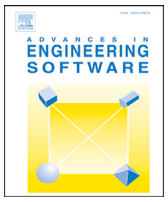
Advances in Engineering Software 196 (2024) 103732

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft



Enhancing lecture capture with deep learning

R.M. Sales [a,b,*], S. Giani [b]

[a] Whittle Laboratory, Cambridge University, Cambridge, United Kingdom
[b] Engineering Department, Durham University, Durham, United Kingdom

ARTICLE INFO

Keywords:
Convolutional neural network
Semantic image segmentation
Binary human segmentation
Learning rate optimisation
Lecture capture technology

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.

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 whiteboard 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 whiteboard (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, several conventional computer vision algorithms [3–6] that could potentially handle this seemingly straightforward task without human supervision. Yet, despite their generally lower complexity, these have been rendered obsolete due to breakthrough discoveries in artificial intelligence (AI) over the last

* Corresponding author at: Whittle Laboratory, Cambridge University, Cambridge, United Kingdom.
E-mail address: rms221@cam.ac.uk (R.M. Sales).

https://doi.org/10.1016/j.advengsoft.2024.103732
Received 11 May 2023; Received in revised form 7 July 2024; Accepted 8 July 2024
Available online 29 July 2024
0965-9978/© 2024 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

decade [7–11], as the latter typically offer superior performance in both accuracy and reliability. 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 than object detection at only a fractionally higher computational 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 [14,15], 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.

Semantic segmentation and related work

Semantic segmentation is the task of assigning a label from a set of categories, *e.g.* {'human', 'car', 'tree', 'background'}, to each pixel of an input image. Sometimes referred to as "dense prediction", this is distinct from object detection and instance segmentation since: (*i*) it works at the individual pixel level rather than predicting bounding boxes [16]; and (*ii*) it does not differentiate between multiple instances of the same category. By working at the level of pixels, the segmentation process preserves two-dimensional spatial information such as the input width and height. It is therefore convenient to store the output predictions as a segmentation mask: a bitmap image in which the pixel values represent the predicted labels of the output in some meaningful way. In the specific case of binary segmentation, the number of recognised categories is reduced to include only 'true' or 'false', which this work uses interchangeably to represent 'human' and 'background' respectively. To allow for comparison, Fig. 1 presents an image from the Tik-Tok image dataset [17] alongside its multi-class (multi-coloured) and binary (black-and-white) segmentation masks. Whilst the underlying mechanisms are identical, multi-class segmentation makes superfluous predictions and is rejected in favour of the more efficient binary approach.

The remainder of this section covers three topics of particular relevance to the human entity detection module: *efficient* fully-convolutional networks (FCNs), *accurate* encoder–decoder models, and *fast* multi-branch models.

Long et al. [18] were among the first to successfully apply FCNs to semantic image segmentation. With their novel deep 'skip' architecture, which combines features from shallow layers containing fine appearance information to deep layers containing coarse context information, they were able to efficiently predict segmentation masks. Since FCNs are no more than CNNs without fully-connected layers, *i.e.* they consist only of convolutional type layers interspaced with pooling and activation functions, they are able to accept arbitrarily-sized inputs and produce correspondingly-sized outputs. The main advantage is that a FCN can be trained end-to-end (and make predictions) on whole-image inputs, whereas an ordinary CNN cannot. Before this realisation,

Fig. 1. A comparison of semantic image segmentation: (i) an image of a person dancing, alongside its (ii) multi-class ground truth and (iii) binary ground-truth segmentation masks. Original image sourced from Kaggle [17].

researchers were reliant on slower and less effective patchwise training and inference methods, with preprocessing and post hoc refinements such as input shifting with output interlacing. A second advantage in swapping out fully-connected layers for convolutional ones was the reduction in network parameters and thus the more efficient learning of features. In testing, Long et al. achieved state-of-the-art segmentation on the PASCAL VOC 2011 [19], with inference taking less than one-fifth of a second.

Not long afterwards, Noh et al. [8] introduced an encoder–decoder model named DeconvNet. The general architecture, which has since become a template for accurate segmentation models, combines an ordinary downsampling convolutional network (an encoder) with a novel upsampling deconvolutional network (a decoder). The structure of the encoder, which is a truncated version of VGG-16 [7] with its fully-connected layers removed, is simply mirrored about its deepest layer to form the decoder. Instead of convolutional and pooling layers, Noh et al. [8] proposed deconvolutional and unpooling layers with switch variables as a means for the decoder to enlarge and densify activation maps, without losing spatial structure. When applied to an image, the convolutional network encodes features in the form of a multi-dimensional feature tensor, which the decoder then uses to predict a detailed segmentation mask. In a similar way to Long et al., the filters of the shallowest deconvolution layers capture coarse regions of shapes, whilst the deepest layers capture more complex details and patterns. With this novel architecture, DeconvNet claimed state-of-the-art accuracy performance on VOC 2012 [20].

To address the ever-present tradeoff between inference speed and segmentation performance, Yu et al. [21] proposed the multi-branch Bilateral Segmentation Network (BiSeNet). This novel architecture simultaneously forward-propagates an input image down two separate and distinct computational paths, *i.e.* a spatial path and a context path, to extract features at multiple scales without compromise. The shallower spatial path employs a stack of three small-stride convolutions to generate a high-resolution (1/8th-scale) feature map containing rich detail information. The deeper context path uses the lightweight Xception39 [22] module with global average pooling to efficiently encode high-level semantic context information and provide a sufficiently large receptive field. On top of these paths, Yu et al. introduce a new 'Feature Fusion Module' as a means to efficiently combine the differently-scaled feature outputs. By considering these at two distant spatial levels, BiSeNet is able to capture the complementary information required to predict detailed and accurate segmentation masks. Despite requiring a complex training strategy with multiple loss functions, Yu et al. managed to achieve 68.4% intersection over union (IOU) at 105 frames per second when being tested at full-resolution on the CityScapes test dataset [23].

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

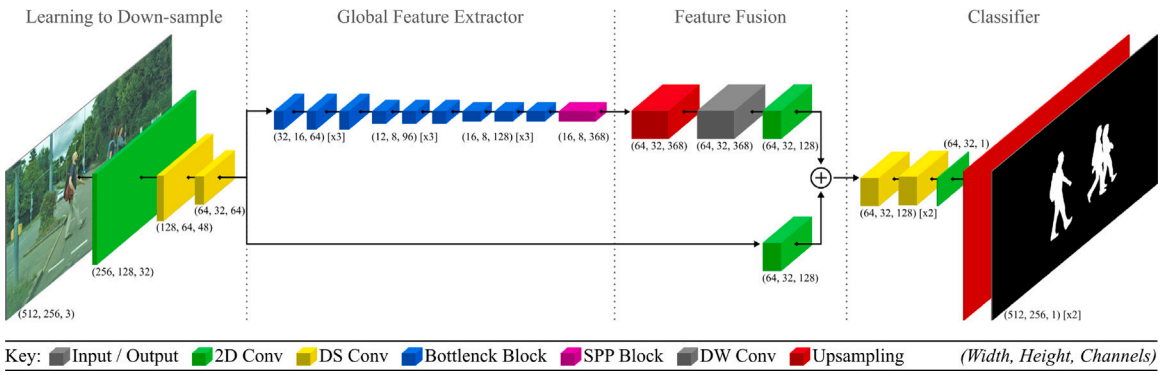


Fig. 2. The human entity detection module is a remodelled version of Fast-SCNN [24] designed specifically for real-time binary semantic image segmentation. The tensor dimensions show the remodelled network configured at 1/4 resoltion (512 × 256px).

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 (Fig. 2) is essentially a remodelled version of the state-of-the-art Fast Segmentation Neural Network (Fast-SCNN) developed by Poudel et al. [24]. 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 [23]. Although newer and potentially more advanced models have since been introduced, this four-year-old network (December, 2023) 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 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. Learning to downsample module

The LTD module employs a standard convolutional layer followed by two depthwise separable convolutional layers, each with stride 2, as a means of efficiently downsampling the input image. In imitation of the spatial path from [21], this shallow block of layers quickly and efficiently encodes rich detailed information at a high resolution. Whereas two-branch-only architectures tend to extract similar low-level features in the first few layers of each branch separately, the LTD module computes features to be used in both branches simultaneously. Not only does this arrangement efficiently share the most expensive computation between the low and high-level branches, but it also saves memory by reducing the spatial dimensions of the input to the GFE by a factor of 8. The low-level output features from the LTD module are to be passed directly into the GFE via an encoder–decoder-like skip connection.

2.1.2. Global feature extractor

The GFE forward propagates high-resolution features from the LTD module down shallow (high-resolution) and deep (low-resolution) branches simultaneously. Since the spatial detail needed to recover object boundaries is already extracted at this depth, the shallow branch directly transmits its input to the FFM without modification. At the same time, the deeper branch employs nine residual bottleneck modules, each with large receptive fields, in order to extract the high-level coarse features required to convey global context information. These residual bottleneck modules are adapted from MobileNet-V2 [25] and use efficient depthwise separable convolutions to reduce the total number of trainable parameters and FLOPS. At the end of the deep path, Poudel et al. attach a spatial pyramid pooling (SPP) block, inspired by Zhao et al. [26], as a means of aggregating context from various spatial regions.

2.1.3. Feature fusion module

Both branches from the GFE meet in the FFM. It is here that spatial detail is efficiently combined with context information, which is necessary to produce detailed and accurate segmentation masks. A combination of bilinear upsampling and convolutional type layers condition the high-level and low-level feature maps into identically sized tensors for their subsequent elementwise addition. In Fast-SCNN, addition is the preferred method of fusion since it introduces no parameters and has minimal computational cost.

2.1.4. Classifier

The feature map from the FFM enters the shallow and thus efficient decoder-like classifier at 1/8th of the original input resolution. The classifier employs two depthwise separable convolutional layers and one pointwise convolutional layer, all with stride 1, so as to produce a single-channel activation map of the same spatial dimensions. This activation map is then enlarged by a factor of 8 via an upsampling layer with bilinear interpolation, after which the final sigmoid layer generates a detailed and accurate binary segmentation mask.

2.1.5. Remodelling fast-SCNN

The original Fast-SCNN model achieves faster than real-time performance on an Nvidia Titan XP card with 12 GB 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 approximately 48 h 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 more 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

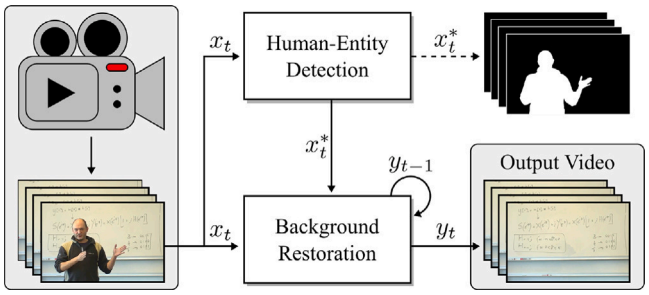


Fig. 3. A black-box depiction of the flow of video frame information via the two parallel computational modules (photograph of Dr C. Balocco kindly provided by Dr S. Giani.).

sizes. The original authors provide no information on how to implement this at any resolution, and so this work elects to build a compatable, 1/4 resolution, and memory efficient SPP block based on the 4-layer pyramid of Zhao et al. [26]. This 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 convolutional 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 [27]. In line with the original Fast-SCNN, the remodelled version is very lightweight with just over 1.16 million trainable parameters. The authors are pleased to provide additional details upon request.

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 functions 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 human entity detection module. The restoration module operates on 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 Fig. 3. As this system operates on video, *i.e.* a sequence of images and segmentation bitmaps, this procedure is simply repeated for each successive frame of footage.

The restoration algorithm works by updating each pixel of each channel 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, y_{t-1} is an ordinary RGB image containing only the most recently unobstructed pixel values from the current set of input images, X_t, where it is possible. Once this internal variable has been updated, it is output as y_t. The result of multiple image inputs is a sequence of video frames where all pixel regions containing human entities have been replaced by the most up-to-date version of the background they are currently obscuring. Conceptually, this is akin to an 'invisibility cloak'. When initialised, 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 matrices with real-integer entries makes them particularly well suited for both linear algebraic transformations and bitwise boolean operations. In each of the three channels, pixels have standard 8-bit unsigned integer luminance values ranging from 00000000_2 to 11111111_2 inclusive whereas binary masks have binary values scaled up to 8-bits unsigned integers: either 00000000_2 or 11111111_2. Computationally, this makes the procedure of updating pixels relatively straightforward. To begin, a bitwise NOT operator is applied to the segmentation mask, x_t^*, 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, independently and sequentially, to all pixels of all three channels of the current input image. This produces a modified image where the values of all pixels containing persons are zeroed and those containing the background are unaltered. An identical process takes place with the unaltered 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 to produce the new output frame. The following pseudocode block, developed in this work, summarises the restoration algorithm more formally.

Algorithm 1: The background restoration procedure.

Input : An image, x_t, and its predicted binary segmentation mask x_t^*.
Output: An image, y_t, with its background restored.

```
for (x_t, x_t*) ∈ {(x_0, x_0*), (x_1, x_1*), ..., (x_n, x_n*)} do
    if t = 0 then
        y_t ← x_t
    else
        y_t ← 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 Fig. 4.

2.3. Evaluation metrics

Before reporting the results, it is important to make clear which metrics are genuinely useful in assessing the predictive performance of the human entity detection module. In the context of binary image segmentation, which is technically an extension of binary classification, the binary accuracy metric provides relatively little valuable insight since it is heavily reliant on the two classes being well-balanced [28]. To illustrate this point: if only 5% of an image contains pixels belonging to the class 'human', then a model that always classifies the whole image as 'background' would claim to achieve 95% binary accuracy. This is clearly counterproductive as this "highly accurate" model is equivalent to having no model at all. By basing the entire metric on the absolute number of true positives and true negatives, binary accuracy fails to draw any attention to the relative proportion of prediction

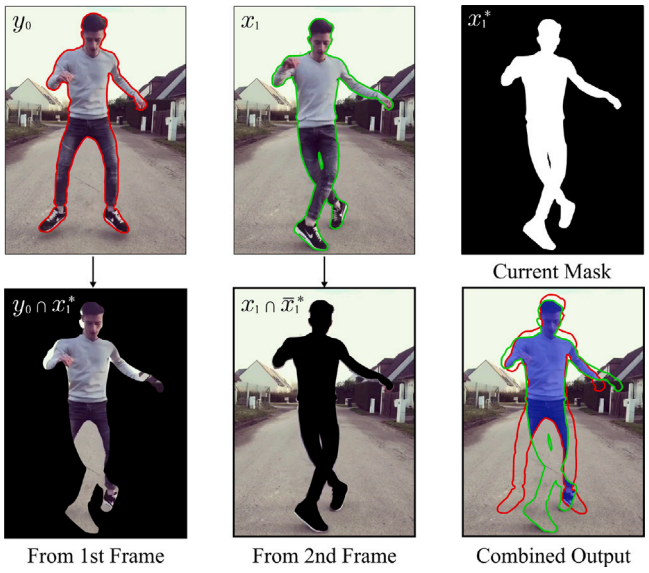


Fig. 4. The first cycle of the background restoration procedure: (bottom left) the current mask capturing the first frame foreground, (bottom centre) the current mask inverse capturing the second frame background, and (bottom right) a summation of the two. Green and red outlines show the dancer's position in the first and second frames respectively. What remains of the foreground after one cycle is shaded blue.

errors, *i.e.* false positives and false negatives. In response to these shortcomings, this study elects to use recall and precision metrics [29], both of which are based on relative predictive performance:

$$\text{Recall} = \frac{TP}{TP+FN} \quad \text{and} \quad \text{Precision} = \frac{TP}{TP+FP} \tag{1}$$

where TP, FN and FP are the numbers of pixels in a binary segmentation mask that correspond to true positive, false negative and false positive predictions respectively. When used in the context of this work, recall quantifies the proportion of ground-truth human pixels that are correctly identified as human, and precision quantifies the proportion of all predicted human pixels that are correctly identified. In other words, a model can have 100% recall regardless of the number of false positive predictions provided it predicts every single human pixel. Equally, a model can have 100% precision regardless of the number of false negative predictions if it never predicts a non-human pixel. It is technically possible to have a model that achieves both, *i.e.* it predicts exactly ground truth, although this would be very unlikely. Clearly, neither of these metrics is very useful on its own: instead, it is standard practice to combine the two in a Dice Similarity Coefficient (DSC) [29], which is defined as the harmonic mean of recall and precision:

$$\text{DSC} = \frac{2TP}{2TP+FP+FN} \tag{2}$$

When training a segmentation model using real image data, there exists a tradeoff between recall and precision that limits the ability to maximise both simultaneously. The DSC is one way of finding the balance point in this tradeoff if false positive and false negative predictions are equally undesirable. Unfortunately, in the context of this work, they are not. From a viewing student's perspective, false positive predictions go unnoticed whereas false negative predictions pose a distraction. In other words, it would be acceptable for the model to over-predict (high recall), but not to under-predict (high precision). For these reasons, it is important to consider recall, precision and DSC jointly.

3. Experimental procedure

This section addresses four technical points concerning the experimental procedure: information regarding the feasibility of transfer learning; the methods of training; details of implementation; and the dataset.

3.1. Transfer learning

Transfer learning can often be used to leverage the positive training outcomes of one study to the benefit of another [30]. The basic framework involves finding a pre-trained model that works well on a similar task, then re-initialising or fine-tuning certain parts of that model to perform a more specific task. Not only can this technique hugely reduce training times, but it can also provide a means to solve many specific problems well, even with small quantities of training data. Unfortunately, since no pre-trained versions of the Fast-SCNN network [24] have yet been published, this work was obliged to train its human-entity detection module from scratch. There are, however, several advantages in this: not only did it provide the opportunity to modify the underlying architecture without constraint, but it also allowed for greater control over the most influential training hyperparameters.

3.2. Training

An empirical technique proposed by Smith [31] 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 $2{\times}10^{-1}$ 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. [24], 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.

3.3. 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 [32]. 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

Table 1
Reference training settings and hyperparameters.

Hyperparameter	Description
Optimiser	SGD (with momentum = 0.9)
Learning rate	2.0×10^{-1} at start → 1.0×10^{-5} at finish
Scheduler	Degree-two polynomial LR decay
Loss	Tversky loss function (with $\beta = 0.9$)
Duration	160 epochs + early stopping
Batch size	32

Table 2
Reference Tik-Tok dataset augmentation details.

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

module was optimised using a parametric loss function based on the popular Tversky Index [33]:

$$TI = \frac{\sum y_i \hat{y}_i}{\sum y_i \hat{y}_i + \sum \beta y_i (1 - \hat{y}_i) + \sum (1 - \beta)(1 - y_i)\hat{y}_i} \tag{3}$$

where y_i is a normalised true pixel value $\{0, 1\}$, $\hat{y}_i$ 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. For the full details of hyperparameter tuning, refer to Section 4.1. For ease of reference, the most relevant training details are summarised in Table 1.

3.4. Implementation

The remodelled Fast-SCNN architecture was built and trained in a Python Jupyter Notebook using the TensorFlow 2.0 machine learning library and Keras API. All training experiments were conducted over the internet on free-to-use cloud-based virtual machines hosted by Google Colab. Whilst this free service did not guarantee specific hardware availability, the GPUs available included Nvidia K80s, T4s, P4s, and P100s (up to 24 GB with 4992 CUDA cores) running both CUDA v11.2 and cuDNN for acceleration.

The complete system, which incorporates both the trained human entity detection module and background restoration module, is an ordinary Python script that was developed and tested in the Spyder IDE using the OpenCV real-time computer vision library. All benchmark experiments were run locally on a laptop installed with an Intel Core i7 CPU, 8 GB of RAM, and an Nvidia Quadro P520 GPU (up to 2 GB with only 384 CUDA cores) running CUDA v11.1 and cuDNN.

3.5. The dataset

The segmentation network was trained on a dataset of 2615 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 [17] 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 [34]. 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 2.

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. Fig. 5 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 Table 1 and Table 2 respectively, and the Tversky loss parameter was 0.9 throughout.

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 [31] 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

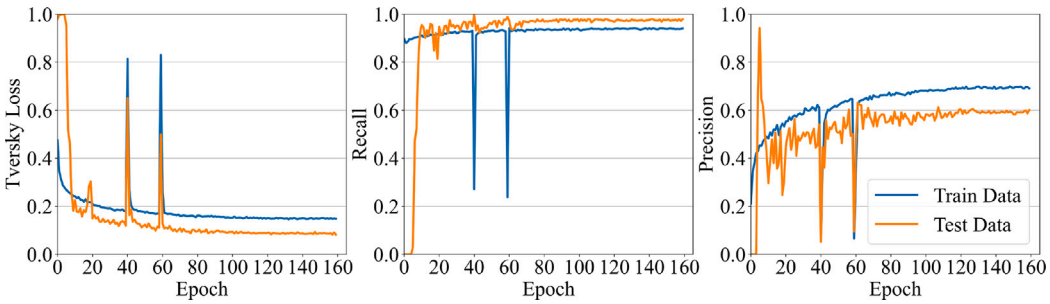


Fig. 5. Training and testing curves of loss (left), recall (centre) and precision (right) for the best performing training run.

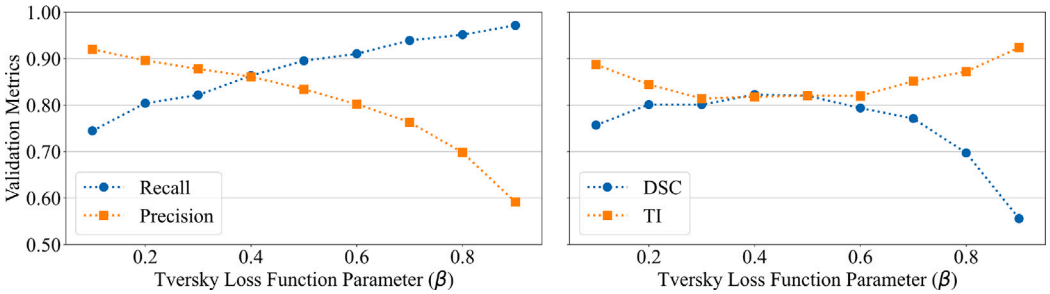


Fig. 6. A cross-validation study on the effect of varying the Tversky loss function parameter (β) between 0.1 and 0.9.

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 Fig. 5. 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. Fig. 6 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, ..., 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.

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 Fig. 6, 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 [29] 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 $\beta = 0.9$.

Fig. 7 presents segmentation masks for models trained with five different Tversky loss parameter values, facilitating visual comparison and providing justification for the preference of the TI over the DSC. Notably, at $\beta = 0.4$, favoured by the DSC, the predicted segmentation mask exposes almost the entire arms and feet of the ground truth subject. In contrast, at $\beta = 0.9$, favoured by the TI, the predicted segmentation mask fully encapsulates the subject's entire body. Overlaying the ground truth segmentation masks onto predicted segmentation masks reveals that, for this specific application, higher recall is beneficial in minimising output contamination. Given that only the segmentation mask corresponding to $\beta = 0.9$ fully encapsulates the ground truth, it is

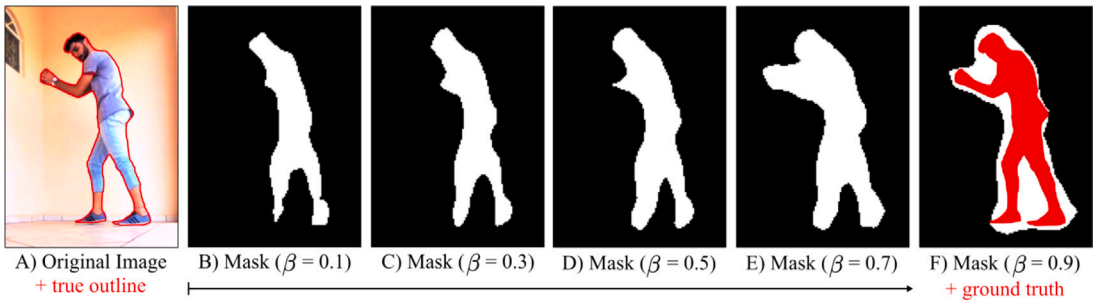


Fig. 7. Binary segmentation masks for various settings of the Tversky loss parameter (β). Original image from Kaggle [17].

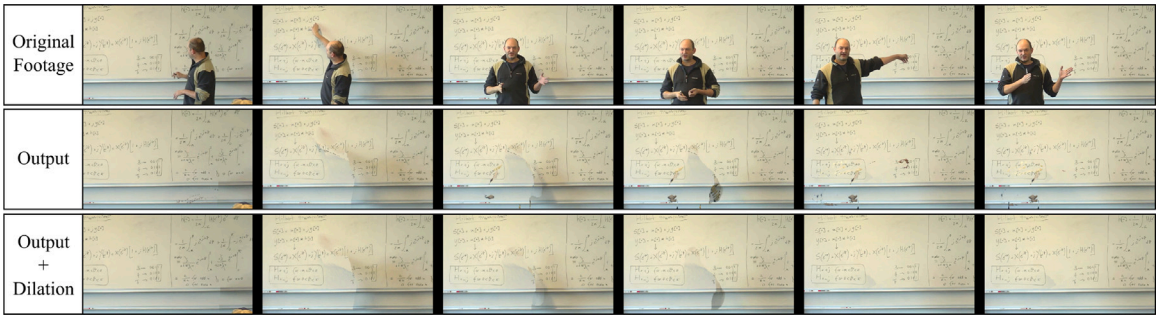


Fig. 8. The complete system in action: (top row) still-image frames extracted from footage; (centre row) the corresponding raw system output; and (bottom row) the raw system output with added dilation.

concluded that 0.9 is the most suitable Tversky loss parameter for the human entity detection module.

4.3. Runtime benchmarking

The 'complete system' refers to the human entity detection and background restoration modules, connected as shown in Fig. 3, with resampling added on either side of the CNN. In all benchmarking experiments, the segmentation model was trained with $\beta = 0.9$ and with all other hyperparameters and augmentations as described in Section 3. When presented with any high-resolution video input, the complete system will, so far as possible, compute and return a corresponding high-resolution video output with all presenters removed. Fig. 8 displays individual frames taken from a typical lecture recording, both before and after these were processed by the complete system. An additional row of differently processed images is included in Fig. 8 to show the effect of 'dilating' binary masks prior to background restoration. This drop-in 'dilation' feature simply convolves the segmentation output with a 10-pixel-diameter kernel as a means of thickening the outline of human forms without disproportionately increasing the number of false positive outcomes.

Having inspected each of the video frames, it appears the system successfully and consistently removes the presenter from the view of the whiteboard. Whilst there are some 'ghost pixels' that follow the presenter in a number of the output frames, these only appear in relatively low numbers and do not impact greatly on the viewing experience. By introducing dilation, these unwanted pixels are removed completely (neglecting shadows) and the whiteboard can be seen clearly in all examples.

The system demanded 2.32 GB of GPU memory during testing, which was 0.32 GB more than the available hardware could provide, and therefore was unable to run at full speed. To be considered as real-time, the system's combined per-frame inference and compute times would need to translate to a frame rate greater than or equal to typical video frame rates, *i.e.* 30 frames per second (FPS). By unloading computation to the CPU, it was only possible to achieve a modest 10.98 FPS, regardless of dilation. This system could arguably be run in real-time [24] with either slightly more computational power, or network quantisation.

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 [24] 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 work 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 [31] 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 efficiently 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.

CRediT authorship contribution statement

R.M. Sales: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **S. Giani:** Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] Robertson B, Flowers MJ. Determining the impact of lecture videos on student outcomes. Learn Teach 2020;13(2):25–40.

[2] Louis-Jean J, Cenat K. Beyond the face-to-face learning: A contextual analysis. Pedagogical Res 2020;5(4):em0077.

[3] Nock R, Nielsen F. Statistical region merging. IEEE Trans Pattern Anal Mach Intell 2004;26(11):1452–8.

[4] Tao W, Jin H, Zhang Y. Color image segmentation based on mean shift and normalized cuts. IEEE Trans Syst Man Cybern B 2007;37(5):1382–9.

[5] Plath N, Toussaint M, Nakajima S. 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, p. 817–24.

[6] Zhao C. Image segmentation based on fast normalized cut. Open Cybern Syst J 2015;9:28–31.

[7] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. In: Bengio Y, LeCun Y, editors. 3rd international conference on learning representations, conference track proceedings. 2015, URL http://arxiv.org/abs/1409.1556.

[8] Noh H, Hong S, Han B. Learning deconvolution network for semantic segmentation. In: 2015 IEEE international conference on computer vision. IEEE Computer Society; 2015, p. 1520–8.

[9] Ronneberger O, Fischer P, Brox T. U-Net: Convolutional networks for biomedical image segmentation. In: 18th international conference of medical image computing and computer-assisted intervention. Springer; 2015, p. 234–41.

[10] Paszke A, Chaurasia A, Kim S, Culurciello E. ENet: A deep neural network architecture for real-time semantic segmentation. 2016, CoRR arXiv:1606.02147, URL http://arxiv.org/abs/1606.02147.

[11] Minaee S, Boykov Y, Porikli F, Plaza A, Kehtarnavaz N, Terzopoulos D. Image segmentation using deep learning: A survey. 2020, CoRR arXiv:2001.05566, URL https://arxiv.org/abs/2001.05566.

[12] Liu L, Ouyang W, Wang X, Fieguth PW, Chen J, Liu X, et al. Deep learning for generic object detection: A survey. Int J Comput Vis 2020;128(2):261–318.

[13] Takos G. A survey on deep learning methods for semantic image segmentation in real-time. 2020, CoRR arXiv:2009.12942, URL https://arxiv.org/abs/2009.12942.

[14] Newson A, Almansa A, Fradet M, Gousseau Y, Pérez P. Towards fast, generic video inpainting. In: Proceedings of the 10th European conference on visual media production. ACM Press; 2013, p. 1–8.

[15] Kim D, Woo S, Lee J-Y, Kweon IS. Deep video inpainting. In: 2019 IEEE/CVF conference on computer vision and pattern recognition. IEEE; 2019, p. 5785–94.

[16] Zou Z, Shi Z, Guo Y, Ye J. Object detection in 20 years: A survey. 2019, arXiv e-prints arXiv:1905.05055, URL https://ui.adsabs.harvard.edu/abs/2019arXiv190505055Z.

[17] Segmentation Full Body TikTok Dancing Dataset, URL https://www.kaggle.com/tapakah68/segmentation-full-body-tiktok-dancing-dataset.

[18] Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation. In: 2015 IEEE conference on computer vision and pattern recognition. IEEE; 2015, p. 3431–40.

[19] Everingham M, Van-Gool L, Williams CKI, Winn J, Zisserman A. The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results.

[20] Everingham M, Van-Gool L, Williams CKI, Winn J, Zisserman A. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results.

[21] Yu C, Wang J, Peng C, Gao C, Yu G, Sang N. BiSeNet: Bilateral Segmentation Network for Real-Time Semantic Segmentation. In: Ferrari V, Hebert M, Sminchisescu C, Weiss Y, editors. 15th European conference of computer vision, proceedings, part XIII. Lecture notes in computer science, Vol. 11217, Springer; 2018, p. 334–49.

[22] Chollet F. Xception: Deep learning with depthwise separable convolutions. In: 2017 IEEE conference on computer vision and pattern recognition. IEEE Computer Society; 2017, p. 1800–7.

[23] Cordts M, Omran M, Ramos S, Rehfeld T, Enzweiler M, Benenson R, et al. The cityscapes dataset for semantic urban scene understanding. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR). IEEE Computer Society; 2016, p. 3213.

[24] Poudel RPK, Liwicki S, Cipolla R. Fast-SCNN: Fast semantic segmentation network. In: 30th british machine vision conference. BMVA Press; 2019, p. 289, URL https://bmvc2019.org/wp-content/uploads/papers/0959-paper.pdf.

[25] Sandler M, Howard AG, Zhu M, Zhmoginov A, Chen L. MobileNetV2: Inverted residuals and linear bottlenecks. In: 2018 IEEE conference on computer vision and pattern recognition. IEEE Computer Society; 2018, p. 4510–20.

[26] Zhao H, Shi J, Qi X, Wang X, Jia J. Pyramid scene parsing network. In: 2017 IEEE conference on computer vision and pattern recognition. IEEE Computer Society; 2017, p. 6230–9.

[27] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A simple way to prevent neural networks from overfitting. J Mach Learn Res 2014;15(56):1929–58, URL http://jmlr.org/papers/v15/srivastava14a.html.

[28] Sun Y, Wong AKC, Kamel MS. Classification of imbalanced data: a review. Int J Pattern Recognit Artif Intell 2009;23(4):687–719.

[29] Hashemi SR, Salehi SSM, Erdogmus D, Prabhu SP, Warfield SK, Gholipour A. Asymmetric loss functions and deep densely-connected networks for highly-imbalanced medical image segmentation: Application to multiple sclerosis lesion detection. IEEE Access 2019;7:1721–35.

[30] Torrey L, Shavlik J. In: Olivas ES, Guerrero JDM, Martinez-Sober M, Lopez AJS, editors. Handbook of research on machine learning applications and trends: algorithms, methods, and techniques. IGI Global; 2010, p. 242–6.

[31] Smith LN. Cyclical learning rates for training neural networks. In: 2017 IEEE winter conference on applications of computer vision. IEEE; 2017, p. 464.

[32] Jadon S. A survey of loss functions for semantic segmentation. In: 2020 IEEE conference on computational intelligence in bioinformatics and computational biology. IEEE; 2020, p. 1–7.

[33] Yeung M, Sala E, Schönlieb C-B, Rundo L. A mixed focal loss function for handling class imbalanced medical image segmentation. 2021, arXiv e-prints arXiv:2102.04525, URL https://ui.adsabs.harvard.edu/abs/2021arXiv210204525Y.

[34] Shorten C, Khoshgoftaar TM. A survey on image data augmentation for deep learning. J Big Data 2019;6(1):60.