Computer Vision for Cancer Cell Cycle Analysis

Imperial College London

Tom Halmos

CID : 01059252

Supervisors: Dr. Philipp Thomas, Dr. Alexis Barr

Department of Life Sciences

August 21, 2020

Contents

1 Abstract **2**

2 Abbreviations **2**

3 Introduction **2**

3.1 Live Cell Imaging 2

3.2 Current Analysis Techniques 4

3.3 Machine Learning for Computer Vision 5

3.4 Aims 9

4 Materials and Methods **11**

4.1 Data Acquisition and Processing 11

4.2 Software and Hardware 12

4.3 Data Augmentation 13

4.4 Evaluation Metrics 13

5 Results **15**

5.1 Hyper-parameter assignment 15

5.1.1 Learning Rate 15

5.1.2 Binary Threshold 16

5.2 Challenges for U-Net analysis in CCC research 18

5.2.1 Large Image Size 18

5.2.2 Lack of Training Data 19

5.2.3 Cell Crowding 23

5.3 Comparison of U-Net and Watershed Segmentation 24

5.4 Proof of Concept 26

6 Discussion **27**

6.1 Over-Fitting 27

6.2 Track-Net Re-design 28

6.3 Responsible Research and Innovation 29

6.4 Conclusions and Future Work 30

7 Acknowledgements **31**

8 Bibliography 31

1 Abstract

Recent developments in deep learning for computer vision now extend simple feature extraction to a far deeper understanding of image content. This comes at a time when developments in microscopy, fluorescent labelling, and high-throughput experimental techniques can generate significant amounts of image data for analysis. This is a major bottle neck in cancer cell cycle research, where a lack of fully-automated image analysis tools is slowing further insight into the mechanisms of cell proliferation decisions. This project implements a novel neural network design with previous success in biomedical image analysis, for fully automatic single cell measurements in cancer cell cycle research. Several challenges for automated image analysis in the field are addressed, and a proof of concept analysis pipeline is presented, evaluated, and compared to current analysis techniques.

2 Abbreviations

CDK	Cyclin Dependent Kinase
PCNA	Proliferating Cell Nuclear Antigen
CCC	Cancer Cell Cycle
CNN	Convolutional Neural Network
DeLTA	Deep Learning for time-lapse Analysis

3 Introduction

3.1 Live Cell Imaging

Live cell imaging is a popular technique in the biomedical sciences where dynamical measurements of single cells are used to investigate processes such as cell fate determination and plasticity (Jeknić, Kudo & Covert, 2019; Spiller et al., 2010). In the field of cancer cell cycle research (CCC), this technique is used to better understand cell proliferation decisions and inform

improved cancer therapeutics.

Healthy mammalian tissue depends on these tightly regulated cell proliferation decisions by balancing regenerative growth, with non-proliferative maintenance (Yao, 2014). Factors such as cell crowding, DNA damage, or the absence of growth factors, can trigger a non-proliferative state known as quiescence, in which cells do not divide. Quiescence induced by DNA damage is thought to prevent the propagation of non-lethal, but potentially harmful mutation among a population of cells (Barr et al., 2017). In this case, quiescence entry is controlled by the CDK inhibitor p21, which accumulates under DNA damage and ultimately prevents mitosis. p21 is subject to mutual inhibition from other cell cycle agents, resulting in a bi-stability of its cellular concentration and a bifurcation of the quiescence-proliferation decision (Heldt et al., 2018). Understanding this regulatory system is significant for cancer therapeutics, where hyper-proliferation (quiescence exit) is tumorigenic, and hypo-proliferation (quiescence entry) can confer resistance to anti-proliferative drugs (Chen et al., 2016).

In current research, quiescence regulation is investigated by dynamical measurements of cell cycle proteins in living cells. These proteins, such as p21, or the cell cycle progression reporter PCNA (proliferating cell nuclear antigen) are tagged with different fluorescent markers and imaged using a multi-channel microscope (Barr et al., 2017) (see Figure 1).

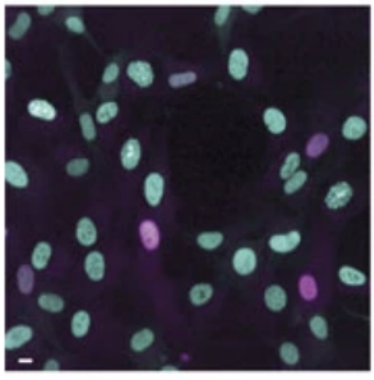


Figure 1: **Typical multi-channel image taken in CCC research.** In this experiment, cell cycle stage is monitored in individual hTert-RPE1 cell nuclei with mRuby tagged PCNA (turquoise), and later correlated with p21-GFP levels (magenta) for insights into the role of p21 in cell fate determination (Barr et al., 2017). Scale bar is 10 μm.

These imaging experiments generate time-lapse footage of crowded, free-moving, asynchronously

cycling cells, which need to be segmented into individual nuclei and tracked across multiple frames before single-cell measurements can be extracted (see Figure 2). Once segmented and tracked, metrics such as nuclear area and PCNA fluorescence are extracted and used to predict cell cycle stage. The progression through each stage is then correlated to the activity of a fluorescently tagged protein to investigate its role in the cell proliferation decision.

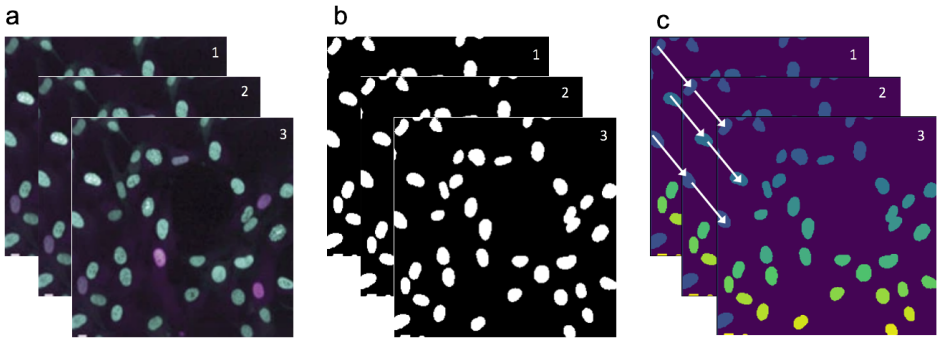


Figure 2: **Image analysis pipeline for a typical experiment in CCC research. (a)** Raw time-lapse images **(b)** Segmentation masks **(c)** Annotated nuclear trajectories. Frame numbers (top right) correspond to the 10-minute imaging interval.

3.2 Current Analysis Techniques

Typically, segmentation is based on the 'Watershed' algorithm, where image intensity minima are treated as basins and metaphorically filled with water. Watershed is generally successful for the segmentation of biomedical images (El Allaoui, 2012), but is prone to splitting single objects into multiple segments, and requires manual parameter optimisation between images (Kornilov & Safonov, 2018).

Cell tracking is typically based on cell-cell overlap between consecutive time-lapse frames, which is only effective when cell density and motility are low (Chalfoun et al., 2010). Recent work has adapted a maximum likelihood approach to tracking based on the Viterbi algorithm (Quach & Farooq, 1994), to deal with the challenges of tracking live cells, such as mitosis, apoptosis and migration out of the field of view (Magnusson et al., 2015). A custom analysis package developed for CCC research known as NucliTrack, implements this approach while prioritising user-friendliness for non-computer scientists (Cooper et al., 2017).

These semi-automatic image analysis methods see continued use in live cell imaging research given their accessibility, and relatively high accuracy (Hsu, Altschuler & Wu, 2019; Barr et al., 2017; Stewart-Ornstein & Lahav, 2016). However, they require close supervision to prevent error accumulation, which can can quickly make data meaningless, and are now a bottleneck to more high-throughput analysis, particularly in CCC research. Fully automated, supervision-free analysis, would make high-throughput experiments more feasible, and ultimately accelerate the therapeutic benefits of CCC research to the public.

3.3 Machine Learning for Computer Vision

Computer vision is a field of computer science which aims to automate the analysis of digital images. Modern techniques in the field use a machine learning approach to extend simple feature extraction algorithms, such as edge detection (Canny, 1986), to a deeper understanding of image content (Moen et al., 2019).

These machine learning algorithms allow computers to perform a task without direct instruction. Early forms of these algorithms were able to detect and classify simple patterns in various data sets, but were limited by the need to first transform the data into a more suitable representation. By extending the learning process to include this transformation, machine learning algorithms could automatically learn the most effective representation for data sets previously too complex for manual feature extraction. Deep-learning methods combine multiple transformations to extract data features in increasingly abstract representations of the input data. In practice, this makes deep learning an effective image analysis tool, by transforming the image into an abstract representation where noise, artefacts, and contextual information common to image data can be suppressed, and the feature of interest more easily recognised (LeCun, Bengio & Hinton, 2015).

These feature extractions are an emergent property of the 'neuron', the simple processing unit and building block of a neural network. Like their biological counterparts, neurons return a single output from multiple inputs, making them simple decision making units. In a

neural network, neurons communicate by passing this output to other neurons, where each neuron-level decision contributes to the output of the whole network. Mathematically, the transformation of input to output within the neuron is controlled by a set of parameters known as weights, which can be adjusted to change the behaviour of the neuron. Training a neural network finds the optimal set of weights that produce the lowest error on a given task. This error is calculated by a 'loss' function, which evaluates the network output on a training data set that contains examples of how the task should be performed. An algorithm known as back propagation is then used to tweak the network weights in the correct direction to reduce the value of this loss function. This process can be visualised by considering the loss function as a multi-dimensional surface, where each point corresponds to a set of weights and an associated loss value. Finding the optimal network therefore corresponds to descending the gradient of the loss surface until the global minima is located, where the set of weights are most effective for a given task (LeCun, Bengio & Hinton, 2015)(see Figure 3).

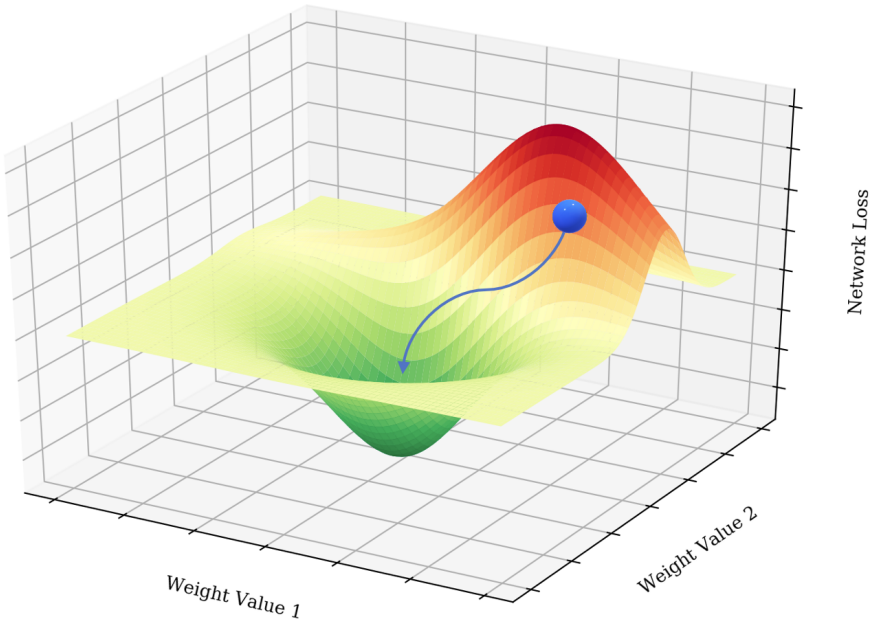


Figure 3: **Visualisation of Machine Learning.** In this case, network loss is represented by a three dimensional surface defined by two weight values from a single neuron. Each point on the surface corresponds to a possible state of the network (combination of weights) with an associated loss value. Given a random starting point (blue sphere), back propagation and gradient descent algorithms are used to iteratively update the network towards the global minima (blue arrow), where the set of weights are most effective for a given task.

For early image analysis tasks, neurons were arranged into multilayered networks where each neuron in the first layer corresponded to a single pixel in the input image. A major breakthrough in deep-learning for image analysis came with the development of the convolutional neural network (CNN) by LeCun et al. (1990), which redesigned the way neural networks process image data. Rather than a single pixel corresponding to a single input, the CNN arranges neurons into multiple convolutional kernels, which scan across the image as specialised feature extractors. 'Pooling layers' then uncouple these features from their exact location and appearance, allowing them to be detected more robustly across multiple images (LeCun, Bengio & Hinton, 2015).

From the CNN, the U-Net CNN was recently developed, with a larger and more complex architecture (Ronneberger, Fischer & Brox, 2015). While CNNs make excellent feature extractors for classification, such as recognising whiskers and classifying a cat, they are not designed to maintain the location and resolution of individual features during analysis. In the U-Net CNN, up-sampling layers combine the abstract convolutions of the deepest layers, with high resolution information from earlier layers to reconstruct a labeled version of the original image, rather than a general classification (see Figure 4). For this reason, the U-Net CNN is effective in the segmentation of biomedical images, where each pixel is classified as foreground or background (Oktay et al., 2018; Lugagne, Lin & Dunlop, 2020; Zhang & Xing, 2018; Li et al., 2018).

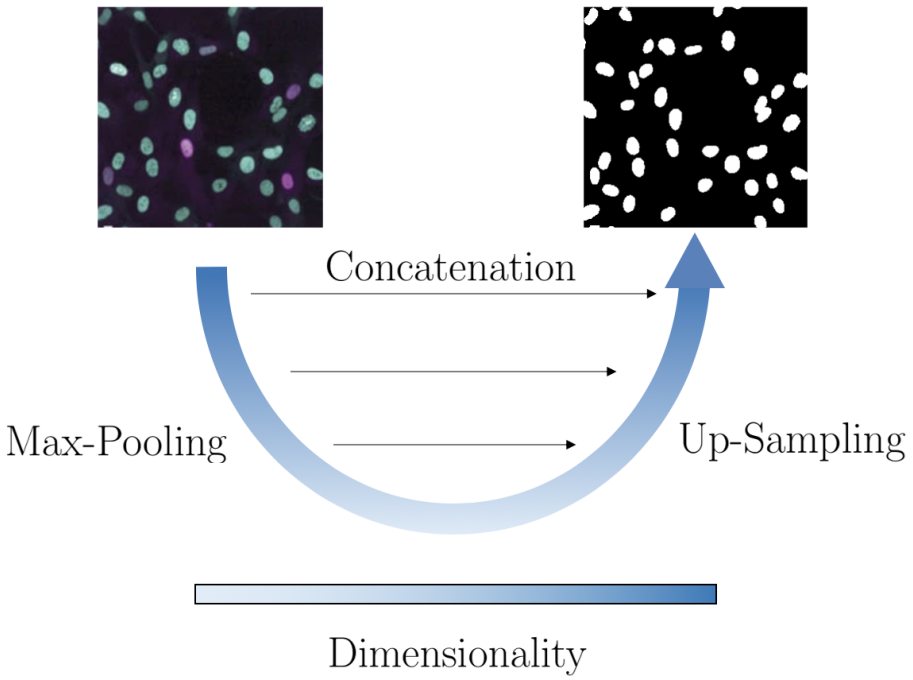


Figure 4: **Schematic of a typical U-Net CNN.** The U-Net CNN transforms a raw image into a binary segmentation mask by classifying each pixel as background or foreground. The input is passed through the U-Net by a series of convolutions, which lead into a lower dimensional space in the contracting path via Max-Pooling operations. This is followed by a return to the original image dimensions by Up-Sampling operations in the expanding path. The contracting path extracts deep image features, while the expanding path uses concatenation with earlier, high resolution layers, to piece these features back together into a segmentation of the original image (Ronneberger, Fischer & Brox, 2015).

Recently, a U-Net CNN has also been implemented for cell tracking, as part of the DeLTA image analysis framework (Lugagne, Lin & Dunlop, 2020). The tracking U-Net matches corresponding cells between frames, and handles mitosis by identifying the daughter cell (see Figure 5). This tracking U-Net was combined with a segmentation U-Net into a fully automatic image analysis pipeline on which the work in this project is modelled (see Figure 6).

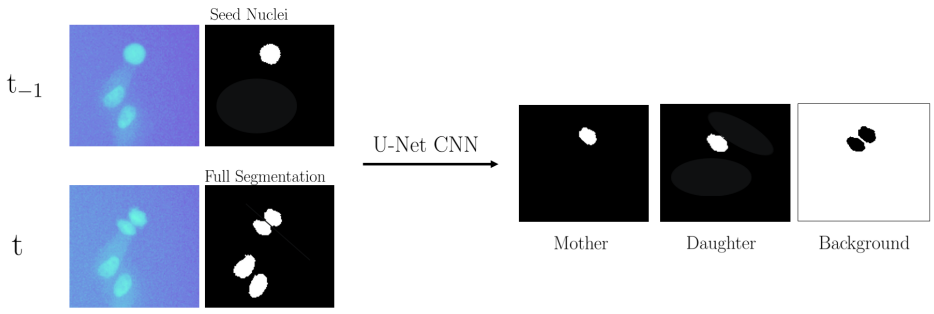


Figure 5: **Schematic of a U-Net CNN used for cell tracking.** The inputs of the U-Net are the raw image and segmentation mask of a seed nucleus from the previous time point, along with the raw image and full segmentation mask of the current time point. The output arrays select the nuclei to which the seed nucleus tracks to in the current segmentation. In this case a mitosis event requires the output of both a mother and daughter nuclei. The background array is the inverse of the mother and daughter array combined, and used during training (Lugagne, Lin & Dunlop, 2020).

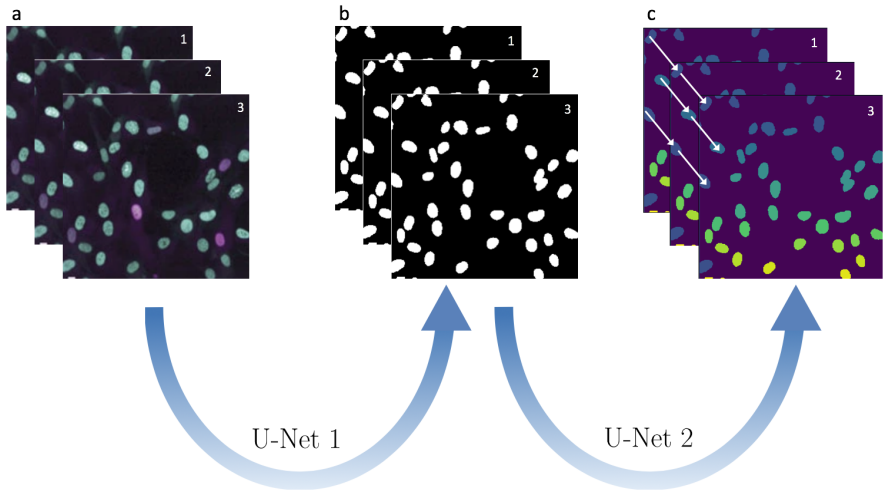


Figure 6: **Schematic of the DeLTA double U-Net image analysis pipeline.** Raw time-lapse images (a) are converted to binary segmentation masks (b) by the segmentation U-Net. Each nucleus is then tracked across multiple frames using the tracking U-Net (c). Both U-Nets have the same architecture but are trained to perform different tasks (Lugagne, Lin & Dunlop, 2020).

3.4 Aims

This project aims to adapt and implement the DeLTA double U-Net pipeline for fully automatic image analysis in CCC research. This will extend U-Net analysis from the bacterial cells confined to microfluidic chambers analysed by DeLTA, to the free moving mammalian cell cultures imaged in CCC research. Future work can then integrate the U-Net pipeline

into a software tool that will minimise manual supervision and increase analysis throughput.

To implement the U-Net pipeline, several challenges presented by the image data in CCC research must be addressed. Firstly, images are considerably larger than those analysed by DeLTA, which can lead to memory issues during network training and significantly higher analysis times. Secondly, no training data set is available, and manually generating one is time consuming. Networks trained on small data sets are unlikely to be able to generalise to new experiments, cell lines or microscopes. Cell crowding in the images themselves is a major challenge for segmentation, and can lead to error accumulation during tracking if cells are merged erroneously. Moreover, cells in CCC research move freely and show more morphological variation than the *E. coli* cells analysed by DeLTA.

This project will address these challenges as follows;

1. **Large image size**
   (a) Reduce the size of the segmentation U-Net to decrease computational cost, while ensuring no loss in segmentation quality.
2. **Lack of training data**
   (a) Use extensive data augmentation techniques to amplify the size of a small image data set.
   (b) Investigate the applicability of existing training data sets from similar fields of research.
3. **Cell crowding**
   (a) Implement a weighted training scheme to focus on areas of high cell density (Ronneberger, Fischer & Brox, 2015).

4 Materials and Methods

4.1 Data Acquisition and Processing

Training images for the U-Nets were provided by the Barr Lab (LMS) with additional images sourced from the Broad Bioimage Benchmark (BBBC). Barr Lab images consisted of m-Ruby PCNA tagged A549 cells imaged with an Opera HC spinning disk confocal microscope (PerkinElmer). BBBC images were sourced from the BBBC039v1 U2OS Hoechst data set (Caicedo et al. 2018), which was specifically designed for the testing and validation of image analysis algorithms (Ljosa, Sokolnicki & Carpenter, 2012).

Raw images were processed in Python using the Open Source Computer Vision Library (OpenCV) into gray scale, 32 bit, single channel arrays with normalised intensity. To resize images to 1024x1024 pixels, Barr Lab images were cropped down from 1080x1080, and U2OS

images of size 520x696 were combined in pairs with random offset into larger images and cropped to the correct dimensions.

Ground truth masks (ideal segmentations) were manually produced using the K-Space Segmentation Tool Set which was chosen for complete manual control over segmentation (McGuinness & O'Connor, 2008). The training data set for tracking was generated using a graphical user interface developed by the Dunlop Lab in Matlab R2018b, which automatically generates U-Net training arrays from manually selected cell trajectories (Lugagne, Lin & Dunlop, 2020). The interface was adapted for this project to account for modification in the tracking U-Net design (see 6.2). In total, 828 cell tracks and 23 full images were manually annotated using these methods.

4.2 Software and Hardware

U-Nets were trained using a pixel-wise cross entropy loss function based on the original U-Net publication (Ronneberger, Fischer & Brox, 2015) (see Equation 1),

$$-\sum_{x \in X} w(x)p(x)log(q(x)) \tag{1}$$

where $p(x)$ and $q(x)$ refer to the true label and predicted label of a pixel x respectively, and $p(x)log(q(x))$ calculates the cross entropy at that position. The cross entropy is then multiplied by the value of the weight map at the same position ($w(x)$) This calculation is repeated for each pixel pair in the ground truth and corresponding predicted mask ($x \in X$) to find the total pixel-wise cross entropy loss. This loss function supports the weighting of cross entropy on the pixel-level, which allows training to be directed to areas typically difficult to segment, such as those with high cell density. Without this weighting, these areas can be overlooked during training given that they make up only a fraction of the input image.

Weight maps are generated pixel-wise by finding the average intensity in a square window of side 40 pixels at each position in a ground truth mask. A global threshold of 75% maximum average intensity is set to remove noise, and nuclear pixels are assigned a value of 25%

maximum average intensity for partial weighting during training. 'Min-max' normalisation is then used to scale all values into the range 1-10. The Python implementation of this algorithm is available on the GitHub for this project (www.github.com/tomhalmos/cyclevision).

Training was implemented with Tensforflow 2.0/Keras in Python3.6 and run from a virtual environment maintained with the Anaconda data science platform. A YAML file of dependencies is available at the GitHub repository for this project along with U-Net training and data processing scripts. Python scripts were written and tested locally on a MacBook Pro followed by execution on a remote SSH server kindly provided by Philipp Thomas and the Imperial College Mathematics Department.

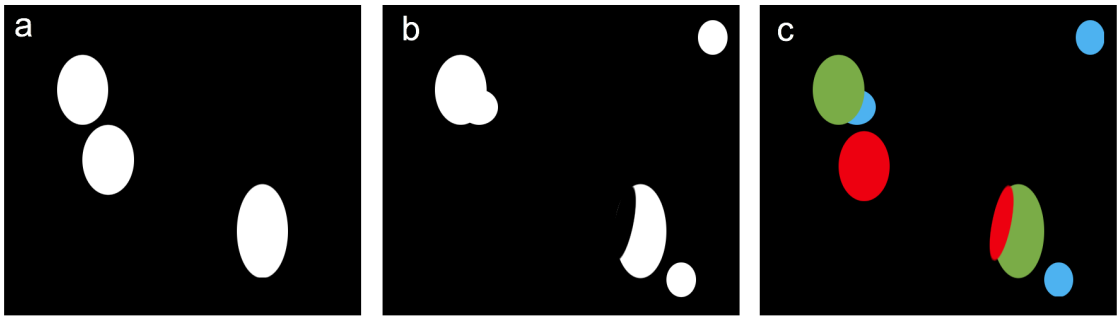
4.3 Data Augmentation

Data augmentation was performed on-the-fly during training with custom Python data generators. Training images were passed to a series of transformation algorithms with a 50% chance of application, such that images could remain unchanged or be transformed by one or multiple operations. Rotation and flipping were implemented manually using the Scipy and Numpy Python packages, elastic deformation was implemented as described in the original U-Net publication using the ElasticDeform package (Ronneberger, Fischer & Brox, 2015), and code for illumination distortion was provided by Daniel Eaton for DeLTA (Lugagne, Lin & Dunlop, 2020).

4.4 Evaluation Metrics

To evaluate segmentation quality the machine learning metrics 'precision and recall' were adapted to CCC images at both the pixel and object-level. In general, precision is a measure of relevancy, and takes into account the number of false positive errors committed. Recall is measure of inclusion, and considers the number of false negative errors. For example, a segmentation method that incorrectly predicts many nuclei where there are none, would have low precision, while a method that fails to segment multiple true nuclei would have low recall.

On both the object and pixel level, precision and recall are calculated by comparing the output of a segmentation algorithm to the ground truth segmentation mask (see Figure 7).



Object Level Precision:

$$\frac{True\ positive\ nuclei}{Total\ predicted\ nuclei} = \frac{2}{4} = 50\%$$

Pixel Level Precision:

$$\frac{True\ positive\ pixels}{Total\ predicted\ pixels} = \frac{G}{G+B} = \sim 90\%$$

Object Level Recall:

$$\frac{True\ positive\ nuclei}{Total\ true\ nuclei} = \frac{2}{3} = 66.6\%$$

Pixel Level Recall:

$$\frac{True\ positive\ pixels}{Total\ true\ pixels} = \frac{G}{G+R} = \sim 75\%$$

Figure 7: **Calculation of object and pixel-level precision and recall. a)** Ground truth mask **b)** Predicted segmentation **c)** Prediction labeled with true positive pixels (green), false negative pixels (red) and false positive pixels (blue). On the object level, two true positive nuclei are correctly segmented, along with two false positive and one false negative nuclei. This results in a precision of 50%, given that half of the predicted objects are actually nuclei, and a recall of 66.6% given that two of the three true nuclei are correctly identified. On the pixel level, the letters G, B and R refer to the color of the pixel group in the annotated prediction, and result in estimated precision and recall scores of 90 and 75% respectively. Only error pixels associated with a true positive nucleus are included in the calculation. The pixel-level metrics are therefore useful in evaluating the accuracy of the size and shape of any correctly identified nuclear objects, while object-level metrics are more useful for evaluating the inclusion and loss of the nuclear objects themselves.

For the object-level calculation, an overlap based method is used to map ground truth nuclei to predicted nuclei. This information is then used to determine whether a predicted nucleus is true positive, or false positive/negative. For example, a predicted nucleus that does not map to any ground truth nuclei, is labeled as false positive. The mapping information is also used to count the number of split, created, absent and merge type segmentation errors (SCAM). Where 'split' errors refer to a single true nucleus that is segmented into multiple nuclei, 'created' and 'absent' to false positive and false negative errors respectively, and 'merge' errors to multiple true nuclei that are segmented into a single nucleus. Full python scripts for the automatic calculation of SCAM errors along with object and pixel-level

precision and recall are freely available on the GitHub repository for this project.

5 Results

5.1 Hyper-parameter assignment

The default parameters that define U-Net structure and training behaviour were replicated from DeLTA and the original U-Net publication (Ronneberger, Fischer & Brox, 2015). These parameters, known as 'hyper-parameters', need to be optimised for each new implementation of a neural network and are not automatically learnt or improved during training. Unsuitable hyper-parameters can prevent effective training as well as waste computational resources and time (Yamashita et al., 2018).

5.1.1 Learning Rate

The learning rate hyper-parameter controls the size of the step taken across the surface of the loss function during gradient descent (see Figure 3). Too small, and the global optimum is located too slowly, or training can become trapped in local minimum. Too large, and training may overshoot any minima and not converge. The suitability of a learning rate can be evaluated from the network loss decay curve produced during training. In this case, a learning rate of 10^{-4} was found to be most suitable, in agreement with the value used in DeLTA (see Figure 8).

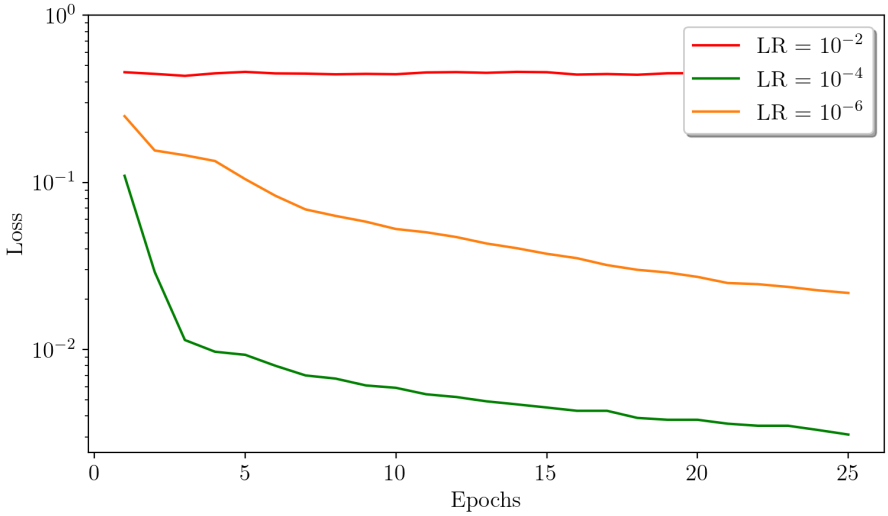


Figure 8: **Assessment of the learning rate hyper-parameter.** Three identical U-Nets were trained with a learning rate (LR) of 10^{-2}, 10^{-4} or 10^{-6} respectively. Loss decay curves were generated by calculating network loss at the end of each training cycle (epoch). A learning rate of 10^{-2} (red) was too large for convergence, while 10^{-6} (orange) converges slowly and to a non-global minimum. A learning rate of 10^{-4} avoids both these issues and converges quickly.

5.1.2 Binary Threshold

The output of the segmentation U-Net is an array in which pixel intensity indicates the probability that a given pixel in the input image belongs to a nucleus (see Figure 9; Raw output). The binary threshold hyper-parameter sets the cut-off probability used to determine whether a pixel is nuclear or background and create the binary segmentation mask.

A visual evaluation showed that a default binary threshold of 0.5 caused pixel-level over-segmentation and merging, while a higher threshold resulted in the loss of weakly predicted nuclei. Both these effects were observed in different parts of the same image, making the global binary threshold difficult to optimise (see Figure 9).

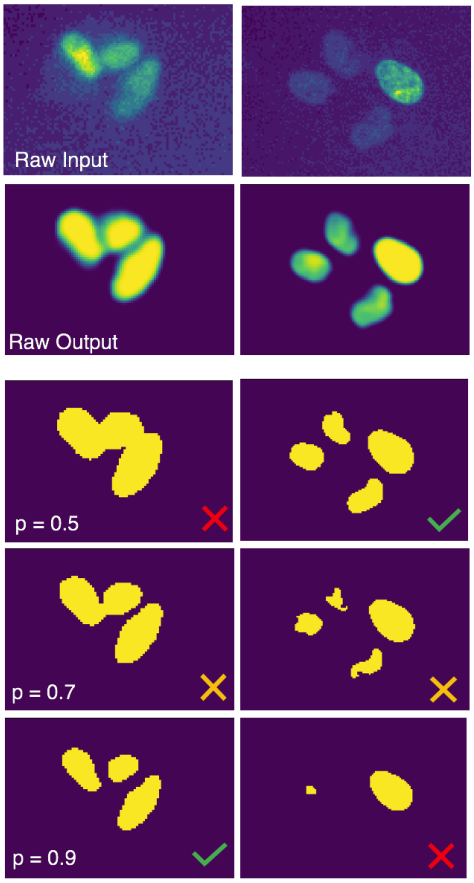


Figure 9: **The qualitative effect of the binary threshold hyper-parameter on segmentation quality.** Two regions of the same image (raw input), where analysed using a U-Net (raw output), and then segmented using an increasing binary threshold (p). The colored markers refer to poor (red cross), acceptable (orange cross) and good (green tick) segmentation quality. A low binary threshold (p=0.5) effectively segments the weakly predicted nuclei (right column) but fails to segment the group of crowded nuclei (left column). A higher threshold (p=0.7) partially retains the weakly predicted nuclei, but does not fully segment the crowded nuclei. The highest binary threshold (p=0.9) effectively segments the crowded nuclei, but omits several of the weakly predicted ones. None of the binary threshold values generate a high quality segmentation in both cases.

To quantify this effect, a set of validation images were analysed by the U-Net and segmented with a range of binary threshold values. The number of split, absent, created and merge type segmentation errors (SCAM), as well as pixel-level precision and recall (see Materials and Methods; Evaluation Metrics) were calculated from each segmentation mask (see Figure 10). By considering both object and pixel-level metrics, the binary segmentation threshold was increased from the default 0.5 to 0.75.

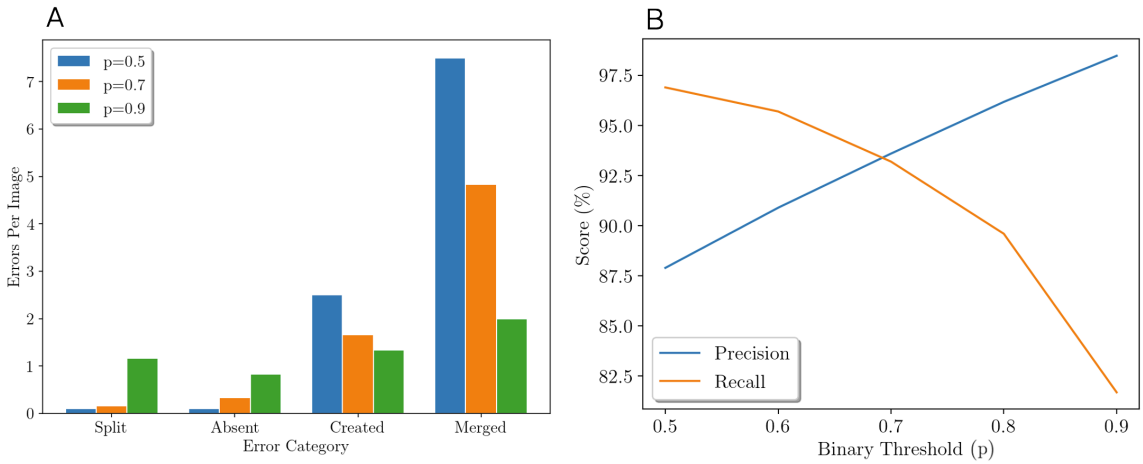


Figure 10: **The quantitative effect of the binary threshold hyper-parameter on segmentation quality. A)** Object-level SCAM errors **B)** Pixel-level precision and recall. As indicated by the visual analysis in Figure 9, a higher binary threshold increases the number of split and absent type segmentation errors, while decreasing the number of created and merged nuclei. In absolute terms, total SCAM errors are lowest at higher binary thresholds given the large decrease in merge type errors. Pixel-level precision and recall (B) shows a similar trade off but in terms of nuclear segmentation shape and size. The data shows that pixel-level segmentation is most accurate with a binary threshold of 0.7. A binary threshold of 0.75 was chosen at a small cost to precision-level recall to further reduce merge type segmentation errors.

5.2 Challenges for U-Net analysis in CCC research

Once the above hyper-parameters were optimised, solutions to each of the challenges for U-Net implementation in CCC research were investigated.

5.2.1 Large Image Size

To efficiently analyse large CCC images, the size of the U-Net was reduced in an attempt to decrease computational costs and analysis time. U-Net size is measured by the number of trainable parameters, which is calculated from the number of convolutional filters per layer, and the number of those layers. There is currently competing evidence as to which size factor is more important for network performance, (He et al., 2015; Zagoruyko & Komodakis, 2017) but in general, reducing network size is known to cause under-fitting, where a network has insufficient computational power to perform a task (Dietterich, 1995). Any reduction in the number of convolutional filters (network width) caused a significant loss in segmentation quality, specifically merge type errors (see Figure 11). This is most likely due to the under-fitting effect, and for this reason, full U-Net size was maintained at the cost

of analysis time and computational requirements.

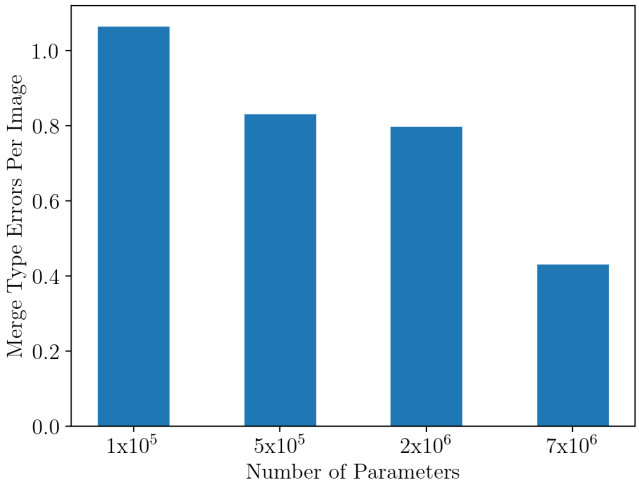


Figure 11: **Effect of network size on segmentation quality.** Four U-Nets of decreasing size were trained and evaluated for merge type segmentation errors. Size was reduced by decreasing the number of convolutional filters per layer, and keeping the number of layers constant. Any reduction in size from $7x10^6$ parameters lead to a significant rise in average merge type errors per image as a result of under-fitting. The number of parameters is shown to one significant figure.

5.2.2 Lack of Training Data

Neural networks trained on small data sets are prone to over-fitting, where the network learns specific features of the training data set rather than the general rules of the task. For this project no training data set was available, and to generate the same size data set used for DeLTA would require $\sim$ 200 hours of manual segmentation. To address this, an existing data set from a similar field was tested for applicability, and a technique known as data augmentation was used to artificially increase the size of the training data set.

A set of Hoechst-labeled U2OS cell images (see Materials and Methods; Data Acquisition and Processing) was combined with CCC images of PCNA-labeled A549 cells to create a larger training data set. U2OS nuclear area was shown to be more diverse but still within the typical range of A549 nuclei (see Figure 12). The U2OS nuclei were deemed similar enough to be useful in training an A549 U-Net, while providing robustness with examples of diverse nuclear morphology.

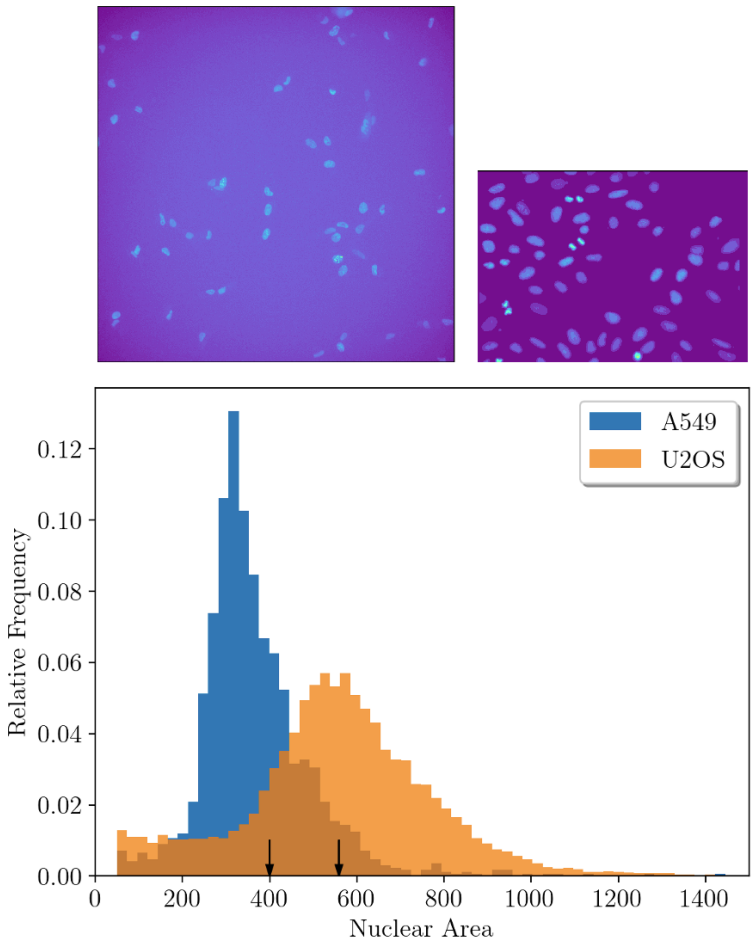


Figure 12: **Comparison of U2OS and A549 cell types.** Top: (Left) Typical A549 PCNA-labeled image, (Right) Typical U2OS Hoechst-labeled image. Bottom: Relative frequency histograms of nuclear area for the two cell types with arrows at mean nuclear area. U2OS nuclei show more diversity, but average nuclear size is comparable for both cell types

Networks trained on the combined U2OS/A549 data set showed improved segmentation quality on a validation data set of A549 cells (see Figure 13).

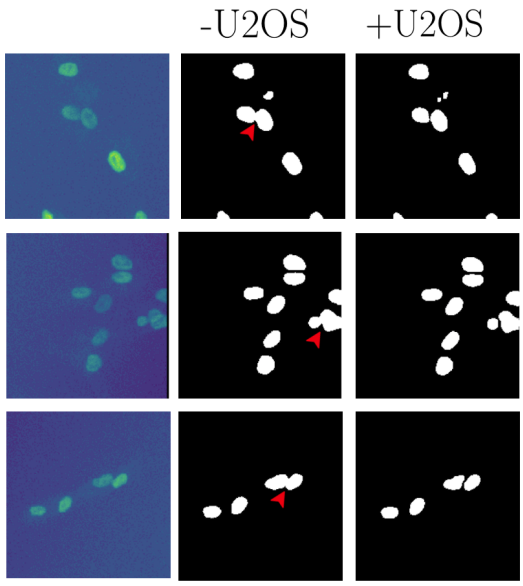


Figure 13: **The effect of U2OS training images on A549 segmentation quality.** Several A549 PCNA-labeled images were analysed by U-Nets trained with (+) or without (-) the U2OS training data set. The -U2OS U-Net committed more merge type segmentation errors (red arrows) than the the +U2OS U-Net.

Networks trained on the combined data set also dealt with variations in image illumination more effectively, making a more robust U-Net (see Figure 14).

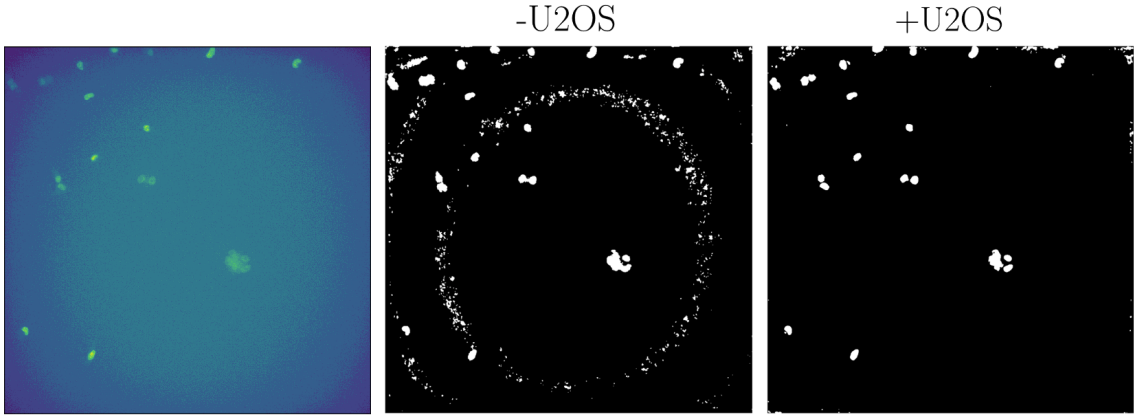


Figure 14: **The effect of U2OS training images on the segmentation of illumination artefacts.** The +U2OS U-Net ignores the flat-field illumination artefact seen in the raw image on the left, while the -U2OS U-Net erroneously segments the ring regions.

These results show that combining a small set of task-specific images (A549) with a larger set of similar but diverse images (U2OS), can improve U-Net accuracy and generalisation.

Data augmentation techniques were then used to artificially expand the size of the combined

U2OS/PCNA data set. This included simple transformations such as flipping and rotation, as well as more complex elastic and illumination distortions. Each of these were implemented for CCC images (see Figure 15) and applied in sequence to generate large image variation (see Figure 16).

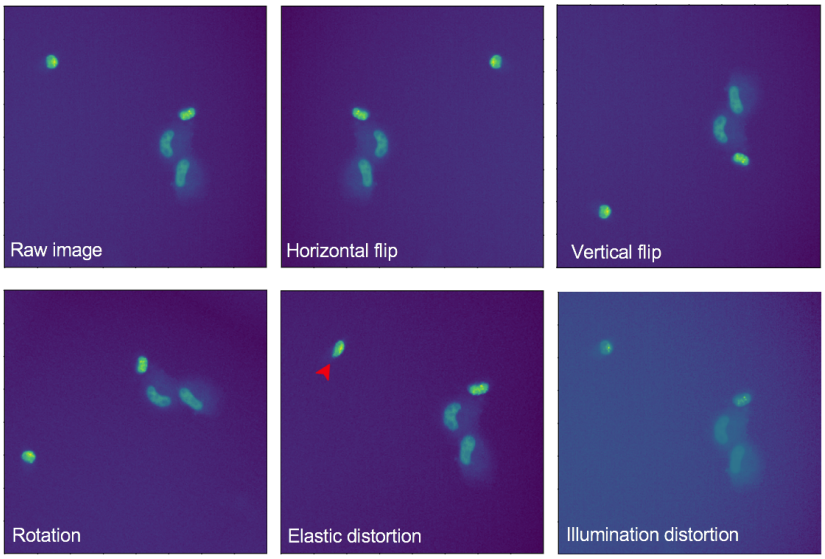


Figure 15: **Data augmentation techniques.** In each panel the raw image (top left) has been transformed with the labeled data augmentation method. The red arrow highlights an area of elastic distortion.

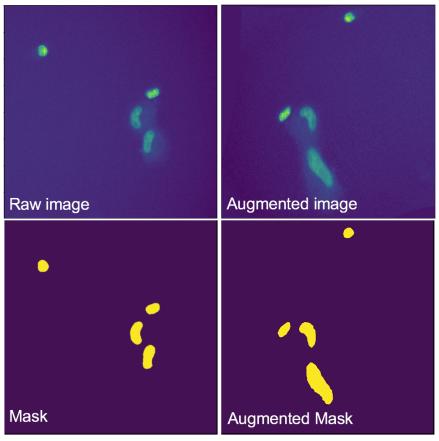


Figure 16: **Combined effect of the data augmentation techniques.** A raw image and its corresponding segmentation mask is transformed using multiple data augmentation techniques to generate a distinct image

This technique increases the size of the training data set from $\sim$ 250 raw images, to a theoretical maximum of 52,000 images.

5.2.3 Cell Crowding

A common segmentation error is the merging of two or more nearby nuclei, which results in analysis errors and loss of information. In the original U-Net publication, weight maps were used to direct training to areas of high cell density (Ronneberger, Fischer & Brox, 2015). The maps were calculated using millions of pixel to pixel distances, which was unfeasible with the computational resources available in this project. An algorithm was developed to be less computationally intensive, while still effectively highlighting the region between two nearby nuclei (see Figure 17).

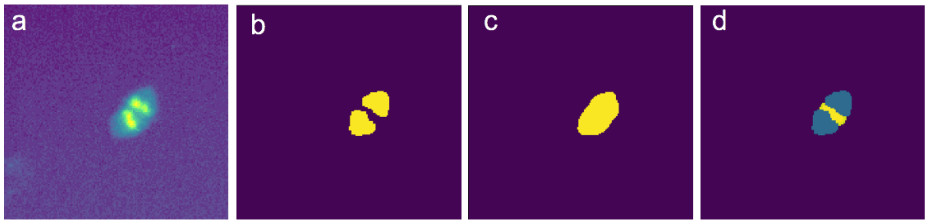


Figure 17: **Computation of the pixel-wise weight map.** (**a**) Raw image (**b**) Ground truth (**c**) Prediction (merge error) (**d**) Weight map. The weight map algorithm (see Materials and Methods; Software and Hardware) effectively highlights the region between the two nuclei. The weighting enforces training in this area against the merge error seen in panel c.

The effect of the weight maps was positive on segmentation quality, with improvements seen in created, absent and merge type errors (See Figure 18).

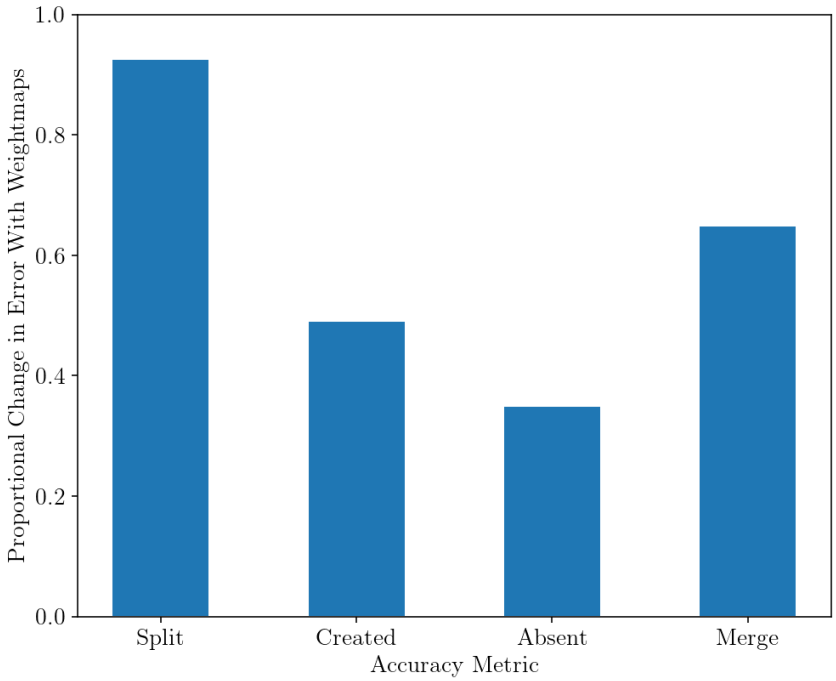


Figure 18: **The effect of weight maps on segmentation quality.** The proportional change of the four SCAM segmentation errors was evaluated between a U-Net trained without weight maps, against one with weight maps. There was little to no improvement seen in split type errors, while created, absent and merge type errors show on average a two fold improvement.

5.3 Comparison of U-Net and Watershed Segmentation

The most widely used non-machine learning image segmentation algorithm is Watershed, and is the basis for the segmentation software used in CCC research. On a typical CCC image, U-Net segmentation performs better than Watershed, and even identifies a nucleus that may have been erroneously omitted from the ground truth mask (see Figure 19).

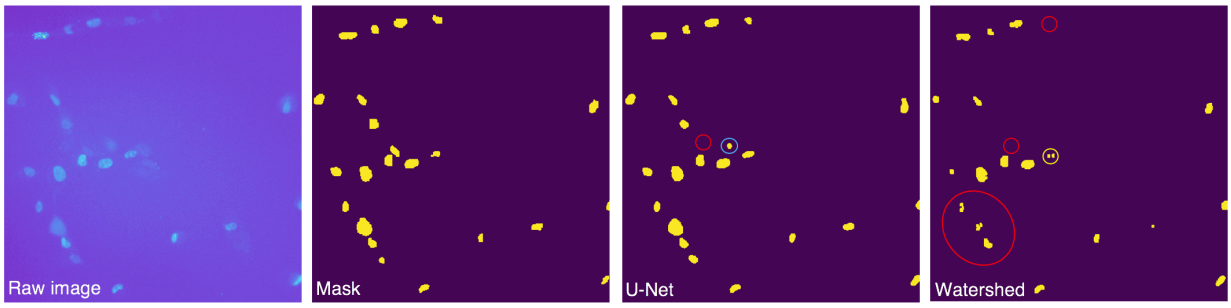


Figure 19: **Image segmentation by U-Net and Watershed.** In the U-Net prediction of the raw image, one nucleus is omitted (red circle), but another is segmented that may have been mistakenly omitted from the ground truth mask (blue circle). Several nuclei are omitted or poorly segmented from the Watershed segmentation (red circles), as well as split into multiple segments (yellow circle)

From this visual analysis, the U-Net appears to more accurately segment nuclei on the pixel-level, as well as commit fewer object-level segmentation errors. To more quantitatively compare the two, segmentation quality metrics were collected for both methods on a diverse range of validation images (see Figure 20). On the object-level, the U-Net scores higher on recall, indicating the omission of fewer true nuclei during segmentation. At the pixel-level, both methods score highly for precision, with the U-Net scoring slightly higher for recall, resulting in lower under-segmentation as expected from the visual example above (large red circle). For the SCAM segmentation errors, the U-Net performs significantly better on split, absent and merge errors, but 'creates' more nuclei than Watershed. This could be explained by the segmentation of true nuclei erroneously omitted from the ground truth segmentation mask as seen in Figure 19. High numbers of split errors by Watershed have been previously reported and are in agreement with the results of this analysis (Kornilov & Safonov, 2018).

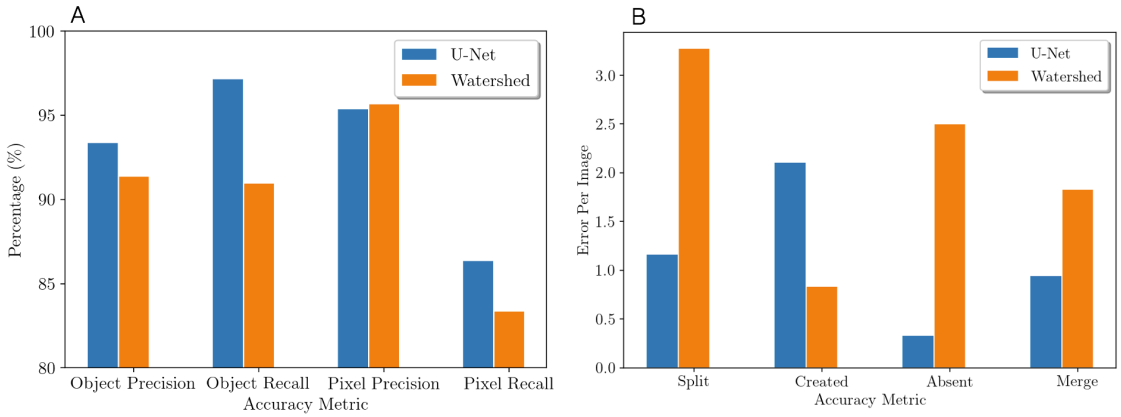


Figure 20: **Segmentation quality metrics for U-Net and Watershed. A)** Object and pixel-level precision and recall **B)** SCAM segmentation errors. In this analysis, the U-Net and Watershed algorithm were used to segment several validation images and their output compared to ground truth segmentation masks. Each metric is then calculated from this comparison. For example, Watershed committed on average ~ 3 splitting errors per image when compared to the ground truth mask, while the U-Net only committed ~ 1. The ideal segmentation method would score highly on object and pixel-level precision and recall, while committing as few as possible SCAM segmentation errors.

The Watershed algorithm requires the manual tuning of several parameters to achieve the best results on a given image. The parameter set used for the above comparison was chosen as the most generally effective on the set of images, rather than being optimised for each. This led to poorer performance overall, but was deemed a fairer comparison than

finding optimal parameter sets for each image. In terms of computation time, Watershed is significantly faster (<1s) than U-Net segmentation ($\sim$ 20s), but this does not include the time required to manually optimise Watershed parameters.

5.4 Proof of Concept

The segmentation and tracking U-Nets were combined into a pipeline to automate the full analysis process. This includes input data processing and U-Net segmentation and tracking, followed by single-cell metric extraction and output data processing. In Figure 21, a 10-frame window in which an isolated A549 nucleus undergoes mitosis demonstrates the performance of the pipeline on a simple, but typical CCC scenario. Nuclear area and fluorescent data was generated without user input, parameter optimisation, or manual error correction, and serves as a proof of concept for fully automated experimental analysis in CCC research. Naturally, as the U-Net pipeline is further developed, this validation can be extended to more difficult analysis scenarios such as areas of high cell density, overlap, and motility.

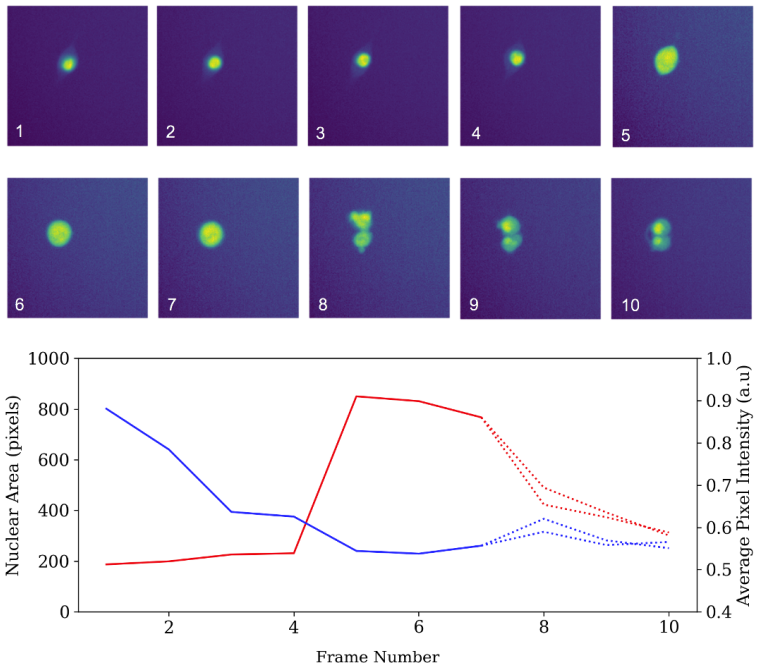


Figure 21: **Validation of the U-Net analysis pipeline.** A single A549 PCNA-labeled nucleus is segmented, tracked and analysed across a 10-frame window (100 minutes) in which it undergoes mitosis and generates a daughter nucleus. The trace of nuclear area (red) and intensity (blue) follow the outcome expected from the raw images. Nuclear area remains constant up to frame 4 and then increases $\sim$ 3-fold as the nucleus enters mitosis. This is followed by division into two nuclei, which are tracked and analysed as distinct objects (dotted lines). Fluorescence intensity declines as the nucleus nears mitosis (frames 1-5), and then remains relatively constant through mitosis and early G1 (frames 5-10).

6 Discussion

6.1 Over-Fitting

Large networks trained on small data sets are prone to over-fitting, where the network learns specific features of the training data rather than the general rules of the task. This can be avoided by reducing the size of the network, or increasing the size of the training data set. Reducing network size was not possible (5.2.1), but the size of the training data set was increased with U2OS images and data augmentation (5.2.2). Typically, the degree of over-fitting would then be assessed by a validation data set that contains images not used during training. This was implemented in the early stages of this project, but there was not sufficient data to create a large and robust validation data set. Cross-validation is an alternative method for monitoring over-fitting and popular in machine learning research (Stone, 1974). However, cross-validation requires the training of multiple networks with

different fractions of the training data, which was unfeasible in this project given that GPU computing was not available and network training times approached 1 week. GPU computing can reduce training times to $\sim$ 10 hours (Ronneberger, Fischer & Brox, 2015) and should be considered for future work in this field.

Given that over-fitting could not be monitored, and only partially avoided by increasing the size of the training data set, an additional technique known as drop-out was implemented. Drop-out randomly removes neurons during training to prevent co-dependencies (Srivastava et al., 2014) and has been used previously with U-Nets for biomedical image analysis (Ronneberger, Fischer & Brox, 2015; Lugagne, Lin & Dunlop, 2020). The amount of drop-out is set by a hyper-parameter in the range 0-1, which refers to the probability of retaining a given neuron in each round of training. DeLTA use a value of 0.5, which falls in the optimal range of 0.2-0.6 reported by the developers of the drop-out technique (Srivastava et al., 2014). However, Srivastava et al. report that a larger drop-out value can lead to under-fitting due to the loss of multiple neurons during training, and CCC U-Nets with fewer neurons were shown to be particularly prone to under-fitting (5.2.1). For this reason a lower drop-out probability of 0.3 was chosen to avoid under-fitting while still falling within the reported optimal drop-out range. Srivastava et al. also report that for 'extremely small' data sets of less than 500 images, drop-out has no positive effect on network performance. It is unclear whether this applies to our pre-augmentation data set size of $\sim$ 250 images, or the theoretically larger augmented data set. In future work, both U-Nets should be cross validated to ensure that drop-out is working as expected and preventing over-fitting on the small training data set.

6.2 Track-Net Re-design

The DeLTA tracking U-Net (see Figure 5) was modified to take only binary segmentation masks as inputs. It is not clear why the raw image and segmentation masks are provided as inputs in terms of the information required to effectively track nuclei. The modified tracking U-Net was able to perform tracking, and deal with mitosis (see Figure 21). In theory, tracking uniform-intensity binary objects is a simpler task than the equivalent using raw images that

contain noise and artefacts. One key area for future work is the development of a robust post-processing algorithm that converts tracking U-Net output to continuous cell trajectories. Because this was only partly achieved in this project, a quantitative comparison between the two U-Net designs was not possible.

6.3 Responsible Research and Innovation

The automation of data analysis in CCC research could have several positive impacts on the field. The removal of the analysis bottle-neck may allow experimental through-put to increase, generating more data more quickly. The introduction of deep learning to the field may encourage its use beyond data analysis, such as morphology based sorting for cell cycle synchronisation (Gu et al., 2019). There is also potential for researchers to take an active role in the implementation of the technology and develop transferable skills in computer programming. All of the above should contribute to advancement in the field, and therefore to the mission goal of the field itself, to better understand cell proliferation decisions and improve cancer therapeutics.

However, negative impacts may include less thoughtful experimental design, given that analysis would be less laborious and time consuming, as well as a disconnect between researchers and the analysis of their own data. This could be most serious if errors are mistakenly built into the analysis software because the developer is an outside specialist with no experience in the field of application. To avoid this, researchers in the field should be kept well informed of decisions made during the development of analysis software, as well as given the opportunity to learn and engage with the technology itself. The developer should maintain transparency with researchers, ensure that all source code is clearly annotated, and engage as much as possible within the field of application by attending lab meetings and seminars. In general, this broader issue can be avoided if interdisciplinary research is treated as collaboration, not outsourcing.

6.4 Conclusions and Future Work

In this project, the double U-Net analysis pipeline shown to be successful in the analysis of *E. coli* cells by DeLTA, has been extended to the free moving mammalian cell cultures imaged in CCC research. In doing so, two U-Nets were optimised, and the challenges presented by image data in CCC research addressed;

1. **Hyper-parameters**
   (a) The learning rate hyper-parameter used in the development of DeLTA (Lugagne, Lin & Dunlop, 2020) was shown to be suitable for this implementation and allow proper convergence of the U-Net during training (5.1.1).
   (b) The binary threshold hyper-parameter was optimised to 0.75 by balancing SCAM segmentation errors with pixel-level precision and recall (5.1.2)
2. **Training data**
   (a) The set of U2OS images used to enlarge the training data set was shown to improve A549 segmentation quality as well as performance on previously unseen image artefacts (5.2.2). This result will be useful to further machine learning research in biomedical image analysis where task-specific training data sets are unavailable or difficult to generate.
   (b) Data augmentation techniques were implemented for CCC images to further increase the size of the data set. These included complex elastic and illumination distortions (5.2.2)
3. **Weight Maps**
   (a) A computationally efficient algorithm was developed to generate pixel-wise weight maps which were shown to improve segmentation quality on three of the four SCAM errors (5.2.3).

Once optimised, the segmentation U-Net was compared to the Watershed method and shown to be more accurate in the majority of the segmentation metrics evaluated (5.3).

This quantitative comparison, along with several others in this project, was made using a calculation of precision and recall on both the object and pixel level. The algorithms developed for the automatic calculation of these metrics will be highly useful to further research in the area by contributing to the 'test' portion of the design-build-test-learn cycle central to synthetic biology. Finally, the resulting analysis pipeline was shown to be effective on a simple CCC research scenario and serves as a proof-of-concept on which to base further development.

In future work, the U-Net pipeline should be validated with the standardised data sets developed by Ljosa et al. (2012) and benchmarked with the evaluation metrics used by Tian et al. (2020) to fully contextualise the U-Net pipeline amongst other analysis techniques. Work should then focus on robust post-processing and visualisation scripts to develop the U-Net pipeline into a piece of fully automatic analysis software. As exemplified by NucliTrack (Cooper et al., 2017), user-friendliness should be prioritised to encourage adoption of the software tool amongst researchers, and fully unlock the benefits of automated image analysis for cancer cell cycle research.

7 Acknowledgements

I would like to give thanks to Dr Philipp Thomas for his guidance throughout this project and providing a unique opportunity for a Life Science student to undertake research in Machine Learning. I would also like to thank Dr Alexis Barr for welcoming me into her lab and providing me with the space, tools and data necessary to complete this project, as well as all members of the Barr Lab for their advice, time and useful discussions.

8 Bibliography

Barr, A.R., Cooper, S., Heldt, F.S., Butera, F., et al. (2017) DNA damage during S-phase mediates the proliferation quiescence decision in the subsequent G1 via p21 expression. Nature Communications. 8 (1), 14728. Available from: doi:10.1038/ncomms14728.

Caicedo, J.C., Cooper, S., Heigwer, F., Warchal, S., et al. (2017) Data-analysis strategies for image-based cell profiling. Nature Methods. 14 (9), 849–863. Available from: doi:10.1038/nmeth.4397.

Canny, J. (1986) A computational approach to edge detection. Ieee Transactions on Pattern Analysis and Machine Inligence.

Chalfoun, J., Cardone, A., Dima, A.A., Allen, D.P., et al. (2010) Overlap-Based Cell Tracker. Journal of Research of the National Institute of Standards and Technology. 115 (6), 477–486. Available from: doi:10.6028/jres.115.034.

Chen, W., Dong, J., Haiech, J., Kilhoffer, M.-C., et al. (2016) Cancer Stem Cell Quiescence and Plasticity as Major Challenges in Cancer Therapy. Stem Cells International. 2016. Available from: doi:10.1155/2016/1740936.

Cooper, S., Barr, A.R., Glen, R. & Bakal, C. (2017) NucliTrack: an integrated nuclei tracking application. Bioinformatics. 33 (20), 3320–3322. Available from: doi:10.1093/bioinformatics/btx404.

Dietterich, T. (1995) Overfitting and undercomputing in machine learning. ACM Computing Surveys (CSUR). 27 (3), 326–327. Available from: doi:10.1145/212094.212114.

El Allaoui, A. (2012) Medical Image Segmentation by MarkerControlled Watershed and Mathematical Morphology. The International journal of Multimedia & Its Applications. 4 (3), 1–9. Available from: doi:10.5121/ijma.2012.4301.

Gu, Y., Zhang, A.C., Han, Y., Li, J., et al. (2019) Machine Learning Based Real-Time Image-Guided Cell Sorting and Classification. Cytometry. Part A: The Journal of the International Society for Analytical Cytology. 95 (5), 499–509. Available from: doi:10.1002/cyto.a.23764.

He, K., Zhang, X., Ren, S. & Sun, J. (2015) Deep Residual Learning for Image Recognition. arXiv:1512.03385.

Heldt, F.S., Barr, A.R., Cooper, S., Bakal, C., et al. (2018) A comprehensive model for the proliferation–quiescence decision in response to endogenous DNA damage in human cells. Proceedings of the National Academy of Sciences. 115 (10), 2532–2537. Available from: doi:10.1073/pnas.1715345115.

Hsu, C.-H., Altschuler, S.J. & Wu, L.F. (2019) Patterns of Early p21 Dynamics Determine Proliferation-Senescence Cell Fate after Chemotherapy. Cell. 178 (2), 361-373.e12. Available from: doi:10.1016/j.cell.2019.05.041.

Jeknić, S., Kudo, T. & Covert, M.W. (2019) Techniques for Studying Decoding of Single Cell Dynamics. Frontiers in Immunology. 10. Available from: doi:10.3389/fimmu.2019.00755.

Kornilov, A. & Safonov, I. (2018) An Overview of Watershed Algorithm Implementations in Open Source Libraries. Journal of Imaging. 4 (10), 123. Available from: doi:10.3390/jimaging4100123.

LeCun, Y., Bengio, Y. & Hinton, G. (2015) Deep learning. Nature. 521 (7553), 436–444. Available from: doi:10.1038/nature14539.

LeCun, Y., Boser, B.E., Denker, J.S., Henderson, D., et al. (1990) Handwritten Digit Recognition with a Back-Propagation Network. In: D. S. Touretzky (ed.). Advances in Neural Information Processing Systems 2. Morgan-Kaufmann. pp. 396–404

Li, J., Sarma, K.V., Chung Ho, K., Gertych, A., et al. (2018) A Multi-scale U-Net for Semantic Segmentation of Histological Images from Radical Prostatectomies. AMIA Annual Symposium Proceedings. 2017, 1140–1148.

Ljosa, V., Sokolnicki, K.L. & Carpenter, A.E. (2012) Annotated high-throughput microscopy image sets for validation. Nature Methods. 9 (7), 637–637 Available from: doi:10.1038/nmeth.2083.

Lugagne, J.-B., Lin, H. & Dunlop, M.J. (2020) DeLTA: Automated cell segmentation, tracking, and lineage reconstruction using deep learning. PLOS Computational Biology. 16 (4), e1007673.
Available from: doi:10.1371/journal.pcbi.1007673.

Magnusson, K.E.G., Jalden, J., Gilbert, P.M. & Blau, H.M. (2015) Global linking of cell tracks using the Viterbi algorithm. IEEE transactions on medical imaging. 34 (4), 911–929. Available from: doi:10.1109/TMI.2014.2370951.

McGuinness, K. & O'Connor, N.E. (2008) The K-Space segmentation tool set.

Moen, E., Bannon, D., Kudo, T., Graf, W., et al. (2019) Deep learning for cellular image analysis. Nature Methods. 16 (12), 1233–1246. Available from: doi:10.1038/s41592-019-0403-1.

Oktay, O., Schlemper, J., Folgoc, L.L., Lee, M., et al. (2018) Attention U-Net: Learning Where to Look for the Pancreas. arXiv:1804.03999. Available from: http://arxiv.org/abs/1804.03999.

Quach, T. & Farooq, M. (1994) Maximum likelihood track formation with the Viterbi algorithm. In: Proceedings

of 1994 33rd IEEE Conference on Decision and Control. December 1994 pp. 271–276 vol.1. Available from: doi:10.1109/CDC.1994.410918.

Ronneberger, O., Fischer, P. & Brox, T. (2015) U-Net: Convolutional Networks for Biomedical Image Segmentation. arXiv:1505.04597. Available from: http://arxiv.org/abs/1505.04597.

Spiller, D.G., Wood, C.D., Rand, D.A. & White, M.R.H. (2010) Measurement of single-cell dynamics. Nature. 465 (7299), 736–745. Available from: doi:10.1038/nature09232.

Stewart-Ornstein, J. & Lahav, G. (2016) Dynamics of CDKN1A in single cells defined by an endogenous fluorescent tagging toolkit. Cell reports. 14 (7), 1800–1811. Available from: doi:10.1016/j.celrep.2016.01.045.

Stone, M. (1974) Cross-Validatory Choice and Assessment of Statistical Predictions. Journal of the Royal Statistical Society: Series B (Methodological). 36 (2), 111–133. Available from: doi:10.1111/j.2517-6161.1974.tb00994.x.

Tian, C., Yang, C. & Spencer, S.L. (2020) EllipTrack: A Global-Local Cell-Tracking Pipeline for 2D Fluorescence Time-Lapse Microscopy. Cell Reports. 32 (5). Available from: doi:10.1016/j.celrep.2020.107984 [Accessed: 7 August 2020].

Yamashita, R., Nishio, M., Do, R.K.G. & Togashi, K. (2018) Convolutional neural networks: an overview and application in radiology. Insights into Imaging. 9 (4), 611–629. Available from: doi:10.1007/s13244-018-0639-9.

Yao, G. (2014) Modelling mammalian cellular quiescence. Interface Focus. 4 (3), 20130074. Available from: doi:10.1098/rsfs.2013.0074.

Zagoruyko, S. & Komodakis, N. (2017) Wide Residual Networks. arXiv:1605.07146. Available from: http://arxiv.org/abs/1605.07146.

Zhang, C. & Xing, Y. (2018) CT artifact reduction via U-net CNN. In: Medical Imaging 2018: Image Processing. 5 March 2018 International Society for Optics and Photonics. p. 105741R. Available from: doi:10.1117/12.2293903.