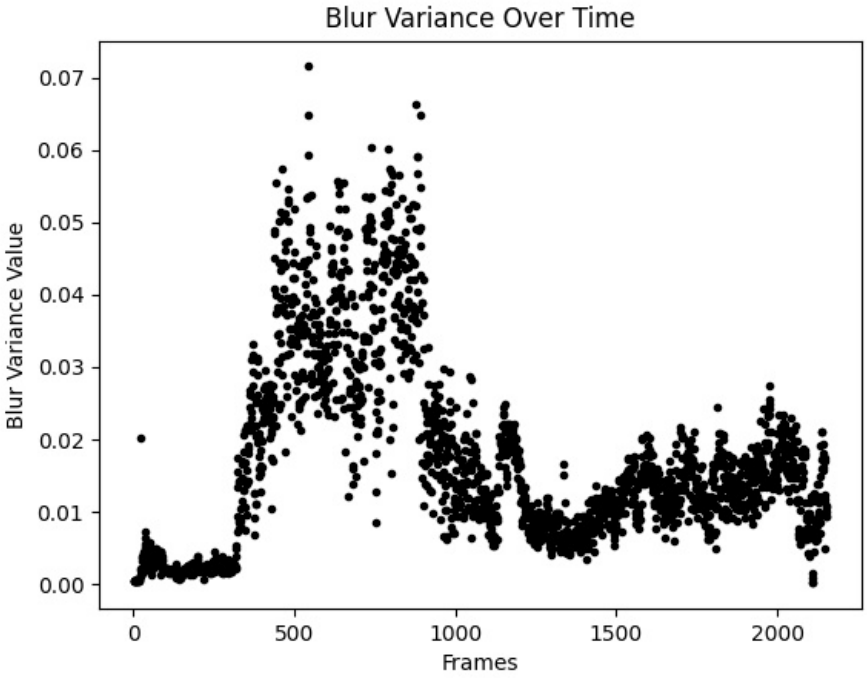

Figure 5.7: Normalised variances, calculated from laplacian blur values found from 16x16 equal sections within each frame. High variance indicates more isolated areas of blur within each frame.

Finally, in Table 5.3, the effect of removing blurry frames from the test data is presented. The model used in this test was the non-preprocessed, averaged model-8 ensemble. The test set consisted of the labeled second set of videos, containing 32,865 frames. If blurry frames are lowering the accuracy of the deep learning models, then excluding frames below a certain blur threshold should increase predictive performance. Four threshold values were used, 0 (no threshold), 400, 500 and 600. Each ascending threshold excluded more frames from the dataset. Interestingly, as seen in the Table, AUC performance dropped as blurry frames were removed, while sensitivity and specificity tended to increase. This would suggest that although the model is making less false predictions with a higher blur threshold, it loses confidence in its predictions. As the model is trained on the first set of videos with a blur threshold of 0, this data suggests that it has learned the features of blurry instruments and uses blur as part of the decision making process. The high sensitivity and specificity would indicate that removing the blurry frames from the training set would increase testing performance, as visual features other than blur would become more relevant to the model. This, however, would reduce the usability of the deep learning framework, as reduced frames lower the accuracy of the generated clinical summary reports. This effect of reduced frame rates is explored in Chapter 6.

	Thresh=0	**Thresh=400**	**Thresh=500**	**Thresh=600**
Percent Excluded	0.00%	17.7%	52.8%	76.5%
Mean AUC	**0.982**	0.981	0.977	0.975
Sensitivity	0.968	0.967	0.973	**0.978**
Specificity	0.9978	0.9982	**0.9989**	**0.9989**

Table 5.3: Predictive performance of the non-preprocessed, averaged model-8 ensemble, tested on the second set of 10 videos excluding frames below certain blur thresholds.

To conclude this chapter a brief summary is presented. Instrument Activity Sequences are extracted from a time ordered prediction vector, each representing a sequence of frames where a single instrument or idle time was used. These are then used to form a definitive representation of the surgery, from which surgeon operative skill

assessments can be recorded more reliably and a number of administrative tasks can be performed more efficiently. A graphical representation of these Instrument Activity Sequences is also generated, providing a simple and clear summary of the surgery timeline. Several further factors were examined, including the potential use of eye movement to predict surgical complications and the measurement of frame blur to increase predictive performance. The clinical relevance of this project is stated, but not evaluated clinically. Future research would include evaluating the deep learning framework for its relevance and effectiveness in clinical practice.

Chapter 6

The Deep Learning Framework for Embedded Systems

6.1 Embedded Systems

The deep learning framework proposed in this thesis offers a processing system for cataract surgery videos, and its predictive performance and clinical relevance has been evaluated in previous chapters. However, practically implementing the system into clinical practice is an entirely new challenge. Currently, there is a research project developing an online cataract surgery exploration system [39]. This system will be hosted online and will use deep learning to divide cataract surgery videos into relevant phases, creating a video library. A similar approach could be used for the deep learning framework, where uploaded surgery videos could be processed automatically through cloud hosted services. There are a few disadvantages to using cloud services to outsource deep learning, however. Surgeons typically do not have the available time to perform additional tasks such as uploading surgery videos to be processed. Automating the uploading of surgery videos may be unreliable, as there are several points of failure and dependence on other systems. There are also privacy and security issues with this approach [53], and patient permission is required to upload confidential files to the cloud. Additionally, the surgery videos are quite large, around 1 to 2 gigabytes,

which may lead to some upload delays. This approach be a viable option if built into an existing system such as the exploration system proposed in [39], but an alternative lower scale solution is considered within this chapter.

The term "embedded systems" refers to computer systems that are (typically) used within physical applications like digital watches, home appliances, vehicles or robotics. These are usually self-contained computer chips with direct control over the physical or electronic aspects of the device. Due to being only implemented for one or two specific computing tasks, they are designed to be minimal and lightweight. With the success of machine learning, several standalone embedded systems have been developed with Graphical Processing Units (GPUs) installed, allowing developers to run modern deep learning networks. Nvidia produces the world leading Jetson series of advanced AI embedded systems, with the first model, the Nvidia Jetson TK1, being released in 2014. One of the products in this series, the Jetson Nano, is used in this chapter to evaluate the practicality of running the deep learning framework directly in the clinic, in real-time during cataract surgery operations. This would bring simplicity of design, immediate clinical results and reduced security and privacy concerns, as all processing is done on the device.

The proposed solution involves directly connecting the Jetson Nano to the live microscope camera feed, allowing the live processing of the operation using the deep learning framework. Once the operation is complete, and the Jetson Nano has completed processing the video, the clinical summary report would be generated and securely emailed directly to the surgeon. The only required interaction between the Nano and the surgeon would be the initial starting of the process, all other processes would be fully automated. To evaluate that this proposed solution is viable, the framework must be demonstrated to operate on the Jetson Nano architecture, complete in or near real-time, and present accurate and representative clinical summary reports. As the Jetson Nano is lightweight and lacks computing resources, an optimised and reduced version of the deep learning framework is developed, requiring this performance validation.

6.2 Configuring the Jetson Nano

The Jetson Nano 4GB, used for evaluation, was released in 2019 for developers. The development pack, JetPack, was initially released in 2014 alongside the original TK1 model. This pack includes the linux operating system and contains many of the drivers and programs typically required for deep learning applications. The Nano 4GB has only 4 gigabytes of 64-bit LPDDR4 RAM (working memory), which is shared between the 128-core Nvidia Maxwell architecture-based GPU and Quad-core ARM64 A57 Central Processing Unit (CPU). One immediate issue that arises is the use of the ARM64 CPU architecture. Only software programs built for that particular architecture can be installed properly. The majority of development environments, like Windows and most Linux versions, use x86-64 CPU architectures. While not being a barrier, it does add an extra complication, as some versions of Python libraries and resources are not available for the ARM64 architecture.

In the configuration used in this thesis, the Nano is powered via a 5W, 2.4A micro USB cable and, rather than installing a wireless adapter, connects to the internet via an Ethernet cable. To run deep learning models and to utilise the GPU several programs and drivers are required. These are TensorFlow, CUDA, cuDNN and Python. These four programs must appropriately match versions; this is where most of the difficulty in building the environment lies. A section of the TensorFlow documentation, [54], contains a table of the four programs and their compatible versions. The latest version of JetPack to date, v4.6, claims to support Jetson Nano, but does not. JetPack v4.6 includes CUDA v10.2, while the lowest version of the ARM64 based-architecture TensorFlow version it supports is v2.4.0, which requires CUDA v11.0. Instead, an older version of JetPack is used, JetPack v4.4, which includes CUDA v10.2 and TensorFlow v2.2.0. These two are compatible, allowing the installation of cuDNN v7.6 and Python v3.6 to create the full set of dependent programs. Several other required Python packages are installed via an ARM64 version of Pip at their latest versions, including OpenCV, MoviePy, NumPy, MatPlotLib and SciPy. Thereafter, the Jetson Nano 4GB

environment has been configured and is set to run the deep learning framework.

6.3 Optimising the Framework to Run Batch-like

Embedded systems are resource-constrained environments, where they are required to consume very low power and tend to have low computational power and low working memory. These constraints require heavy optimisation of both the chip architectures and the deep learning models [55]. In the case of the Jetson Nano 4GB used in this thesis, it only contains 4 gigabytes of working memory, shared by both CPU and GPU. This memory is required to contain the operating system programs, the input video and the predictive models. The operating system uses 1.1 gigabytes of memory, leaving 2.9 gigabytes of memory for the deep learning framework. Each InceptionV3 based model uses roughly 450 megabytes of memory, and so an ensemble using four models would use roughly 2 gigabytes of memory, leaving 900 megabytes for the video input. The purpose of this evaluation is to show that the framework is accurate and can process in real-time on the Jetson Nano, as such, the video input is expected to be a live video stream. To simulate this, a full video is processed from start to finish as if it were a stream. Unfortunately MoviePy, the Python library used to read video files, operates by reading the entire video into memory before allowing individual frames to be accessed. This process requires roughly 1.3 to 1.5 gigabytes of memory per video, leaving only 500-700 megabytes of memory for the predictive models and framework to operate within. In practice, frames would enter memory one at a time from a video stream, be processed, and then be removed, using only a fraction of the memory of a full video. This choice of using saved videos over actual video streams does not invalidate the experiments in this chapter, as successfully processing a full video within the memory constraints would imply that the easier task of processing a video stream could also be accomplished. In hindsight, the videos could have been reduced to a 299x299 resolution prior to processing, which would have saved valuable memory space.

However, the fact that there is only enough memory to operate one model at a time

dampens the predictive performance. In a previous chapter it was shown via Table 4.7 that a single InceptionV3 model achieved a mean AUC of 0.917 while the averaged ensemble achieved a mean AUC of 0.936. The ensemble version of the program has a higher predictive performance than the single model. To operate the deep learning framework within these tight memory constraints, in real-time, considerable optimisation is required. Although it was also shown in Table 4.9 that preprocessing frames also reduced predictive performance, the optimised variation of the deep learning framework includes processes to preprocess input frames.

In the original pipeline all video frames are preprocessed together and the predictions for every frame are calculated in one operation. As the goal for the optimised variation is to complete processing in real-time, at the end of the actual surgery, each frame must be preprocessed and have predictions calculated individually, in sequential order. Figure 6.1 shows a diagram of the optimised deep learning framework. Starting from the video input step, frames are extracted into batches of a set number; memory constraints limited the amount of frames able to be extracted at once to 50. This batch of frames is then divided and preprocessed using four CPU threads, which utilises the full computing power of the four cores within the CPU. These threads, once completed, send the now preprocessed batch of frames to the GPU where the InceptionV3 model calculates instrument predictions. These predictions are saved in memory and the batch is discarded, clearing memory space for the next batch of frames. This process continues until all frames have been processed. The predictions are then used to find Instrument Activity Sequences and the clinical summary report is generated.

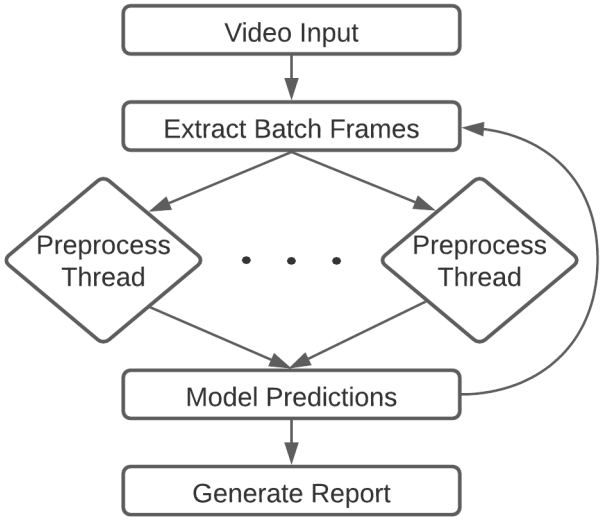

Figure 6.1: Optimised batch-like variation of the deep learning framework.

A complication of this process is the fact that thread processes run in parallel, and do not necessarily conclude in the same order that they were begun. Frames are required to be passed through the framework in sequential time order, and to maintain this order a system using ID records is implemented. This system is also used to manage the largely solved issue of frames being lost during preprocessing. Frame IDs were managed by creating an ascending list of consecutive integers. Each time a frame was processed correctly it's frame number ID was appended to the list. Thus, if a frame was dropped during preprocessing, the consecutive nature of the list would be broken and the number of frames missing between correctly processed frames could be determined. This use of IDs ensures that every frame is accounted for and is in its proper place. Removing the requirement of preprocessing frames makes both IDs and threads unnecessary, greatly simplifying the process.

6.4 Run Time and Accuracy

After the Jetson Nano software environment is properly configured and the deep learning framework is optimised to run within the memory restraints, the evaluation of the

system for use in clinical practice can be performed. There are three primary aspects that must be shown: that the system can be demonstrated to operate correctly on the Jetson Nano, that its run time is less than or equal to the average surgery video length, and that it continues to provide accurate and representative reports when run in the resource constrained environment. The first of these requirements is not straightforward; the Jetson Nano operates on a different computing architecture to the original development environment, which restricts the use of certain libraries and dependent software versions. Additionally, the Nano has tight memory restraints, which requires further engineering to the deep learning framework pipeline to optimise it. The second requirement ensures that the framework can operate for both offline videos and online video streams. The maximum frame rate that allows for real-time processing must be found. This section contains a series of experiments to verify the deep learning framework meets this criteria.

A demonstration of the operability of the deep learning framework on the Jetson Nano 4GB is provided through the presentation of a clinical summary report. This report was generated by running the optimised variation of the Deep Learning Model on the Jetson Nano using the single InceptionV3 model. Frames are extracted at 6 FPS and are not preprocessed. Figure 6.2 shows an Instrument Sequence Chart generated for a randomly selected surgery video from the second set of 10 videos. All instruments are present and follow the pattern of instrument activity physically performed within the cataract surgery video. Table 6.1 shows the finer details of the surgery, also included in the report. As accurate Instrument Activity Sequences are required to generate these two results it is demonstrated that the deep learning framework can operate on the resource constrained environment of the Jetson Nano 4GB.

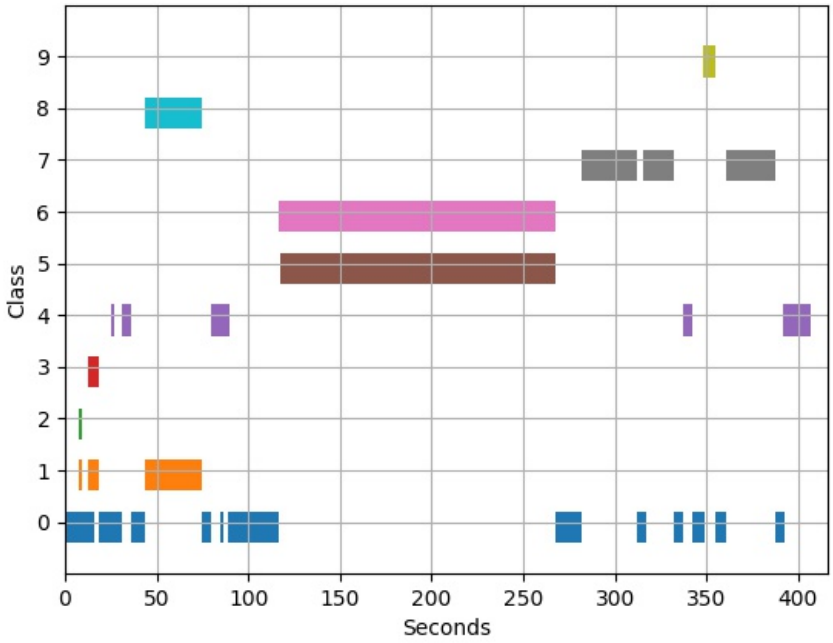

Figure 6.2: Instrument sequence chart generated using the optimised variation of the deep learning framework on the Jetson Nano 4GB, with a real time capable model configuration. Key: **(0)** Idle Class, **(1)** Forceps, **(2)** 2.4mm Keratome Blade, **(3)** 1.1mm Paracentesis Blade, **(4)** Rycroft Cannula, **(5)** Chopper, **(6)** Phacoemulsification Probe, **(7)** Irrigation/Aspiration Probe, **(8)** Rhexis Forceps, **(9)** Lens Cartridge Injector.

Instrument	**# of Uses**	**Proportion of Video:**	**Total Duration:**
Idle class	14	27.1%	01:52
.12 Forceps	3	9.19%	00:38
2.4mm Keratome Blade	1	0.30%	00:01
1.1mm Paracentesis Blade	2	1.38%	00:05
Rycroft Cannula	5	8.69%	00:36
Chopper	1	36.0%	2:29
Phacoemulsification Probe	1	36.1%	2:30
Irrigation/Aspiration	4	17.6%	01:13
Rhexis Forceps	1	7.46%	00:31
Lens Cartridge Injector	1	1.57%	00:06

Table 6.1: Instrument activity statistics for a particular cataract surgery video.

The second aspect of the optimised variation of the deep learning framework to be evaluated is the run time of the program. Ideally, processing a surgical video in a

shorter time than its run length is preferred. This would allow the live operation of the framework during a surgery. To achieve this, on the resource constrained Jetson Nano, three factors are addressed. The first factor is the choice of predictive model. As only a single model can be loaded under the RAM constraints of the Nano, the choices are limited to the single model variations of the InceptionV3, ResNet-152 and ResNet-50 based architectures (model-#e notation). In a number of previous experiments an ensemble of the four variations of each model were used, but this is not possible on the Nano. Of these three architectures, ResNet-152 is not considered, as it is requires more RAM than ResNet-50 but was found to have lower predictive performance. The second factor is the preprocessing of frames. Preprocessed frames were shown in section 4.4 to result in lower predictive performance and also require additional processing time. The third factor is the FPS rate; this is the rate at which frames are extracted from the videos. It is expected that a higher FPS rate will provide more frames, giving imbalanced classes a higher chance of being positively predicted, and also provide finer instrument activity information within the video. However, a higher FPS rate will also require more processing, as the number of frames will increase. Four FPS rates are addressed, 25, 12, 6 and 1. All available surgery videos in the combined set of 39 videos have an initial FPS of 25.

To evaluate these three factors the average time required to process a batch of 50 frames is measured. This was done by recording the time taken to process each batch within a single video and calculating the average time. The same video was used for all four model configurations. The first batch time tended to be much longer than the following batches, as the predictive model is loaded during processing the first batch of frames. Additionally, the last batch is likely to be shorter than the previous batches, as it may not contain the full 50 frames. As all four model configuration's batch times were calculated using the same video, these two outlier batches should not interfere with the comparison between configurations. Once the average batch time was calculated for each configuration, the average number of frames, at each FPS rate, is calculated

across all 39 available surgery videos. Thus, the estimated time to process the averaged number of frames was calculated for each configuration, at each FPS rate. As a side note, extracting Instrument Activity Sequences and generating the clinical summary report requires an additional ten seconds of processing time.

The results of this evaluation are shown in Table 6.2. ResNet-50 tends to process frames faster than InceptionV3, and not preprocessing frames is faster than preprocessing frames by roughly a factor of 2. Additionally, the lower FPS rates required less processing time. These are not surprising results, as ResNet-50 has considerably less layers and variables than InceptionV3, and it is clear that not preprocessing frames and having fewer frames to process is more time efficient. The average video duration is **10.79 minutes**, and therefore several configurations, notably the InceptionV3 model at 6 FPS with no preprocessing, have been shown to run in real time. The third and final evaluation of the optimised variation of the deep learning framework remains: can these real time configurations provide accurate and representative clinical summary reports?

	InceptionV3 Preprocessing	**ResNet-50 Preprocessing**	**InceptionV3 NoPreprocessing**	**ResNet-50 NoPreprocessing**
Avg. Batch Time	11.26s	8.54s	5.40s	4.32s
25 FPS Run Time	60.78m	46.10m	29.15m	23.32m
12 FPS Run Time	29.17m	22.13m	13.99m	11.19m
6 FPS Run Time	14.59m	11.06m	**7.00m**	**5.56m**
1 FPS Run Time	**2.43m**	**1.84m**	**1.17m**	**0.93m**

Table 6.2: Average batch time refers to the average number of seconds required to process each batch of 50 frames. FPS Run Time refers to the estimated number of minutes required to process the average number of frames per video at descending FPS rates. Bold text indicates times that are equal to or shorter than the average video duration.

Not preprocessing frames has been shown to result in higher predictive performance, and ResNet-50 has been shown to run in less time than InceptionV3. As such, the predictive performances of the two single variations are compared in Table 6.3. These two models have been previously compared, with results shown in Table 4.6. However,

differing from the previous evaluation, now the models are evaluated on the second dataset, without preprocessing frames and by using single models trained on all four subsets (the model-#e variations). The single InceptionV3 model gains a higher mean AUC score and also gains a higher sensitivity score. The difference in predictive performance between these two models is, however, slight. Both models do have substantial mean AUC and sensitivity scores, indicating they make accurate and reliable predictions.

	InceptionV3 No Preprocessing	**ResNet-50 No Preprocessing**
Mean AUC	**0.968**	0.963
Sensitivity	**0.926**	0.914
Specificity	0.9954	**0.9967**

Table 6.3: Comparison of the predictive performances of the non-ensemble (model-#e) versions of InceptionV3 and ResNet-50 architectures trained on the complete first dataset and evaluated on the second set of 10 videos with no preprocessing applied to frames.

One of the potential issues that arises when extracting a low number of frames from the video is that small, imbalanced instrument classes may be skipped over. For example, the average number of frames belonging to class 2 in the second set of 10 videos is 59.5 frames per video at 25 FPS, or roughly two and a half seconds worth of video. At an extraction rate of 6 FPS this is reduced to 14.3 frames per video, and at 1 FPS this is reduced to 2.4 frames per video. At 1 FPS, the probability that none of the frames belonging to class 2 will be predicted as a true positive by the InceptionV3 model in Table 6.3 is 0.0019 ($\approx 0.2\%$ of all videos). At a frame extraction rate of 6 FPS the probability of not finding any true positives for class 2 is reduced to $6.7e^{-15}\%$. Only one positive frame is required to record an Instrument Activity Sequence. Thus, there is a slight disadvantage to using an extraction rate of 1 FPS. Furthermore, 1 FPS reduces the precision of Instrument Activity Sequences start and end times to individual seconds. Currently, however, the report does only present video annotations

to the second rather than the millisecond, leading to the case where some sequence durations are rounded down to "0 seconds" long. This can be observed in Table 5.2.

There is a notable difference between model predictive performance and the accuracy of the clinical summary report. The accuracy and precision of the report depends largely upon the predictive model performance, but can also be affected by the FPS extraction rate (as discussed) or grace window errors occurring while generating Instrument Activity Sequences. Errors involving the grace window are most noticeable with the idle class, as instrument sequences shorter than the grace window length will not interrupt an idle sequence. This problem could be solved by determining variable grace window lengths for each individual class. There are no easily accessible methods available to quantify the accuracy of the clinical summary report. It is proposed, however, that the optimised variant of the deep learning framework operating in real time on the Jetson Nano 4GB produces accurate and representative clinical summary reports. This conclusion is drawn by comparing the Instrument Activity Sequences presented in the clinical summary reports with the cataract surgery procedure outlined in the background chapter, and is also reinforced by the substantial predictive performance of the models.

Thus, the deep learning framework has satisfied the three evaluation requirements set out to demonstrate its ability to operate on the resource constrained environment of the Jetson Nano 4GB. It has been demonstrated to operate properly and completely on the Nano, a number of model configurations have been shown to process videos in real time, and it has been shown to provide accurate and representative clinical summary reports using these real time model configurations. This experiment verifies that using physical embedded systems connected to a microscope's live video stream is a viable option for processing cataract surgeries using the deep learning framework.

Chapter 7

Discussion

7.1 Results and Conclusions

A number of research questions were proposed at the beginning of this thesis. Can deep learning provide high predictive performance for detecting various instruments when they appear in individual frames of cataract eye surgeries? Can these predictions be combined over the length of the video to produce relevant clinical summary reports that represent cataract surgery videos? Can that framework be accurately operated on embedded systems in real time? The experiments conducted throughout the course of this thesis have been designed to answer these questions, and have found a unanimous "yes". First, a novel dataset was sampled from 39 cataract surgeries and labeled for evaluation purposes. The deep learning framework was then proposed and a number of deep learning models were trained and evaluated. Two applications of this framework were then further explored: the clinical relevance of the framework and the verification of its operability on the resource constrained environment of embedded systems.

A brief summary of the results of these experiments follows; a novel dataset, drawn from 39 cataract surgery videos, was labeled by instrument and idle state into 10 classes, providing 90,287 individually labeled frames extracted at a rate of 6 FPS. This was divided into two primary datasets, the first initial training set of 29 videos and a second prospective test set of 10 videos. The deep learning framework was

then developed, relying substantially upon deep learning models for its accuracy and performance. These deep learning models were trained and evaluated upon the first dataset using 4-fold cross validation. The maximum mean predictive AUC scored by these models on the first dataset was 0.976, with a maximum sensitivity of 0.966. Further experimentation revealed that evaluating an ensemble of InceptionV3 models, pretrained on ImageNet and further trained with no frozen layers on the first dataset, without applying preprocessing to frames, provided a mean predictive AUC of 0.982 and a sensitivity of 0.968 on the second dataset. This is the highest predictive performance found by any experiment in this thesis.

Several experiments involving the tracking of eye movement and blurry frames throughout a surgery were also conducted. Eye movement plots were generated, and a trend was able to be distinguished. The significance of this result was not further explored, but could be in future research. Blur was measured through a laplacian variance method, but it was found that many individual factors contributed to the blur of frames. Differentiating these factors by segmenting the image into sections was attempted, but held no substantial results. These were both significant topics identified throughout the course of this research, but do not contribute to the original research questions. It's likely that further research of these two subjects would be greatly beneficial.

It was shown that the framework was able to generate Instrument Activity Sequences from predictions made upon sequences of video frames. These were used to create visual charts of the instrument activity timelines and to provide statistics that offered deeper video analysis, including the percentage of time that an instrument was active for within a video, and the video durations of instrument usages. It was demonstrated that an optimised variation of the deep learning framework could be operated on a Jetson Nano 4GB. Several model configurations were calculated to run within real time on the embedded system, including an InceptionV3 model, which had non-preprocessed frames extracted at 6 FPS and was calculated to take 7.00 minutes to process a video of the average length of 10.79 minutes. It was determined that this model configura-

tion, and several others operating within the run time constraints, produced accurate and representative clinical summary reports. This proposal is warranted due to the single InceptionV3 model gaining the high mean predictive AUC score of 0.968 and due to the Instrument Activity Sequences being observed to closely follow the outlined standardised cataract surgery procedure.

In regards to the first research question, the deep learning models have provided high predictive performance, showing that the surgery videos can be classified with a high degree of confidence. The 10 instrument classes each gained high predictive accuracy. This is strong evidence that suggests the clinical summary reports are accurate and representative of the surgery videos. Conclusions regarding the second research question can also be drawn, as the report has been shown to offer a number of objective statistics for surgeon operative evaluations and instrument activity timeline visualisations for a number of administrative tasks, quality improvement and surgical training. The third research question has also been concluded, with several model configurations providing accurate and real time clinical summary reports when run on the Jetson Nano environment. These results show that surgical videos can be processed in a clinical setting, providing a practical deployment of the new technology.

Producing the deep learning framework for production still offers several further challenges. A Jetson Nano 4GB can run the framework satisfactorily, but other cheaper and lightweight alternatives may lower costs in production, or a more powerful system may allow for faster processing times, and therefore higher frame rates and larger models could be used. Additionally, the interactions between system and surgeon needs to be minimised for simplicity and ease of use. It may not be practical to have a full interface system setup in the clinic, including keyboard, mouse and monitor. The system could easily detect the start and end of surgeries, by recording the number of consecutive idle frames. A period of 1 minute of idle time typically indicates that a surgery has finished, and if followed by an instrument may indicate that a surgery has begun. This would allow the system to analyse the microscope live stream constantly throughout

a series of surgeries, generating and automatically emailing reports to surgeons when each surgery concludes. Often a large amount of operations will occur in a single day, so this could be an efficient solution.

A further development may involve the automation of distributing reports to surgeons. The design of this feature would require careful consideration to maintain compliance with the New Zealand Health Information Privacy Code. Python offers several resources to allow the creation of a PDF report containing the clinical summary report findings, and the ability to distribute this report to surgeons. This is a key step towards automating the generation of clinical summary reports for cataract surgery videos.

The majority of research in this area is aimed towards instrument and phase classification. The primary findings of this thesis contribute to the applications of these classification models. The paper [39] has proposed an online system for using phase classification models to create a digital library of cataract surgery videos, broken into five relevant phases. This would allow surgeons to navigate their own surgery videos efficiently, while allowing the retrieval of phase samples extracted from other videos. In a similar fashion our research divides surgery videos into instrument activity sequences (rather than surgical phases), but in the process also gathers quantifiable and visually representative information. This allows for deeper analysis of cataract surgery videos, and could be implemented within the online system for automatic report generation. Instrument classification also provides a more precise representation of the surgery than phase classification, and has been shown within the literature to be more accurate. Translating instrument annotations into surgical phases could be achieved through a set of rules. For example, a sequence of the Phacoemulsification probe and a concurrent sequence of the Chopper instrument would indicate that the nuclear disassembly phase is taking place. This may not be very accurate, as some different surgical steps reuse the exact same instruments, but is a possibility.

The datasets currently available to the public are limited to the CATARACTS dataset [32] and the Cataract-101 dataset [43]. The dataset created to evaluate the

Deep Learning Model consists of 39 videos and 12 instrument classes (including idle class). Two classes were discarded to evaluate the framework, leaving 10 classes to classify, but they are maintained within the dataset. This dataset, if released publicly, would contribute to training and evaluating other researcher's classifier models and frameworks. It is suspected that the lack of research in this area is largely due to the lack of publicly available datasets, and so the addition of a novel dataset may increase interest in classifying cataract surgery videos.

7.2 Limitations

The dataset created and labeled early in this thesis contains several limitations. One of these is the lack of variety. All videos were performed by a single surgeon, and rarely differ in technique, instruments used or duration. Several videos do have additional steps, however, like the insertion of Malyugan rings or Toric axis markers. Several videos also start late, skipping the initial surgical steps. The CATARACTS dataset contains 21 individual instruments, showing that other clinics may use a larger array of instruments. Additionally, all videos were recorded using the same camera model and the same pixel resolution. These factors make it difficult to predict how well the deep learning models developed in this thesis generalise to surgeries performed in other clinics, performed by other surgeons or recorded by other camera models. Further testing is needed to evaluate the generalisation of the dataset and trained models.

Furthermore, the models were trained using only preprocessed frames, but experiments showed that they gain higher predictive performance on non-preprocessed frames. It is likely that models trained on non-preprocessed frames will gain an even higher predictive performance, but this is not able to be evaluated. The initial assumption was that preprocessing would be beneficial to predictive performance, and so the first set of 29 labeled videos in our dataset are saved in preprocessed form. This limits the usefulness of the preprocessed dataset for continued research. As this was a prototype, these insights could be used to create an improved second version in the event the project

continues.

As the optimised variation of the deep learning framework was evaluated using a video recording, rather than a live video feed, there were limited RAM resources. These constraints only allowed a single deep learning model to be utilised. It was shown in Chapter 4 that an ensemble of predictive models provided higher predictive performance, but this was unable to be evaluated on the Jetson Nano, as the offline videos required most of the memory resources. Hence, the Jetson Nano may provide higher levels of surgery representation with a much less memory intensive live video stream. Alternatively, other embedded systems containing up to four times more memory would also suffice, offering redundancy.

7.3 Implementation and Technical Difficulties

A number of substantial difficulties were encountered during the process of developing the deep learning framework. For future reference for researchers these are mentioned in this section. It was noted late into the project that the first and second sets of videos had a disproportionate amount of frames. The first dataset, with 29 videos, had only double the number of frames of the second dataset, which had 10 videos. Initially this was thought to be due to the preprocessing method dropping a much larger number of frames than previously thought. Much testing and debugging was conducted to discover potential errors within the preprocessing method, and an error was located. The preprocessing method was incorrectly set to drop frames when no pupil candidate circles were located by the Hough Circle Transform algorithm. It was found, however, that this bug occurred rarely, and was not responsible for the disproportionate number of frames. After much further analysis, it was found that OpenCV does not return duplicate frames compressed by the MP4 video format. The second set of video frames had been extracted via MoviePy, which did not have this issue. It was calculated that the videos consisted of roughly 40% duplicate frames, which explained the disparity between the two datasets. As duplicate frames hold little to no effect on the deep

learning models, this was not a significant issue for the training or evaluation results, but does lead to timing inconsistencies when analysing the results and generating the reports.

Configuring the Jetson Nano was very difficult, and a number of strategies were attempted. Firstly, following the Nvidia documentation, JetPack v4.6 was installed and the supporting four programs, TensorFlow, CUDA, cuDNN and Python, were installed at the recommended versions. This failed, as the TensorFlow and CUDA versions provided by JetPack v4.6 are incompatible. The next attempt was to build the environment from source, downloading necessary packages by hand. This also failed, as the Nano uses the ARM64 architecture, and the required libraries were unable to be located. Next, Anaconda was used to attempt to build a virtual environment. This also failed, likely due to having to use the JetPack v4.6 ARM64 CUDA version alongside the x86-64 Tensorflow build provided by Anaconda. Next, the older JetPack v4.5 version was used, which suffered the same incompatibility problem as v4.6. Finally, JetPack v4.4 was used, and a working deep learning ready environment was able to be built. Considerable effort, time and research was conducted to correctly configure the Jetson Nano.

Engineering the optimised variation of the deep learning framework also provided considerable difficulty. As described in Chapter 6, IDs were managed by creating an ascending list of consecutive integers. Each time a frame was processed correctly it's frame number ID was appended to the list. Thus, if a frame was dropped during preprocessing, the consecutive nature of the list would be broken and the number of frames missing between correctly processed frames could be determined. As IDs are used to keep track of frames that are dropped during preprocessing, this mechanic had to be engineered to work within the optimised variation of the deep learning framework. This ID system was required to be extended to processing multiple batches of frames and to these batches being divided and preprocessed via parallel multi-threads, without losing the consistency of the entire list of consecutive integers. A very meticulous ID

system was hence created, and much difficulty was encountered in ensuring counters and indexes were correct and that ID consistency was maintained.

7.4 Future Research Directions

The research conducted in this thesis opens up several avenues of potential applications and further research. This thesis has provided a working prototype of an embedded system running the deep learning framework. To expand upon this, a clinical trial of the practicality of running the framework in real-time in a clinical setting and the usefulness of the clinical summary reports for surgeons could be performed. This would help add direction, applicability and feedback for future development. A number of additional real-time features could be explored, including live annotations for observers, complication detection or detecting unusual instrument use patterns.

The deep learning models evaluated in this thesis demonstrate that high instrument predictive performance can be achieved on our dataset. To further evaluate the deep learning framework, these models should also be evaluated on a wider range of cataract surgery videos, to test how well they generalise. Videos from other clinics, other operating surgeons and other camera models should be evaluated.

Another interesting research direction raised within this thesis is the tracking of patient eye movements and using this as an indicator of higher complication rates. A dataset containing a much larger number of videos would be required to evaluate this, including several videos where complications occurred. A measurement of the skewed density function is recommended to make an appropriate measurement of the degree of eye movement throughout the video, which would provide a single metric for the volume of eye movement. An example of the eye movement density function is given in the section 5.3.

Detecting abnormal surgeries could hold some usefulness, especially for training purposes. Quantifying the abnormality of videos using a single metric could provide a measure of surgeon operative performance. It may also be a predictor for surgical

complications. To quantify the level of abnormality within a cataract surgery video the quantification of the video instrument activity timeline is required. This could, most simply, be done by using a frame prediction vector provided by the deep learning models. This vector would have the same length as the number of frames in the video, and each element would contain a list of binary instrument prediction values classified from the corresponding frame. This would provide a two dimensional representation of the instrument activity timeline throughout the video. These vectors could be compared to one another using Dynamic Time Warping, which allows for comparisons between vectors of differing lengths. Alternatively, the videos could be broken into n sections, and if an instrument is present in the majority of frames in a section it is given a true value in the vector element corresponding to that section. Thus, the two dimensional vector representations of the videos would be of the same size: $(n,\ 10)$. This allows a direct comparison, and the K-Means algorithm (among others) could be used to detect outlier or abnormal videos. This would consider proportional features between surgeries, but not time duration features. A "fingerprint" for each surgeon could be engineered, averaged from a number of expert surgeries. A surgeon's surgeries could then be measured by how they differ from the fingerprint in terms of these representation vectors. This could provide a measure of abnormality using a single metric.

Chapter 8

Conclusion

This thesis has involved the proposal of the deep learning framework and its evaluation. It has been demonstrated that an optimised prototype of the framework can operate in real time on an embedded system, providing accurate and representative clinical summary reports. Deep learning models have been trained and evaluated with two novel datasets, sampled from 39 cataract surgery videos. Further experiments have also been conducted to maximise the predictive performance of the deep learning models. Future research directions have been explored, including the use of eye movement for predicting surgical complications, detecting blurry instruments, and the use of image preprocessing in classification. Additionally, the clinical relevance of the reports has been proposed and discussed. The usefulness and operability of an automatic pipeline, from video live stream to a clinical summary report securely provided directly to a surgeon, has been validated. This pipeline has been demonstrated to operate in real time, able to be operated in a clinical environment on an embedded system.

The research in this thesis builds upon the existing body of research in this area, creating an application for the powerful classification models developed for cataract surgery. The embedded nature of the deep learning framework allows it to be operated securely and privately, while also allowing a straightforward and practical approach to developing an industry-ready product. The findings showing that our preprocessing methods are counterproductive to predictive performance are also valuable to this area

of research, as other research papers use similar methods. Being able to automatically quantify the contents of a cataract surgery video allows the building of needed objective surgeon assessment tools. Additionally, this framework can be used to improve surgeon training techniques by offering surgery timeline visualisations, instrument annotation statistics and video annotations.

In an aging society, the demand for cataract surgery is growing rapidly. Developing new technologies to accelerate the speed of training surgeons and to support cataract surgeries is paramount to maintaining a high quality of life. This thesis has explored the use of deep learning in providing surgery analysis to surgeons, integrating a new technology into the long history of cataract surgery techniques. To the best of our knowledge, this work is the first venture into objectively representing and analysing surgeries with the use of Instrument Activity Sequences. The success of this deep learning framework and the validation of its operation on embedded systems allows a novel, practical integration of deep learning into the workflow of modern cataract surgery techniques. A bright future of new deep learning applications await for cataract surgery.

Bibliography

[1] Dr. Noel Greis. Aging society – the global trend, its consequences, and the role of technology. Accessed 18/02/2022.

[2] Nation Eye Institute. Cataract data and statistics. Accessed 16/02/2022.

[3] West SK. Looking forward to 2020 a focus on the epidemiology of eye diseases. In *Epidemiol Rev. 2000;22(1)*, pages 64–70, 2000.

[4] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A.W.M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, 2017.

[5] Allen Foster. Vision 2020: The cataract challenge. *Community Eye Health*, 13, 01 2000.

[6] MD Francis Clark. A brief look at cataract statistics. Accessed 16/02/2022.

[7] Causes and treatment of cataracts, Feb 2021. Accessed: 31/5/2021.

[8] Bickol N. Mukesh, Anhchuong Le, Peter N. Dimitrov, Shazia Ahmed, Hugh R. Taylor, and Catherine A. McCarty. Development of cataract and associated risk factors the visual impairment project. *Archives of Ophthalmology*, 124(1):79–85, 01 2006.

[9] Cataract Surgery Information. Basic eye anatomy. Accessed 25/02/2022.

[10] Davis G. The evolution of cataract surgery. In *Missouri medicine, 113(1)*, pages 58–62, 2016.

[11] A. Haripriya, D. F. Chang, M. Reena, and M. Shekhar. Complication rates of phacoemulsification and manual small-incision cataract surgery at aravind eye hospital. In *Journal of cataract and refractive surgery, 38(8)*, pages 1360–1369, 2012.

[12] Cataracts cataract surgery, 2006.

[13] Types of cataract surgery. Accessed: 1/6/2021.

[14] The basics of phacoemulsification, May 2011. Accessed: 1/6/2021.

[15] Phacoemulsification video primer. Accessed: 1/6/2021.

[16] MD Mitchell A. Jackson. Update on capsular tension rings, May 2010. Accessed: 10/11/2021.

[17] Florent Lalys, Laurent Riffaud, David Bouget, and Pierre Jannin. A framework for the recognition of high-level surgical tasks from video images for cataract surgeries. *IEEE transactions on bio-medical engineering*, 59:966–76, 12 2011.

[18] Katia Charriere, Gwenole Quellec, Mathieu Lamard, Gouenou Coatrieux, Beatrice Cochener, and Guy Cazuguel. Automated surgical step recognition in normalized cataract surgery videos. volume 2014, pages 4647–50, 08 2014.

[19] Gwenole Quellec, Mathieu Lamard, Beatrice Cochener, and Guy Cazuguel. Real-time segmentation and recognition of surgical tasks in cataract surgery videos. *IEEE Transactions on Medical Imaging*, 33:2352–60, 12 2014.

[20] Katia Charrière, Gwenolé Quellec, Mathieu Lamard, David Martiano, Guy Cazuguel, Gouenou Coatrieux, and Béatrice Cochener. Real-time analysis of cataract surgery videos using statistical models, 2016.

[21] Saad Albawi, Tareq Abed Mohammed, and Saad ALZAWI. Understanding of a convolutional neural network. 08 2017.

[22] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[23] Christian Szegedy, V. Vanhoucke, S. Ioffe, Jonathon Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[24] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[25] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[26] Kyunghyun Cho, Bart van Merrienboer, cCaglar Gulccehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs1406.1078, 2014.

[27] Hassan Al Hajj, Mathieu Lamard, Katia Charrière, Béatrice Cochener, and Gwenolé Quellec. Surgical tool detection in cataract surgery videos through multi-image fusion inside a convolutional neural network. In *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 2002–2005, 2017.

[28] A. Raju, S. Wang, and J. Huang. M2cai surgical tool detection challenge report, 2016.

[29] M. Sahu, A. Mukhopadhyay, A. Szengel, and S. Zachow. Tool and phase recognition using contextual cnn features, 2016.

[30] A. P. Twinanda, S. Shehata, D. Mutter, J. Marescaux, M. de Mathelin, and N. Padoy. Endonet a deep architecture for recognition tasks on laparoscopic videos, 2017.

[31] A. Zia, D. Castro, and I. Essa. Fine-tuning deep architectures for surgical tool detection, 2016.

[32] Hassan Al Hajj, Mathieu Lamard, Pierre-Henri Conze, Soumali Roychowdhury, Xiaowei Hu, Gabija Maršalkaitė, Odysseas Zisimopoulos, Muneer Ahmad Dedmari, Fenqiang Zhao, Jonas Prellberg, Manish Sahu, Adrian Galdran, Teresa Araújo, Duc My Vo, Chandan Panda, Navdeep Dahiya, Satoshi Kondo, Zhengbing Bian, Arash Vahdat, Jonas Bialopetravičius, Evangello Flouty, Chenhui Qiu, Sabrina Dill, Anirban Mukhopadhyay, Pedro Costa, Guilherme Aresta, Senthil Ramamurthy, Sang-Woong Lee, Aurélio Campilho, Stefan Zachow, Shunren Xia, Sailesh Conjeti, Danail Stoyanov, Jogundas Armaitis, Pheng-Ann Heng, William G. Macready, Béatrice Cochener, and Gwenolé Quellec. Cataracts challenge on automatic tool annotation for cataract surgery. *Medical Image Analysis*, 52:24–41, 2019.

[33] Al Hajj H. Lamard M., Conze P. H., Cochener B., and G. Quellec. Monitoring tool usage in surgery videos using boosted convolutional and recurrent neural networks. *Medical image analysis, 47, 203–218*, 2018.

[34] Xiaowei Hu, Lequan Yu, Hao Chen, Jing Qin, and Pheng-Ann Heng. Agnet attention-guided network for surgical tool presence detection. pages 186–194, 09 2017.

[35] Odysseas Zisimopoulos, Evangello Flouty, Imanol Luengo, Petros Giataganas, Jean Nehme, Andre Chow, and Danail Stoyanov. Deepphase surgical phase recognition in cataracts videos, 2018.

[36] Manfred Jüergen Primus, Doris Putzgruber-Adamitsch, Mario Taschwer, Bernd Münzer, Yosuf El-Shabrawi, Laszlo Böszörmenyi, and Klaus Schoeffmann. Frame-based classification of operation phases in cataract surgery videos. In Klaus Schoeffmann, Thanarat H. Chalidabhongse, Chong Wah Ngo, Supavadee Aramvith, Noel E. O'Connor, Yo-Sung Ho, Moncef Gabbouj, and Ahmed Elgammal, editors, *MultiMedia Modeling*, pages 241–253, Cham, 2018. Springer International Publishing.

[37] Felix Yu, Gianluca Silva Croso, Tae Soo Kim, Ziang Song, Felix Parker, Gregory D. Hager, Austin Reiter, S. Swaroop Vedula, Haider Ali, and Shameema Sikder. Assessment of automated identification of phases in videos of cataract surgery using machine learning and deep learning techniques. *JAMA Network Open*, 2(4):e191860–e191860, 04 2019.

[38] Shoji Morita, Hitoshi Tabuchi, Hiroki Masumoto, Tomofusa Yamauchi, and Naotake Kamiura. Real-time extraction of important surgical phases in cataract surgery videos. *Scientific Reports*, 9:16590, November 2019.

[39] Negin Ghamsarian. Enabling relevance-based exploration of cataract videos. In *Proceedings of the 2020 International Conference on Multimedia Retrieval*, ICMR '20, page 378–382, New York, NY, USA, 2020. Association for Computing Machinery.

[40] Mehdi Noroozi, Paramanand Chandramouli, and Paolo Favaro. Motion deblurring in the wild. *CoRR*, abs1701.01486, 2017.

[41] Kaiming He, Georgia Gkioxari, Piotr Doll'ar, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs1703.06870, 2017.

[42] Natalia Sokolova, Klaus Schoeffmann, Mario Taschwer, Doris Putzgruber-Adamitsch, and Yosuf El-Shabrawi. Evaluating the generalization performance of instrument classification in cataract surgery videos. In Yong Man Ro, Wen-Huang Cheng, Junmo Kim, Wei-Ta Chu, Peng Cui, Jung-Woo Choi, Min-Chun Hu, and Wesley De Neve, editors, *MultiMedia Modeling*, pages 626–636, Cham, 2020. Springer International Publishing.

[43] Klaus Schoeffmann, Mario Taschwer, Stephanie Sarny, Bernd Münzer, Manfred Jürgen Primus, and Doris Putzgruber. Cataract-101: Video dataset of 101

cataract surgeries. In *Proceedings of the 9th ACM Multimedia Systems Conference*, MMSys '18, page 421–425, New York, NY, USA, 2018. Association for Computing Machinery.

[44] Gwenole Quellec, Katia Charriere, Mathieu Lamard, Beatrice Cochener, and Guy Cazuguel. Normalizing videos of anterior eye segment surgeries. volume 2014, pages 122–5, 08 2014.

[45] D.H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[46] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning, 2016.

[47] Fabio Villa. Monitoring surgical performance: Current models and limitations. *SOJ*, 02 2017.

[48] Sidharth Puri and Shameema Sikder. Cataract surgical skill assessment tools. *Journal of Cataract Refractive Surgery*, 40(4):657–665, 2014.

[49] Edward Casswell, Tahrina Salam, Paul Sullivan, and Daniel Ezra. Ophthalmology trainees' self-assessment of cataract surgery. *The British journal of ophthalmology*, 100, 09 2015.

[50] Tae Kim, Molly O'Brien, Sidra Zafar, Gregory Hager, Shameema Sikder, and Sunil Vedula. Objective assessment of intraoperative technical skill in capsulorhexis using videos of cataract surgery. *International Journal of Computer Assisted Radiology and Surgery*, 14, 04 2019.

[51] Negin Ghamsarian, Mario Taschwer, and Klaus Schoeffmann. Deblurring cataract surgery videos using a multi-scale deconvolutional neural network. In *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, pages 872–876, 2020.

[52] Adrian Rosebrock. Blur-detection-with-opencv. Accessed 13/01/2022.

[53] Ramya Venkatesh, Ramesh Ragala, and R.Jagadeesh Kannan. Squeezing deep learning into mobile devices. *International Journal of Recent Technology and Engineering (IJRTE)*, 7, 2019.

[54] Build from source. Accessed 17/01/2022.

[55] Taiwo Samuel Ajani, Agbotiname Lucky Imoize, and Aderemi A. Atayero. An overview of machine learning within embedded and mobile devices–optimizations and applications. *Sensors*, 21(13), 2021.

Appendix A

Appendix

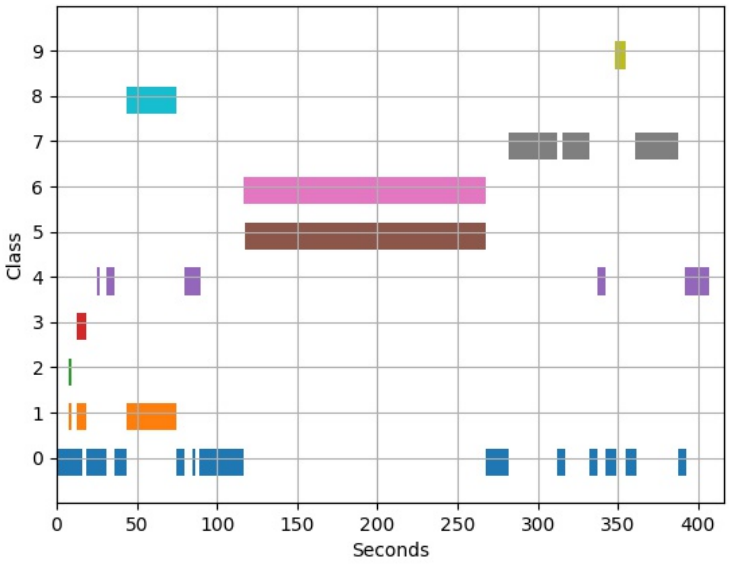

Figure A.1: Instrument sequence chart generated for video OS-2021-06-26_095120. Extracted at 12 FPS, using no preprocessing and using the InceptionV3 model-8e.
Key: **(0)** Idle Class, **(1)** Forceps, **(2)** 2.4mm Keratome Blade, **(3)** 1.1mm Paracentesis Blade, **(4)** Rycroft Cannula, **(5)** Chopper, **(6)** Phacoemulsification Probe, **(7)** Irrigation/Aspiration Probe, **(8)** Rhexis Forceps, **(9)** Lens Cartridge Injector.

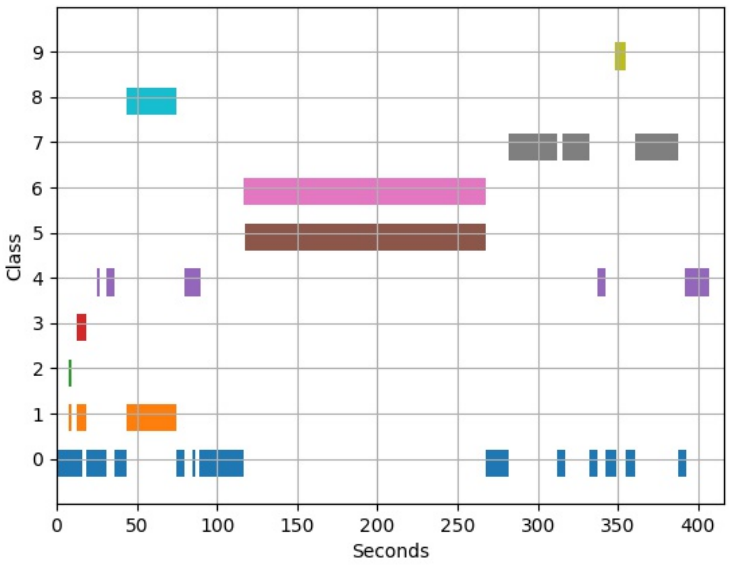

Figure A.2: Instrument sequence chart generated for video OS-2021-06-26_095120. Extracted at 6 FPS, using no preprocessing and using the InceptionV3 model-8e.

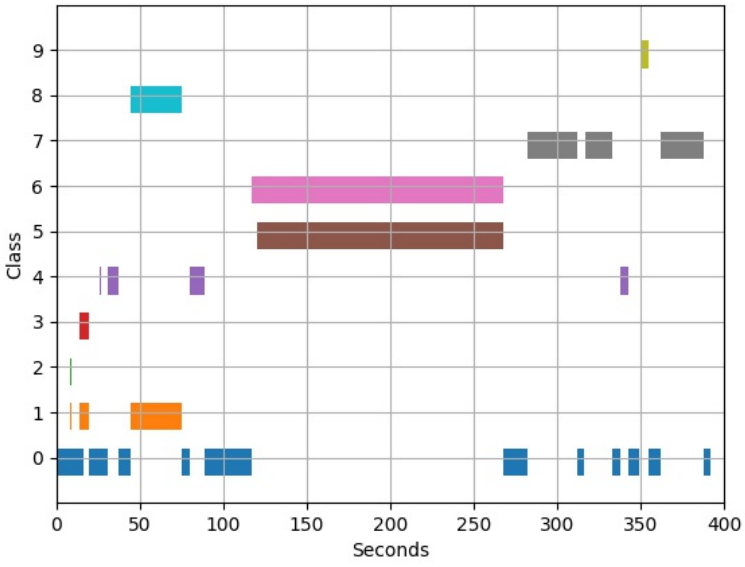

Figure A.3: Instrument sequence chart generated for video OS-2021-06-26_095120. Extracted at 1 FPS, using no preprocessing and using the InceptionV3 model-8e.

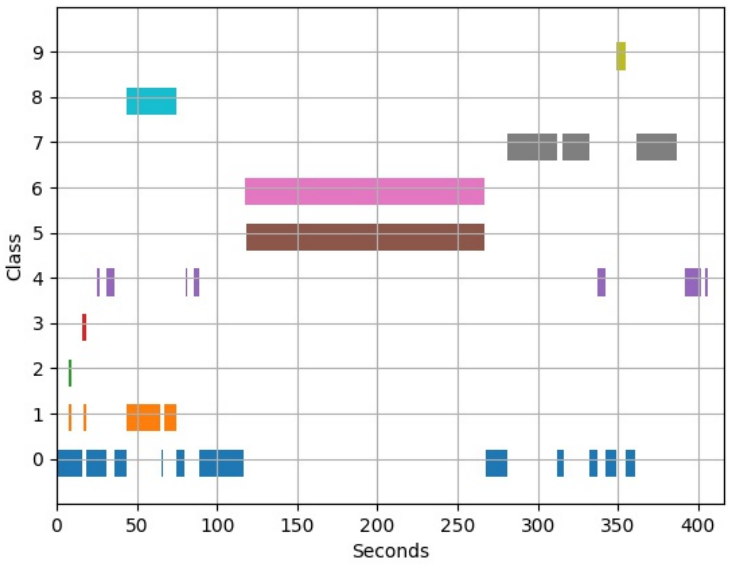

Figure A.4: Instrument sequence chart generated for video OS-2021-06-26_095120. Extracted at 6 FPS, using no preprocessing and using the ResNet-50 model-0e.

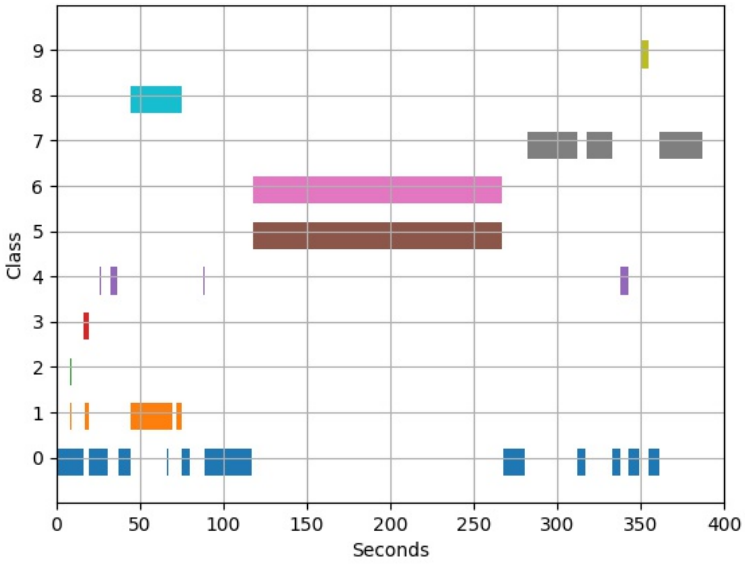

Figure A.5: Instrument sequence chart generated for video OS-2021-06-26_095120. Extracted at 1 FPS, using no preprocessing and using the ResNet-50 model-0e.

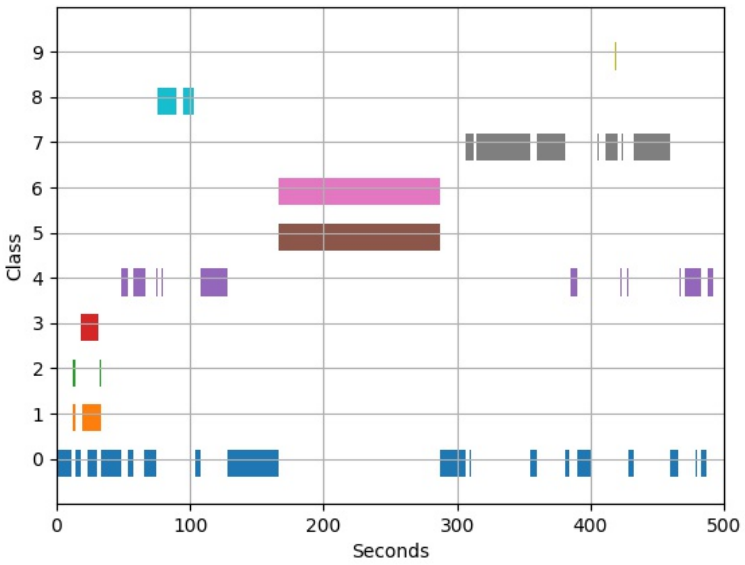

Figure A.6: Instrument sequence chart generated for video OS-2021-06-26_104327. Extracted at 1 FPS, using no preprocessing and using the InceptionV3 model-8e.

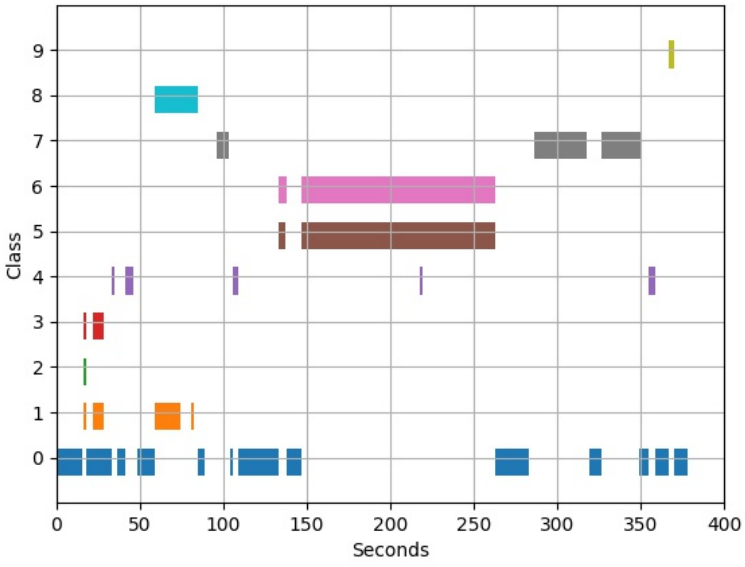

Figure A.7: Instrument sequence chart generated for video OS-2021-06-26_102811. Extracted at 1 FPS, using no preprocessing and using the InceptionV3 model-8e.

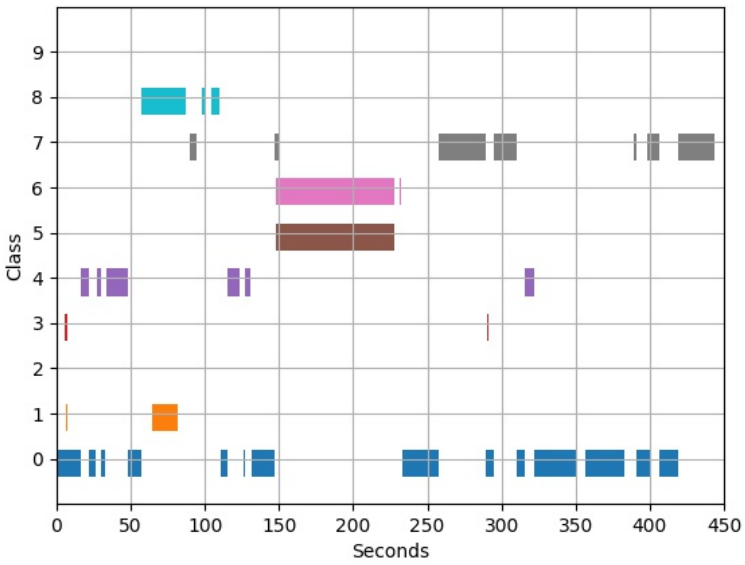

Figure A.8: Instrument sequence chart generated for video OS-2021-06-26_101130. Extracted at 1 FPS, using no preprocessing and using the InceptionV3 model-8e.

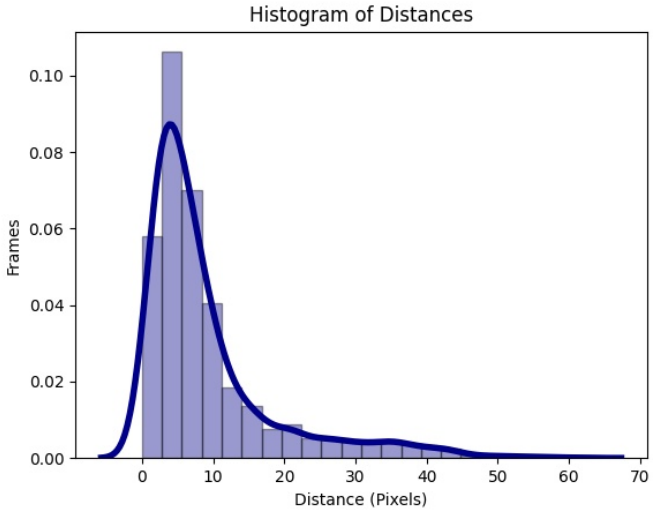

Figure A.9: A histogram and density function plot of the pixel eye movement distances found for the video OS-2021-12-04_074108.

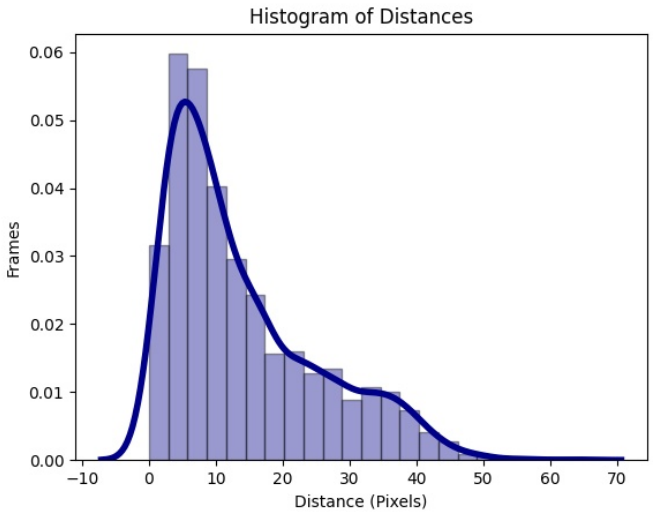

Figure A.10: A histogram and density function plot of the eye movement pixel distances found for the video OS-2021-12-04_075958.

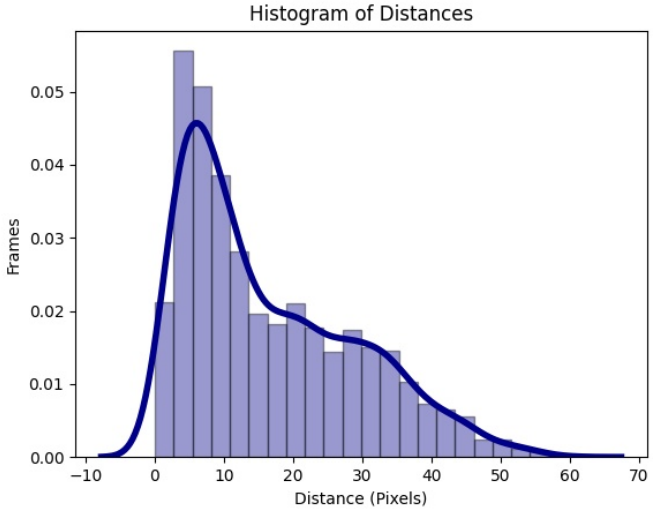

Figure A.11: A histogram and density function plot of the pixel eye movement distances found for the video OS-2021-12-04_081445.

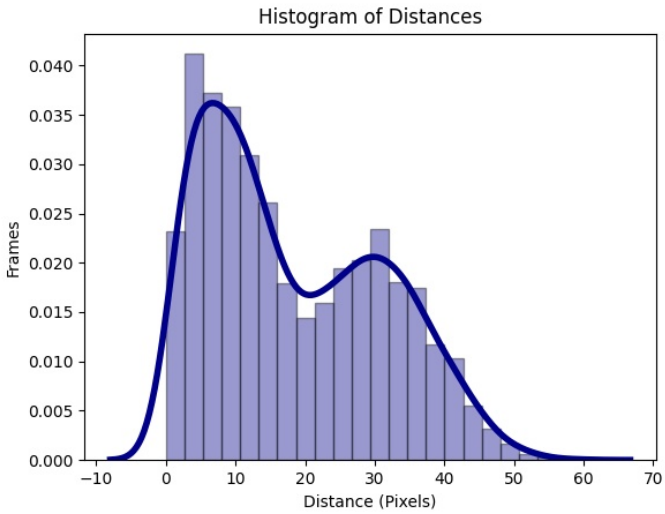

Figure A.12: A histogram and density function plot of the pixel eye movement distances found for the video OS-2021-12-04_084112.