The Eleventh International Conference on
Engineering Computational Technology
23–25 August, 2022 | Montpellier, France

Proceedings of the Eleventh International Conference on
Engineering Computational Technology
Edited by B.H.V. Topping and P. Iványi
Civil-Comp Conferences, Volume 2, Paper 6.2
Civil-Comp Press, Edinburgh, United Kingdom, 2022, doi: 10.4203/ccc.2.6.2
©Civil-Comp Ltd, Edinburgh, UK, 2022

Enhancing Lecture Capture with Deep Learning

R.M. Sales[1,2] and S. Giani[2]

**[1]Whittle Laboratory,
Cambridge University, Cambridge, United Kingdom**

**[2]Engineering Department, Durham University,
Durham, United Kingdom**

Abstract

This paper provides an insight into the development of a state-of-the-art video processing system to address limitations within Durham University's 'Encore' lecture capture solution. The aim of the research described in this paper is to digitally remove the persons presenting from the view of a whiteboard to provide students with a more effective online learning experience. This work enlists a 'human entity detection module', which uses a remodelled version of the Fast Segmentation Neural Network to perform efficient binary image segmentation, and a 'background restoration module', which introduces a novel procedure to retain only background pixels in consecutive video frames. The segmentation network is trained from the outset with a Tversky loss function on a dataset of images extracted from various Tik-Tok dance videos. The most effective training techniques are described in detail, and it is found that these produce asymptotic convergence to within 5% of the final loss in only 40 training epochs. A cross-validation study then concludes that a Tversky parameter of 0.9 is optimal for balancing recall and precision in the context of this work. Finally, it is demonstrated that the system successfully removes the human form from the view of the whiteboard in a real lecture video. Whilst the system is believed to have the potential for real-time usage, it is not possible to prove this owing to hardware limitations. In the conclusions, wider application of this work is also suggested.

Keywords: convolutional neural network,semantic image segmentation, binary human segmentation, learning rate optimisation, lecture capture technology

1 Introduction

In countries where internet access is almost universal, it has become common practice for higher-education institutions to record their lectures for students to access online. Whilst expensive software and/or hardware is often required to ensure that lecture content is recorded clearly, research generally shows that e-learning technologies are worthwhile and contribute positively to student learning outcomes [1]. At Durham University, the 'Encore' lecture capture system is currently only capable of filming and uploading a lecturer's commentary and visual-projector slides; as of yet, no white-board teaching is recorded. In certain disciplines where whiteboards remain the most natural means of conveying knowledge and information, *i.e.* the STEM subjects, this technological limitation could impact the overall quality of learning. If teaching staff have to adopt less effective modes of delivery just to accommodate recording, then the sensible response would be to upgrade the 'Encore' system to capture whiteboards too. But why stop there? With the accelerated transition towards web-based learning, brought about in reaction to the Covid-19 pandemic [2], there has never been a greater need for advanced e-learning technologies. These circumstances provide the opportunity to further enhance the 'Encore' online lecture experience by creating a system that provides students with a clear and unobstructed view of the whiteboard at all times.

In response to this opportunity, the objective of this work was to develop a real-time video processing system that could be used as part of the lecture capture process to digitally remove the persons presenting (the foreground) from the view of a white-board (the background). To achieve this, it was necessary to divide the task into two more clearly defined sub-tasks: first, to develop a system that can detect the location of human beings within a frame of video and; second, to develop a system to restore any whiteboard content that is currently obscured. These were most easily managed with two separate computational modules working in tandem.

Module I: Human Entity Detection

The **human entity detection module** was introduced to predict which regions of a video frame are most likely to contain human beings. There are, of course, a number of traditional computer vision algorithms that could have been adopted to tackle this simple-sounding problem without the need for human supervision [3–6]. However, many have been rendered obsolete as a result of breakthrough discoveries in artificial intelligence in the last decade [7–11]. The most compelling state-of-the-art techniques now include variations of object detection and image segmentation, both of which continue to show increasingly promising results when implemented using deep convolutional neural network (DCNN) architectures [12]. Whilst there is still considerable room for improvement in terms of effectiveness and efficiency, the most recent architectural contributions have demonstrated that real-time detection and segmentation techniques are now fit for application even on modest hardware [13]. Primarily because the latter offered greater predictive exactness at only a fractionally higher cost, the approach adopted in this work was to develop a deep-learning-based binary semantic segmentation model.

Module II: Background Restoration

The **background restoration module** was introduced to digitally reconstruct the regions of the whiteboard that are identified by the human entity detection module as being obscured. In essence, this makes it possible to produce video footage where all teaching staff appear to be removed, provided they have been completely identified during segmentation. Instead of using complex artificial intelligence techniques that generate synthetic imagery to replace spatio-temporal holes in video frames, as in [14] and [15], a straightforward algorithm was developed to recover the actual background (the whiteboard) as it was last observed. In the context of recording a presenter's writing on a whiteboard in real-time, this approach is superior in terms of both accuracy and computational expense: factors that are deemed far more important than temporal coherence or aesthetics in this work.

In the remainder of this paper, Section 2 presents the theory and methods used to develop the human entity detection and the background restoration modules; Section 3 gives a comprehensive account of the experimental procedure used in training and testing; Section 4 presents and comments on the most relevant results, comparing these to the state-of-the-art; while the Conclusions also suggest potential future work.

2 Theory and Methods

2.1 Human Entity Detection

The human-entity detection module (Figure 1) is essentially a remodelled version of the state-of-the-art Fast Segmentation Neural Network (Fast-SCNN) developed by Poudel *et al.* [16]. This network was chosen as a starting point for several reasons: it is designed specifically for faster than real-time semantic image segmentation; its capabilities extend well to high-resolution images; and with only 1.11 million parameters, it is well suited for deployment on general-purpose computer hardware. In testing, the original Fast-SCNN has been shown to achieve an astonishing 68% mean average precision (mAP) at 123.5 frames per second when performing segmentation at full resolution (2048×1024px) on the CityScapes benchmark server [17]. Although newer and potentially more advanced models have since been introduced, this three-year-old network remains highly competitive in terms of accuracy, runtime and computational cost. This enduring high level of performance can largely be attributed to the very efficient arrangement of network modules and the authors' intentional use of low-memory components.

Fast-SCNN is a fully-convolutional neural network that embeds a deep two-branch

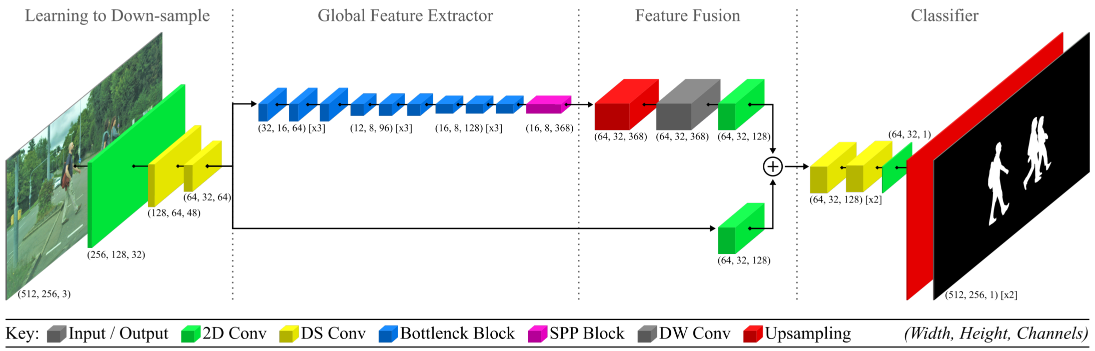

Figure 1: The human entity detection module is a remodelled version of Fast-SCNN [16] designed specifically for real-time binary semantic image segmentation. The tensor dimensions show the remodelled network configured at 1/4 resoltion (512×256px).

structure centrally within an encoder-decoder model. Poudel *et al.* group this architecture into four back-to-back modules which, from input to output, include: a novel 'learning to downsample' (LTD) module, a global feature extractor (GFE), a feature fusion module (FFM), and a classifier.

2.1.1 Remodelling Fast-SCNN

The original Fast-SCNN model achieves faster than real-time performance on an Nvidia Titan XP card with 12GB of GDDR5 memory. However, a number of modifications must be made to prepare it for real-time segmentation on general-purpose hardware. If it were configured at full-resolution on a lesser GPU, the concern is that Fast-SCNN would take about two days to train and would be incapable of making real-time predictions. The solution was simply to trade detail for efficiency and reconfigure the model to accept 1/4 resolution inputs. At a reduced size of 512×256px, each image demands only 1/16th of its original memory footprint and thus inference becomes quicker and training with large batches becomes accessible.

A change of size would ordinarily pose no issue for a FCN. However, Fast-SCNN uses a spatial pyramid pooling (SPP) block with fixed bin sizes. The original authors provide no information on how to implement this at any resolution, and so this work elects to build a new and improved SPP block based on the 4-layer pyramid of Zhao *et al.* [20]. This new block employs 2D average pooling with variable stride and dynamic pool-size as a means to aggregate gridded subregions of the input feature tensor. Four parallel layers with bins arranged in 1^2, 2^2, 4^2, and 8^2 grids divide context information in both spatial dimensions; a pointwise convolutional layer then condenses the number of channels of each bin by a factor of its spatial size. Having weighted each bin equally in terms of parameters, the features are then upsampled with bilinear interpolation and concatenated with the original SPP input.

The final change was the addition of dropout between the last two depthwise separable layers so as to prevent co-adaptation of features in the classifier. The dropout parameter was set to a value of 0.3 during training to provide regularisation and protect against overfitting [21]. In line with the original Fast-SCNN, the remodelled version is very lightweight with just over 1.16 million trainable parameters.

2.2 Background Restoration

From a 'black-box' perspective the background restoration module sequentially accepts a stream of two inputs, x_t and x_t^*, from which it generates one stream of outputs y_t. An additional internal variable also exists with the sole purpose of retaining only the most recent output, y_{t-1}, once it is brought into existence. The inputs include a three-channel RGB image, received directly from the camera, and its corresponding single-channel grayscale segmentation bitmap, as predicted by the detection module. The restoration module computes using information from both inputs, as well as the internal variable, and outputs a single three-channel image, for which the background has been reconstructed, to the real world. The flow of information through this computational module is shown diagrammatically in Figure 2. As this system operates on video, *i.e.* constant streams of input images and bitmaps, this procedure is simply repeated for each successive frame of footage.

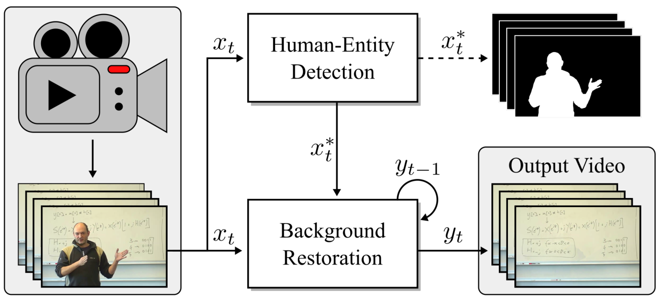

Figure 2: A black-box depiction of the flow of video frame information via the two parallel computational modules (photograph of Dr C. Balocco kindly provided by Dr S. Giani.)

The restoration algorithm works by updating each pixel of the internal variable y_{t-1} to its most recently captured state in x_t, provided that pixel does not fall within the obscured region defined in x_t^*. To be clear, the internal variable is an image data structure containing only the most recently unobstructed pixel values from the set of input images, X_t, where it is possible. Once this internal variable has been completely updated, it is output to the real world as y_t. The result of multiple inputs is a sequence of video still-image frames where all regions containing human entities are replaced by the most up-to-date version of the background they are now obscuring. Conceptually, this is akin to an 'invisibility cloak'. When initialized, the background restoration module produces an output frame, y_0, that is identical to the original input image, x_0: since the system begins with no prior knowledge as to the view behind any obstructions, it must output what it 'sees'. Eventually, as the human entities move within, or even out of the frame, the whole background is revealed and the module can output a frame containing no obstructions at all. While the camera remains stationary, which for most lecture-capture settings it will, this method should produce no distortion.

The fact that images are digitally expressed as tensors with real-integer entries makes them particularly well suited for linear algebraic transformations and bitwise boolean operations. In this work, input image pixels have standard 8-bit unsigned integer values ranging from 0 to 255 inclusive and the binary masks contain binary values {0,1} scaled up to 8-bits: {0,255}. Computationally, this makes the procedure of updating pixels relatively straightforward. To begin, a bitwise NOT operator is applied to the segmentation mask in order to generate a temporarily inverted version where the foreground and background pixel values are switched. A bitwise AND operator then applies this inverted mask to all three channels of the RGB input. This produces a modified version of the input where the values of all pixels containing persons are removed (zero) but those containing the background are unaltered. An identical process takes place with the original mask and the previous output frame to produce an image where only the 'old' pixels in the current region containing persons are preserved. As the disjoint union of the non-zero regions of these images partition the domain, these need only be summed in a bitwise fashion to produce the new output frame. The following pseudocode block, developed in this project, summarises the restoration algorithm more formally.

Algorithm 1: The background restoration procedure.

```
Input  : An image, $x_t$, and its predicted mask $x_t^*$.
Output: An image, $y_t$, with its background restored.
for $(x_t, x_t^*) \in \{(x_0, x_0^*), (x_1, x_1^*), \ldots, (x_n, x_n^*)\}$ do
    if $t = 0$ then
        $y_t \leftarrow x_t$
    else
        $y_t \leftarrow$ UpdateFrame($x_t$, $x_t^*$, $y_{t-1}$)
    end
    return $y_t$
end
Function UpdateFrame($x_t$, $x_t^*$, $y_{t-1}$):
    return (($x_t$ AND NOT $x_t^*$) ADD ($y_{t-1}$ AND $x_t^*$))
end
```

To aid understanding, the restoration procedure for a single iteration is depicted in diagrammatic form in Figure 3.

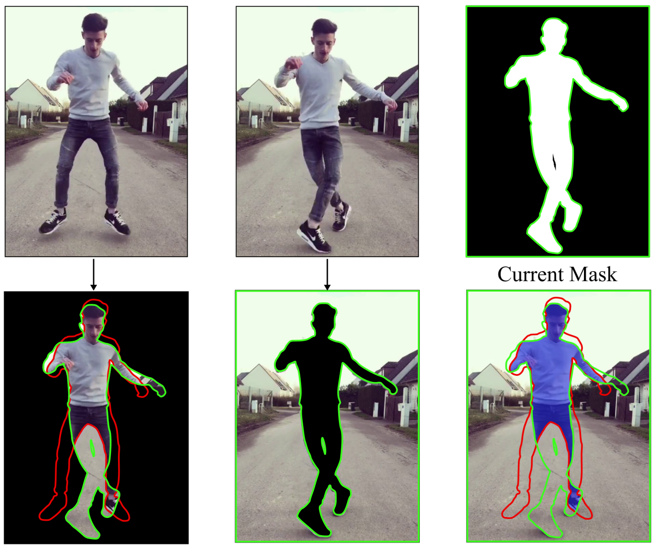

Figure 3: The first cycle of the background restoration procedure: (left) the current mask applied to the previous frame foreground, (centre) the current mask inverse applied to the current frame background, and (right) the two combined. The green and red outlines show the dancer's position in the current and previous frames respectively. What remains of the person after one cycle is in blue. Original images from Kaggle [22].

3 Experimental Procedure

This section addresses three technical points concerning the experimental procedure: the methods of training; details of the chosen loss function; and the dataset. Details regarding the feasibility of transfer learning and the full details of implementation can be found in [23].

3.1 Training

An empirical technique proposed by Smith [24] was used ahead of training to quickly estimate reasonable lower and upper bounds for the learning rate hyperparameter. This involved measuring batch loss metrics for several hundred training batches whilst monotonically increasing the learning rate from small to large values. The optimal learning rate range was then obtained by plotting loss vs. the logarithm of learning rate and identifying the region of greatest negative slope, *i.e.* the steepest consistent drop in loss. This investigation revealed that learning rates at or above 1×10^{-5} would ensure reasonable training times and that rates at or below an upper bound of 2×10^{-1} would avoid divergence in most circumstances. It is important to note that these findings will vary considerably depending on the choice of network, loss function, dataset, and batch size and are by no means a guideline for other studies.

Following the recommendations of Poudel *et al.* [16], the Fast-SCNN based human-entity detection module was trained using stochastic gradient descent (SGD) with a momentum parameter of 0.9 for a total of 160 epochs. All experiments were run at quarter-resolution with a batch size of 32. A learning rate scheduler was added and used throughout training to implement a technique known as 'learning rate annealing'. This involved progressively decaying the learning rate between the above-stated upper and lower bounds according to a degree-two polynomial function. The intuition behind this was to transition from large update steps that quickly but coarsely improve the initial parameter values to smaller update steps that slowly converge towards locally optimal values with much more exactness. Despite its relative simplicity compared to adaptive or cyclical learning rate policies, in practice, annealing is found to be one of the most reliable and effective means of obtaining faster asymptotic convergence at no cost to test performance.

3.2 Loss Function

Nearly all loss functions used in binary segmentation quantify the agreement between predicted segmentation masks and their ideally expected results, *i.e.* the ground truths, by considering each pixel prediction as an independent binary classification [25]. As a result, many contain mathematical terms that reflect the relative pixel-based frequency of each of the four binary predictive outcomes: true positive, true negative, false positive, and false negative. In the context of this work, false negative predictions are highly undesirable since these result in contamination of the output. Clearly, it is a benefit to train using a loss function that allows the user to scale the importance of each outcome individually. With this in mind, the human-entity detection module was optimized using a parametric loss function based on the popular Tversky Index [26]:

$$TI = \frac{\sum y\hat{y}}{\sum y\hat{y} + \sum \beta y(1-\hat{y}) + \sum (1-\beta)(1-y)\hat{y}} \tag{1}$$

where y is a normalised true pixel value $\{0, 1\}$, $\hat{y}$ is a normalised predicted pixel value $[0, 1]$, and β is an adjustable weight coefficient $[0, 1]$. The overall loss metric for each iteration is simply computed by averaging the complement of the aforementioned Tversky Index over the current training batch. A series of experiments on the validation set revealed that setting $\beta = 0.9$ leads to the desired balance between precision and recall. Full details of hyperparameter tuning, as well as a complete summary of the most relevant training details can be found in [23].

3.3 The dataset

The segmentation network was trained on a dataset of 2,615 finely-annotated high-resolution images extracted from a number of anonymised Tik-Tok dance videos. The raw images were sourced from a publicly available Kaggle archive [22] under a creative commons license and, in general, depict at least one entirely unobscured individual in a complex or unique dance pose. At a glance, there appears to be a good balance of indoor and outdoor scenes, containing both male and female individuals in bright and dark lighting conditions. The segmentation masks also contain a good distribution of black and white pixels with neither overpowering the other. To make the dataset usable, each image-mask pair was standardised and then partitioned into one of the training, testing or validation sets according to an 80:10:10 split. Standardisation included the resizing of each image-mask pair to fit within a multiple of the network's input resolution, namely 1024×512px, followed by some horizontal or vertical translation. To prevent distortion, a common scaling factor was applied in both directions and zero-padding was used to bulk out the surplus area. To discourage the SPP layer from favouring central pixel sub-regions, random samples from a uniform distribution were used to determine the extent of each translation. Both interventions aim to maximise the model's predictive ability over a wider range of use cases.

It is important to note that each image is a sample extracted from a relatively small set of videos. Clearly, this will mean that multiple images share broadly the same lighting, scenery, background, and camera perspective. To combat this homogeneity and thus avoid overfitting during training, extensive data augmentation techniques were employed [27]. A mix of geometric transformations (translations, rotations, zooms, shears, horizontal flips) and colour augmentations (brightness scaling) were used to artificially increase the size and overall quality of the training dataset. For ease of reference, the exact augmentation details are summarised in Table 1.

Augmentation	Details / sample bounds
Rotation	± 5 degrees in either direction
Zoom	40% enlarged $\rightarrow$ 100% unchanged
Height shift	± 20 percent image height
Shear	± 20 percent in any direction
Brightness	40% as bright $\rightarrow$ 50% brighter
Horizontal flip	randomly applied

Table 1: Reference Tik-Tok dataset augmentation details.

4 Results and Discussion

4.1 Training Analysis

Training at one-quarter resolution on a Google Colab cloud-based virtual machine took between five and six hours, with each epoch demanding approximately two minutes of computation. Multiple training sessions were needed to fine-tune the hyperparameter values. Figure 4 presents the training and testing curves for the loss, recall and precision metrics corresponding to the best performing training session (see 4.2). In this case, the hyperparameters and dataset augmentations were set according to [23] and Table 1 respectively, and the Tversky loss parameter was 0.9 throughout.

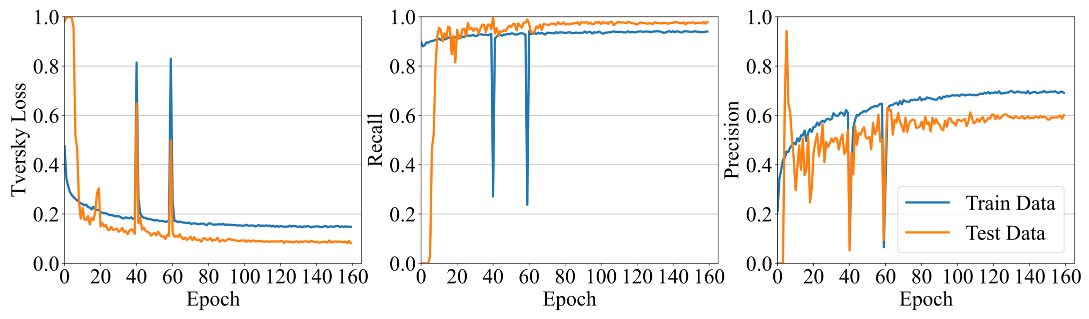

Figure 4: Training and testing curves of loss (left), recall (center) and precision (right) for the best performing training run.

The training metrics consistently followed characteristic learning trajectories as the model became increasingly exposed to the segmentation task. By the end of the first training epoch, the Tversky loss had already improved from a value of 1 to a value of approximately 0.47. In the same timeframe, the recall and precision metrics increased from 0 to about 0.90 and 0.21 respectively. Within 40 epochs, each metric had settled to within 5% of its final value. This unusually fast convergence is largely attributed to the efforts made in finding optimal upper and lower bounds for the learning rate hyperparameter, as well as the adoption of an effective polynomial learning rate annealing strategy. Apart from the anomalous spikes in loss at epochs 40 and 59, the training curve shows no sign of divergence; and instead converges asymptotically until training ends. This asymptotic nature indicates that the model was incapable of further learning and confirms that 160 epochs were sufficient to avoid underfitting. The key points of note are the usefulness of Smith's method [24] in determining an optimal range of learning rates, and the importance of a learning rate scheduler.

None of the test metrics showed any sign of improvement for the first four training epochs, but all increased to surpass their training counterparts by the end of epoch 9. This early 'lag' was observed at the start of all sessions and would typically worsen at decreased initial learning rates. In the absence of any similar results in the literature, it is hypothesised that this phenomenon is connected to the high data variance and relatively low number of images in the test dataset. From epoch 9 onwards, both test loss and recall remained high relative to training, whereas test precision fell and remained below training precision. Generally, the test curves follow the same trajectory as the training curves but with a small and broadly constant generalisation gap. For both loss and recall, this gap should not be mistaken with overfitting. Data augmentations were intentionally applied only to the training dataset to allow for fair comparison between augmentation settings. As a corollary, the model performed well on test images with better per-epoch loss on the testing dataset. If the model were overfitting on these two metrics, this gap would be positive, *i.e.* the test loss and recall would be worse than the training loss and recall, but this was not the case. There is, however, evidence to suggest that the model overfits in terms of precision, as can be seen by the positive generalisation gap in the lower subfigure of Figure 4. Considering that the Tversky loss function parameter (β) was set to a value of 0.9 for this particular training session, *i.e.* the objective was to favour recall over precision, this seems perfectly reasonable. These results demonstrate that the segmentation model trades generalisation in terms of precision for generalisation in terms of recall, thus highlighting the importance of the loss function in conveying the training objective.

4.2 Accuracy Analysis

The sole purpose of hyperparameter fine-tuning is to find the hyperparameter values that yield the most optimal system, either directly through training or indirectly through transfer learning. In this work, the main aim was to identify a value for the Tversky loss parameter that would result in a model best suited to the requirements of the human entity detection module. Of primary interest was the balance point between recall and precision. Figure 5 shows the validation metrics for a family of nine training experiments in which the Tversky loss parameter was iterated through the set $\{0.1, 0.2, 0.3, \ldots, 0.9\}$. In each of these experiments, the human entity detection module was trained from the outset for the entire 160 epoch duration, with all other hyperparameters fixed. Each of the segmentation models were then evaluated on the same validation dataset with no data augmentations.

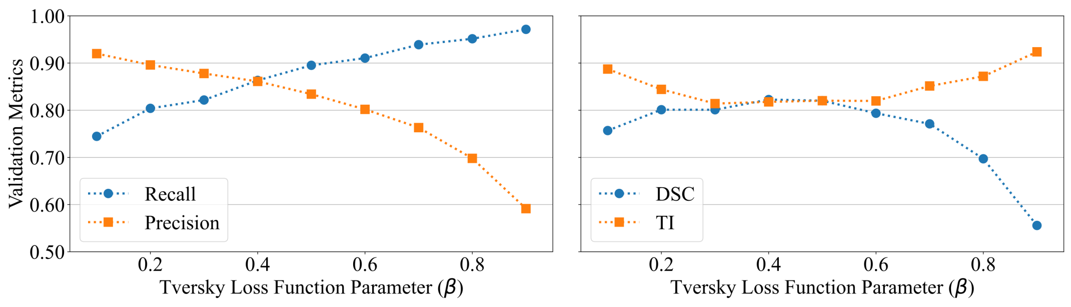

Figure 5: A cross-validation study on the effect of varying the Tversky loss function parameter (β) between 0.1 and 0.9.

The validation results exemplify the nature of the recall-precision tradeoff and clearly demonstrate that it would not be possible to train the segmentation model to maximise both metrics simultaneously. Since the recall curve exhibits monotonic increasing behaviour and the precision curve exhibits monotonic decreasing behaviour, both as functions of the Tversky loss parameter, there is no increment in which these improve together. The recall and precision metrics intersect at $\beta \approx 0.4$, *i.e.* slightly to the left of centre on the x-axis, which indicates that either the model or the dataset is predisposed to generate more false positives than false negatives. As anticipated, training with lower values of β leads to models with higher precision than recall, while training with higher values of β leads to models with higher recall than precision. The maximum values of recall and precision were found to be 0.971 and 0.920 respectively. Although not included in Figure 5, values at either of the extremes were found to encourage fully-saturated outputs: for $\beta = 0$, the model learned to predict all pixels as false, while for $\beta = 1$, the model learned to predict all pixels as true. These findings embody the issue of saturation raised in [28] since extreme parameter values lead to loss functions that depend on precision or recall, but not both.

The Dice Similarity Coefficient (DSC) and Tversky Index (TI) amalgamate recall and precision to allow for model comparison using a single representative value. The difference between the metrics is that unlike the DSC, the TI weights false positive and false negative outcomes relative to the Tversky loss parameter. Even with these metrics, it is not clear which value of the loss parameter is best. The DSC achieved a maximum value of 0.861 when $\beta = 0.4$, and decreased continuously towards each extreme, thus forming a peak-shaped profile. This decrease is noticeably steeper for higher values of β as precision is lost increasingly on that side of the domain: a minimum DSC of 0.728 was achieved when $\beta = 0.9$. In contrast, the TI achieved a minimum value of 0.857 when $\beta = 0.3$, and increased to higher values in both directions, thus forming a valley-shaped profile. This increase is noticeably steeper for higher values of β as the recall is highest on that side of the domain and precision contributes very little to the TI: a maximum value of 0.912 was achieved when $\beta = 0.9$. Since the TI and DSC separate increasingly with higher recall, and given that higher recall is beneficial in this work, the DSC proves to be unhelpful.

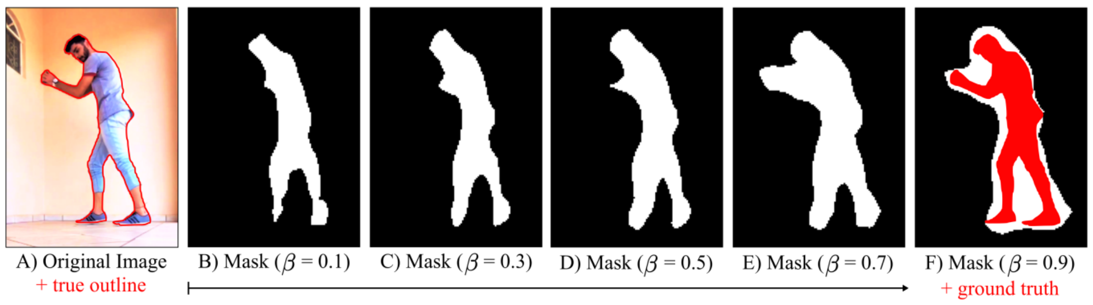

Figure 6: Binary segmentation masks for various settings of the Tversky loss parameter (β). Original image from Kaggle [22].

By choosing to ignore the DSC entirely, a Tversky loss parameter of 0.9 is arguably most appropriate for the human entity detection module. Figure 6 presents segmentation masks for five trained models to allow for visual comparison and to justify trusting the TI but not the DSC. The models are observed to predict human pixels more 'confidently' as β is increased. Given that only the segmentation mask corresponding to $\beta = 0.9$ fully encapsulates the ground truth, it is concluded that 0.9 is the best choice of Tversky loss parameter.

5 Conclusions

In the introduction to this paper, the separate roles of the “human entity detection” and “background restoration” modules were defined. In Section 2, the Fast Segmentation Neural Network [16] was then described. This model was adapted to make its application to real-time image segmentation more computationally efficient. Following this, Section 3 provides an account of the most effective training techniques as well as any relevant implementation details. Lastly, Section 4 presents and critically assesses the key experimental results. Overall, the research project described in this paper has demonstrated the following:

- A system capable of removing presenters entirely from the view of a whiteboard in lecture videos. This system is demonstrated in almost (one-third) real-time.
- The importance of selecting a loss function that can be calibrated to a specific learning objective. In this work, the Tversky loss parameter (β) was successfully used to fine-tune recall and precision for the desired effect.
- The benefits of Smith’s method [24] in locating the optimal learning rates prior to training, then applying these via a learning rate scheduler. Convergence to within 5% was observed in 40 epochs as a result of these efforts.
- A novel procedure, using an algorithm expressed in pseudocode form (Algorithm 1), that combines and updates video frame pixels according to binary masks predicted by an artificial intelligence segmentation model.

Future work should consider quantisation of the model to reduce the GPU load to an acceptable level. Further testing would then be required before introducing the system to Durham University’s ‘Encore’ system. In other fields, this work may offer development opportunities in relation to wider education/presentation systems, as well as any system where unwanted elements need to be removed from video images.

References

[1] B. Robertson and M. J. Flowers, “Determining the impact of lecture videos on student outcomes,” *Learning and Teaching*, vol. 13, no. 2, pp. 25–40, Jun. 2020.

[2] J. Louis-Jean and K. Cenat, “Beyond the Face-to-Face Learning: A Contextual Analysis,” *Pedagogical Research*, vol. 5, no. 4, p. em0077, Aug. 2020.

[3] R. Nock and F. Nielsen, “Statistical Region Merging,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, pp. 1452–1458, Nov. 2004.

[4] W. Tao, H. Jin, and Y. Zhang, “Color Image Segmentation Based on Mean Shift and Normalized Cuts,” *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 5, pp. 1382–1389, Oct. 2007.

[5] N. Plath, M. Toussaint, and S. Nakajima, “Multi-Class Image Segmentation Using Conditional Random Fields and Global Classification,” in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM Press, 2009, pp. 817–824.

[6] C. Zhao, “Image Segmentation Based on Fast Normalized Cut,” *The Open Cybernetics & Systemics Journal*, vol. 9, pp. 28–31, Feb. 2015.

[7] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” in *3rd International Conference on Learning Representations, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1409.1556

[8] H. Noh, S. Hong, and B. Han, “Learning Deconvolution Network for Semantic Segmentation,” in *2015 IEEE International Conference on Computer Vision*. IEEE Computer Society, 2015, pp. 1520–1528.

[9] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” in *18th International Conference of Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015, pp. 234–241. [Online]. Available: https://doi.org/10.1007/978-3-319-24574-4_28

[10] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, “ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation,” *CoRR*, vol. abs/1606.02147, 2016. [Online]. Available: http://arxiv.org/abs/1606.02147

[11] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, “Image Segmentation Using Deep Learning: A Survey,” *CoRR*, vol. abs/2001.05566, 2020. [Online]. Available: https://arxiv.org/abs/2001.05566

[12] L. Liu, W. Ouyang, X. Wang, P. W. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, “Deep Learning for Generic Object Detection: A Survey,” *International Journal of Computer Vision*, vol. 128, no. 2, pp. 261–318, Oct. 2020.

[13] G. Takos, “A Survey on Deep Learning Methods for Semantic Image Segmentation in Real-Time,” *CoRR*, vol. abs/2009.12942, 2020. [Online]. Available: https://arxiv.org/abs/2009.12942

[14] A. Newson, A. Almansa, M. Fradet, Y. Gousseau, and P. Pérez, “Towards Fast, Generic Video Inpainting,” in *Proceedings of the 10th European Conference on Visual Media Production*. ACM Press, 2013, pp. 1–8.

[15] D. Kim, S. Woo, J.-Y. Lee, and I. S. Kweon, “Deep Video Inpainting,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, Jun. 2019, pp. 5785–5794.

[16] R. P. K. Poudel, S. Liwicki, and R. Cipolla, “Fast-SCNN: Fast Semantic Segmentation Network,” in *30th British Machine Vision Conference*. BMVA Press, 2019, p. 289. [Online]. Available: https://bmvc2019.org/wp-content/uploads/papers/0959-paper.pdf

[17] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The Cityscapes Dataset for Semantic Urban Scene Understanding,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Jun. 2016, p. 3213.

[18] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "BiSeNet: Bilateral Segmentation Network for Real-Time Semantic Segmentation," in *15th European Conference of Computer Vision, Proceedings, Part XIII*, ser. Lecture Notes in Computer Science, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., vol. 11217. Springer, 2018, pp. 334–349. [Online]. Available: https://doi.org/10.1007/978-3-030-01261-8_20

[19] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *2018 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, Jun. 2018, pp. 4510–4520.

[20] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid Scene Parsing Network," in *2017 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, Jul. 2017, pp. 6230–6239.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[22] Segmentation Full Body TikTok Dancing Dataset. [Online]. Available: https://www.kaggle.com/tapakah68/segmentation-full-body-tiktok-dancing-dataset.

[23] R. M. Sales and S. Giani, "Enhancing Lecture Capture With Deep Learning," in *Advances in Engineering Software*. Elsevier, Aug. 2022.

[24] L. N. Smith, "Cyclical Learning Rates for Training Neural Networks," in *2017 IEEE Winter Conference on Applications of Computer Vision*. IEEE, Mar. 2017, p. 464.

[25] S. Jadon, "A survey of loss functions for semantic segmentation," in *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology*. IEEE, Oct. 2020, pp. 1–7.

[26] M. Yeung, E. Sala, C.-B. Schönlieb, and L. Rundo, "A Mixed Focal Loss Function for Handling Class Imbalanced Medical Image Segmentation," *arXiv e-prints*, Feb. 2021. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2021arXiv210204525Y

[27] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, Jul. 2019.

[28] S. R. Hashemi, S. S. M. Salehi, D. Erdogmus, S. P. Prabhu, S. K. Warfield, and A. Gholipour, "Asymmetric Loss Functions and Deep Densely-Connected Networks for Highly-Imbalanced Medical Image Segmentation: Application to Multiple Sclerosis Lesion Detection," *IEEE Access*, vol. 7, pp. 1721–1735, 2019.