|  |  |
| --- | --- |
| IEEE JOURNAL OF BIOMEDICAL AND HEALTH INFORMATICS, VOL. 24, NO. 4, APRIL 2020 | 1059 |
|  |  |



Low-Memory CNNs Enabling Real-Time Ultrasound Segmentation Towards Mobile Deployment

Sagar Vaze, Weidi Xie, and Ana I. L. Namburete



***Abstract*—Convolutional Neural Networks (CNNs), whichare currently state-of-the-art for most image analysis tasks, are ill suited to leveraging the key benefits of ultrasound imaging — specifically, ultrasound’s portability and real-time capabilities. CNNs have large memory footprints, which obstructs their implementation on mobile devices, and require numerous floating point operations, which re-sults in slow CPU inference times. In this article, we pro-pose three approaches to training efficient CNNs that can operate in real-time on a CPU (catering to a clinical setting), with a low memory footprint, for minimal compromise in ac-curacy. We first demonstrate the power of ‘thin’ CNNs (with very few feature channels) for fast medical image segmen-tation. We then leverage separable convolutions to further speed up inference, reduce parameter count and facilitate mobile deployment. Lastly, we propose a novel knowledge distillation technique to boost the accuracy of light-weight models, while maintaining inference speed-up. For a negli-gible sacrifice in test set Dice performance on the challeng-ing ultrasound analysis task of nerve segmentation, our final proposed model processes images at 30 fps on a CPU, which is 9***×* **faster than the standard U-Net, while requiring 420***×* **less space in memory. Code for this work is available at:** [**https://github.com/sagarvaze96/lightweight\_unet.**](https://github.com/sagarvaze96/lightweight_unet)

***Index Terms*—Image segmentation, knowledge transfer,mobile applications, real-time systems.**

1. INTRODUCTION

N RECENT YEARS, convolutional neural networks (CNNs) **I** have achieved state-of-the-art performance in most imageanalysis tasks. After first being deployed for classification challenges [1], [2], CNNs have been adapted successfully for other tasks such as detection, registration, and segmentation [3]. Specifically, in the context of segmentation (one of the most widely tackled problems in the medical imaging literature), the U-Net architecture has been extensively utilised and adapted for a number of scenarios [4].

However, though CNNs achieve very high accuracies, their speed and memory requirements remain a significant limitation in their feasibility to real-world deployment. These networks

Manuscript received May 31, 2019; revised November 2, 2019; ac-cepted December 11, 2019. Date of publication February 14, 2020; date of current version April 6, 2020. *(Corresponding author: Ana Nambu-rete.)*

Sagar Vaze and Ana I. L. Namburete are with the Institute of Biomed-ical Engineering, University of Oxford, Oxford OX3 7DQ, U.K. (e-mail: [sagar.vaze96@gmail.com;](mailto:sagar.vaze96@gmail.com) [ana.namburete@eng.ox.ac.uk)](mailto:ana.namburete@eng.ox.ac.uk).

Weidi Xie is with Visual Geometry Group, University of Oxford, Oxford OX1 2JD, U.K. (e-mail: [weidi@robots.ox.ac.uk)](mailto:weidi@robots.ox.ac.uk).

Digital Object Identifier 10.1109/JBHI.2019.2961264

typically contain millions of parameters and require hundreds of millions of floating point operations (FLOPs) for inference [5]. Furthermore, work which is directed toward real-time CNN deployment typically demonstrates performance time on GPUs [6], which are expensive and will often require replacing or upgrading existing clinical hardware. We aim to tackle these problems, developing computationally efficient CNNs capable of real-time inference on CPUs.

Ultrasound is an appropriate medical imaging technique on which to demonstrate efficient image analysis methods, wherein the methods provide low memory costs and fast inference. Despite the presence of higher resolution techniques, ultrasound remains a primary imaging modality in medicine. Though this is due in part to the lack of ionising radiation, the method is often favoured due to its portability, low cost and real-time capabilities — features which are generally absent in other popular techniques such as magnetic resonance imaging (MRI) and computed tomography (CT). For instance, ultrasound is widely used in pre-natal care, in which its high frame rates allow the image capture of the moving fetus. Ultrasound frame rates typically vary between 10 and 30 frames per second (fps) [7]. In this work, we use the widely accepted figure of 30 fps to define ‘real-time’ processing [8].

Ultrasound is also particularly applicable in developing na-tions, most notably in rural areas. In these locations, porta-bility and low cost are crucial, and the market now contains numerous ultrasound probes which are both hand-held and affordable. We reiterate that (particularly in low-income na-tions) many clinics may not have access to high performance GPUs to process ultrasound data, and that we aim to propose solutions that are catered towards CPU architectures for clinical deployment.

*A. Contributions*

In this work we develop CNNs which can both process ultra-sound images in real-time and have a small memory footprint, thus enabling seamless mobile deployment. Our contributions can be summarised as follows, and are illustrated in Fig. 1:

Thin CNNs with few feature channels per layer. These models provide model size reduction and inference speed-up, while maintaining accuracy (Fig. 1(b)).

Separable convolutions in the context of thin CNNs to further reduce model size and inference time (Fig. 1(c)).

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

1060 IEEE JOURNAL OF BIOMEDICAL AND HEALTH INFORMATICS, VOL. 24, NO. 4, APRIL 2020

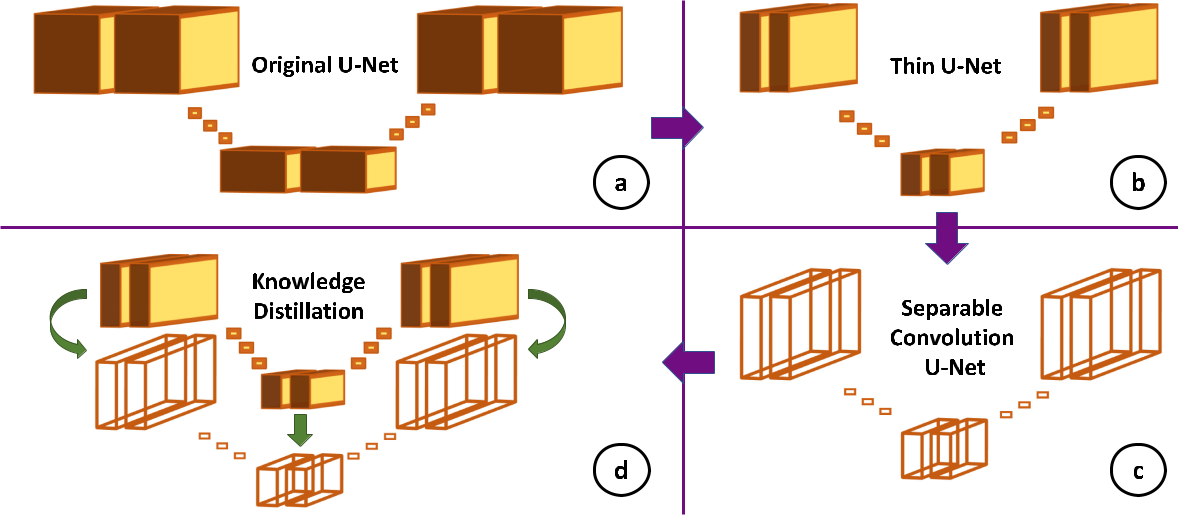


Fig. 1. Methods to reduce CNN inference times and memory footprint while maintaining accuracy, including: ‘thin’ networks with very few feature channels (b); separable convolution layers (c); and knowledge distillation (d).

Knowledge distillation to boost performance of thin, sepa-rable convolution CNNs to accuracies comparable to their regular convolution counterparts (Fig. 1(d)).

Demonstration of real-time CNN inference (30 fps) on

CPU architectures and deployment on mobile devices. Our first contribution is the exploration of very *thin* CNN

architectures to reduce network parameter count and inference time. These architecures contain minimal feature channels (and, therefore, fewer parameters) per convolutional layer. In the com-puter vision field, networks are required to have a large number of parameters to manage a large variation in input appearance and many target classes. This trend has transferred to the medical imaging domain: numerous state-of-the-art networks use hun-dreds of filters [4] or large convolutional kernels [9] to increase model predictive performance. However, we show that there exist tasks in this field for which model size can be drastically reduced without sacrificing accuracy.

Secondly, we investigate the use of *depth-wise separable con-volutions* [10] in thin architectures to further speed up inference.Such convolutions parametrise three-dimensional convolution tensors in an efficient manner. Again, in the computer vision literature, separable convolutions have been incorporated in large architecures to provide inference speed-up and memory savings for little compromise in accuracy [5], [11]. In this paper, we demonstrate the use of separable convolutions in models already designed to have a small memory footprint. We show that, even in very compact architectures, separable convolutions can perform comparably to regular convolutions, but do come with a small compromise in accuracy.

To manage this compromise, we propose a novel *knowledge* *distillation* technique for use in accuracy-critical settings. Here,we aim to boost the performance of separable convolutional networks to accuracies within the range of their regular variants, while maintaining significant inference time gains.

We demonstrate our methods on the task of nerve segmenta-tion from 2D ultrasound images. The task is the segmentation

of the Brachial Plexus (BP), and was proposed as a Kaggle challenge in 2016 [12]. Clinically, the identification of the BP in ultrasound images can assist in the placement of surgical pain management catheters. This dataset also presents a challenging and familiar task which has been widely studied in the ultrasound analysis literature since its publication [13], [14].

This work follows from a MICCAI workshop paper [15] in which separable convolutions were demonstrated in a medical imaging context with a large network architecture to yield signif-icant memory savings. We now take the work further, developing even lighter architectures capable of real-time inference.

Finally, we note that we demonstrate the proposed methods on the most computationally intensive application of standard CNNs. We tackle a semantic segmentation task, for which the architectures used are typically larger and require longer inference times than those used for classification, detection or registration.

1. RELATED WORK

CNNs have achieved state-of-the-art results in most computer vision segmentation tasks, most notably beginning with the ‘Fully Convolutional Network’ (FCN) [16]. The FCN’s con-tribution is to recast the fully connected layers of traditional patch-based segmentation CNNs as 1 *×* 1 convolutional layers, facilitating dense pixel-wise predictions for the entire input im-age (though 1 *×* 1 convolutions were also used in a classification framework by Lin *et al.* [17]). The architecture style became known as an *encoder-decoder* architecture — incorporating a down-sampling path, as in classification networks, followed by an up-sampling path. In the medical imaging field, the most popular segmentation architecture is the U-Net [4], introduced soon after the FCN, which has been widely adapted to achieve state-of-the-art performance for a range of medical imaging applications. In this work, we use the original U-Net design as a baseline against which we compare our proposed architectures.

|  |  |
| --- | --- |
| VAZE *et al.*: LOW-MEMORY CNNs ENABLING REAL-TIME ULTRASOUND SEGMENTATION TOWARDS MOBILE DEPLOYMENT | 1061 |
|  |  |

Real-time CNN inference on ultrasound images has been no-tably demonstrated with the SonoNet [6] (albeit for detection and localisation rather than segmentation). In their work, the authors used a model based on the VGG classification network [18] to analyse ultrasound images at a rate of up to 43.5fps. Crucially, however, the frame rates reported were achieved on a high-end GPU (NVIDIA GeForce GTX 1080). In this work, we aim for real-time inference on CPUs, which are available in clinics and are fundamentally, structurally different to GPUs, making them approximately an order of magnitude slower at CNN inference (see Section IV-C). The precise clinical framerates in ultrasound vary depending on application-specific tissue density and depth. In this work, we use a target of 30 fps to define ‘real-time’ [8].

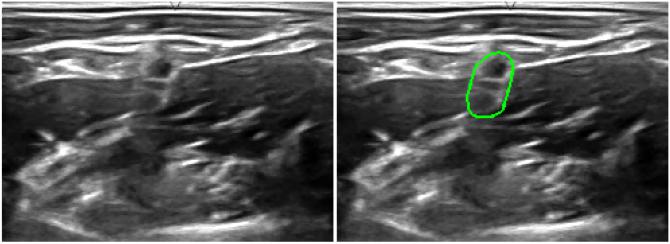
The large number of network parameters is the main obstacle in the way of efficient CNNs. These parameters must be stored on disk and result in the networks having large memory foot-prints, which could prevent mobile application. Furthermore, these parameters contribute numerous FLOPs to the networks’ forward pass, which results in lengthy inference times. A class of methods which deals with this problem efficiently stores network parameters, with hash bucketing [19] and parameter quantisation [20] being used effectively. An extreme form of quantisation, in which parameter values are constrained to one of two values (+1 and *−*1), has been shown to significantly reduce GPU compute time [21].

Separable convolutions constitute another class of methods, which aim to reduce parameter count by parametrising CNN convolutions efficiently. The variant we leverage in this work (the depth-wise separable convolution) was introduced in Lau-rent Sifre’s thesis and decomposes a 3D convolutional tensor into a 2D filter and a 1D vector [10]. Such convolutions have been adopted with success in classification architectures such as Xception [11] and MobileNet [5], before being incorporated in the latest DeepLab segmentation network to improve inference times [22]. In this work, we incorporate depth-wise separable convolutions in a light-weight architecture adapted for a medical image segmentation task. We demonstrate that, even in smaller architectures, separable convolutions can perform comparably to the regular variant, but do come with a compromise in accuracy.

We employ knowledge distillation to remedy this, a technique which aims to transfer salient representational information from a large network to a smaller one. The technique was introduced in its most popular form by Hinton *et al.* [23] who used the larger network’s outputs as ‘soft-targets’ for the smaller network. In this work, we propose an alternative approach better suited to semantic segmentation, which encourages the smaller network to recreate the intermediate feature maps of the larger model. This technique is similar to the *perceptual loss* (commonly used for style transfer [24]). Intermediate feature representations have also been used for knowledge distillation to provide ‘hints’ in a classification framework, by training a regressor to map the larger teacher’s representation to the student’s space. In this work, we use the intermediate feature maps directly as the activations are of the same dimensions.

We also note concurrent work by Mangalam and Salz-mann [25], who demonstrate significant memory savings for a 5% mean Dice compromise in a microscopy task. They train

Fig. 2. A sample from the Kaggle nerve segmentation dataset. Left panel shows the ultrasound image, with the right panel showing the image and overlayed expert annotation.



a thin U-Net using knowledge distillation similarly to Hinton *et al.* [23]. In this work, we propose the use of separable con-volutions and a distillation technique tailored for segmentation, demonstrating real-time CPU inference for a negligible (0.7% mean) sacrifice in accuracy.

1. DATASET

Data for this task was initially published for a Kaggle competi-tion in 2016 [12]. We used a subset of the original training set for the challenge, which contains 5635 2D ultrasound images and their corresponding segmentation masks, annotated by experts, as shown in Fig. 2.

The original dataset contains a number of images where the structure of interest (the Brachial Plexus nerve cluster) is missing and correspondingly contains no foreground label. As such, the original challenge involved predicting blank segmentation masks for samples in which the structure was absent. This could be achieved with post-processing or optimisation of a more complex loss. For the purposes of this work, in which we focus on general methods for fast inference, we remove the images lacking the BP and facilitate a standard CNN training procedure. We note that our reference for accuracy, the original U-Net architecture, was also trained and evaluated on this curated dataset to ensure our comparisons are fair.

The curated dataset, which we use to train the models in this work, consists of 2323 images. The images were resized to 128 *×* 256 pixels (from 420 *×* 580 pixels) to approximately maintain their original aspect ratio while also allowing repeated max-pooling of the input images during CNN inference. Further details on the choice and effect of image size are given in Section V-D2.

The 2323 2D slices in the dataset were divided in an 80/20 split (1858/465 images) for training and testing the proposed models, with the training set further subdivided into five folds used for cross-validation experiments.

IV. METHODS

In this section, we discuss the methods proposed in this paper. We begin by outlining the original U-Net [4] in Section IV-A, which we use as a benchmark in the context of the Kaggle nerve segmentation challenge. Section IV-B then discusses thin CNNs for faster inference, wherein the number of channels in each convolutional layer is substantially reduced. Separable

Here, *θ* represents all CNN parameters; *IoU* represents the

intersection-over-union score of the manual labels (*Y* ) and the

ˆ

network predictions (*Y* ); and *λ*1 signifies the weight decay (L2-regularisation) strength. The sum is taken over the *n* examples in the training set.

The *IoU* loss (a differentiable proxy for the Dice coefficient) was selected to best approximate the qualitative visual similar-ity between the manual annotation and CNN prediction. The loss is defined as follows, wherein the manual annotation and network prediction are represented as vectorised quantities of dimensionality *d*:

With the exception of the distilled networks (see Section IV-D), all models were trained with a common process. The networks’ parameters were optimised with respect to the following loss function:

convolutions are detailed in the context of thin architectures in Section IV-C [10]. Finally, Section IV-D describes the dis-tillation process which we use to boost separable convolution performance.

*A. Experimental Setup*

*i*=1

*L*

*ˆ*

*1(θ;Y ,Y ) =*

*n*

*1062*

IEEE JOURNAL OF BIOMEDICAL AND HEALTH INFORMATICS, VOL. 24, NO. 4, APRIL 2020

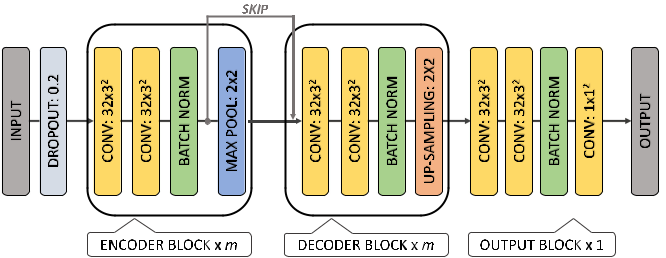
**

Fig. 3. Block diagram of a U-Net type architecture, with *m* blocks in the encoder and decoder paths. A single block contains a combination of the following layers: convolutional (number of channels *×* filter size2); batch normalisation; dropout; max-pooling *or* up-sampling. Not shown is

*IoU* (*yi, y*ˆ*i*) + *λ*1*||θ||*2 (1) a drop-out layer after the first decoder block, with a drop out rate of 0.5.

The original U-Net architecture can also be represented like this, with *m* = 4and2 *×* 2convolutional layers after every up-sampling operation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *IoU* (*yi, y*ˆ*i*) = *−* | *d* | 2 *yi* *·* *y*ˆ*i* | (2) |  |
|  |
|  |  |
|  | *j*=1 *yi,j* + *y*ˆ*i,j* | |  |  |

Optimisation was carried out using mini-batch gradient de-scent (SGD) with the learning rate annealed from its initial value by a factor of 10 every 15 epochs. Furthermore, warm restarts were used (wherein the learning rate scheduler was reset every 40 epochs) to ensure the networks’ convergence [26]. The experiments were run for a total of 100 epochs. We chose our optimisation procedure to be simple and in line with standard practise so as to clearly demonstrate our contributions.

The training hyper-parameters to be set were the initial learn-ing rate and *λ*1. We empirically observed that 1 *×* 10*−*2 and 1 *×* 10*−*3 were suitable for most networks.

The experiments were carried out using the Keras frame-work with a TensorFlow back-end, using a NVIDIA GeForce GTX 1080Ti for GPU inference time measurement, with CPU inference times measured on an Intel Xeon E5-2667. When evaluating on an unseen test set (see Section V), inference times were also measured on a laptop CPU (Intel i5-4200 M) and two mobile devices (Google Pixel 2 and Samsung Galaxy Tab A).

*B. Original U-Net*

The U-Net has been adapted and modified numerous times for a range of segmentation challenges. In this work, we use the architecture as originally presented in [4] to provide a baseline.

Specifically, the U-Net has been implemented as a type of *encoder-decoder* network: containing a down-sampling pathwhich compresses image information into a minimal repre-sentation, before up-sampling to recover the original image dimensions and provide dense pixel-wise predictions. The *en-coder* path consists of four *blocks* (seeFig. 3), with each block

containing two convolutional layers (3 *×* 3 filters) followed by a max-pooling layer (2 *×* 2). The *decoder* path is nearly sym-metrical in terms of block count, but with up-sampling in place of max-pooling. Furthermore, the decoder in the original U-Net contains additional 2 *×* 2 convolutional layers directly after the up-sampling layers. The architecture also contains an *output* *block* which takes the results of the decoder, passes them throughtwo 3 *×* 3 convolutional layers, and one 1 *×* 1 convolutional layer, resulting in the final prediction.

The original U-Net incorporates *dropout* regularisation on the input layer (*p* = 0*.*2) and on the most down-sampled block (*p* = 0*.*5). It also incorporates *skip connections* which pass feature maps from intermediate blocks in the encoder directly to the corresponding block in the decoder: this ameliorates the coarseness of the up-sampling process. One change which we have made to this architecture is to include *batch-normalisation* in every block [27]. The ReLU non-linearity is used for all neurons with the exception of the output layer. In the output layer, a softmax function provides a confidence of nerve pres-ence between 0 and 1 at each pixel location.

Although the U-Net achieves state-of-the-art segmentation accuracy for many applications, it contains many parameters, the majority of which are the learned weights and biases of the convolutional layers within the architecture. These many parameters (*≈* 30 million) could cause difficulties in deploying the model on low-memory mobile devices. TensorFlow, a widely used framework for deploying neural networks, stores each pa-rameter as a 32-bit float, making the U-Net take over one hundred mega-bytes in memory. Correspondingly, many parameters lead to a large number of FLOPs in the network’s forward pass. Each parameter requires a number of multiplications and additions, thus resulting in lengthy inference times. In this work, we aim to develop light-weight models with low inference times in the pursuit of real-time image analysis.

*C. Thin CNNs*

A primary source of the many parameters in the U-Net is the large network *width:* it contains 64 filters per convolutional layer in the first block, with the number of filters doubling after each max-pooling layer, leading to 1024 filters in each of the most down-sampled layers.

|  |  |
| --- | --- |
| VAZE *et al.*: LOW-MEMORY CNNs ENABLING REAL-TIME ULTRASOUND SEGMENTATION TOWARDS MOBILE DEPLOYMENT | 1063 |
|  |  |

Thus, the first architecture we propose is identical to the original U-Net but with surplus filters removed (Fig. 1(b)). Reducing the number of filters in each convolutional layer has a drastic effect on the number of network parameters. The number of parameters per layer can be described as follows:

|  |  |
| --- | --- |
| *|θl|* = *K*2*N M* | (3) |

Where *|θl|* describes the number of elements in the *lth* convo-lutional layer’s parameter vector; *N* is the channel-wise width of the input to a convolutional layer; and *M* is the width of the layer’s output. Thus, it can be seen that by halving the number of filters in every convolutional layer (thereby halving both *N* and

* for most layers in the architecture) the number of network parameters quarters. We also remove the 2 *×* 2 convolutions in the original U-Net’s up-sampling path.

We choose a network width of 32 filters per convolutional layer throughout the architecture. Traditionally, the number of filters doubles after down-sampling to preserve the information capacity of later layers. However, we found that this pattern could be broken without affecting the network’s predictive power. However, if the number of filters is reduced too dras-tically, the model performance always drops due to insufficient representational capacity. Note that the exact number of filters is a hyper-parameter which should be tuned to each individual task, and the value chosen here serves as an illustrative example.

It is significant that the network can be drastically thinned without compromising accuracy (see Section V for results on a held-out test set). Many of the CNNs deployed in the medical imaging field have taken inspiration from models developed for general computer vision applications. These networks are typically large (wide and deep) with hundreds of filters per convolutional layer. The most popular architectures, such as VGG16 [18] or ResNet [28], are evaluated on the ImageNet dataset (containing a million natural images with large variation from 1000 classes) and thus require the capacity afforded by numerous parameters to encode this information. However, in the medical field, in which the images are captured under a more constrained environment, the data have already been purified: the distribution of images is far more uniform, with most images for a given task having a qualitatively similar appearance. Thus, we propose that small architectures have sufficient capacity to model trends in medical images, and the filter reduction mostly removes redundant representations from the larger network. Nevertheless, it has been empirically observed that smaller models will almost always come with some performance com-promise [29].

*D. Receptive Field Analysis*

For this task, we found that the size of the *receptive field* of the CNN had a strong effect on model performance. The receptive field of a given neuron in a network describes the size of the region in the input image which can influence the neuron’s activation value. Larger receptive fields allow networks to spatially contextualise predictions. This is particularly impor-tant for ultrasound analysis given the fuzzy boundaries at object edges, shadows and confounding objects in the image. At these

pixel locations, significant spatial and contextual information is required to make an accurate prediction.

The receptive field of a neuron at a later stage of the network can be increased in three ways: (i) increasing the dimensions of the convolutional kernels (*K*); (ii) increasing the number of convolutional layers; (iii) increasing the number of down-sampling layers. As large convolution kernels would signifi-cantly increase the parameter count and hence inference time (note the dependency on the squared kernel size, *K*2, in Equation

(3)), we focused on methods (ii) and (iii). Both of these methods can be encompassed under the term *network depth*. Thus, we experimented on the effect of the number of blocks (*m*) in the encoder and decoder paths. In the experiments detailed in this section, a single block contains two convolutional layers with 3 *×* 3 kernels and 32 filters, followed by a max-pooling (or up-sampling) layer. Each block also contains batch-normalisation between the convolutional and max-pooling layers. The resultant network architecture is illustrated in Fig. 3.

We ranged the number of blocks from zero (wherein the input passes directly to the output block) to seven. Adding more than seven blocks is inappropriate for our task, as the spatial dimensions of the feature maps after down-sampling seven times would be 1 *×* 2 neurons.

The results of these tests are indicated by the blue lines in Fig. 5, which shows the effect of network depth (parametrised by the number of down-sampling blocks, *m*) on Dice coefficient and inference time. Note the plateauing of network accuracy at greater depths, and the drastic deprecation of accuracy with insufficient network depth in Fig. 5(a). The reduction of *m* from 4 to 3 comes with a Dice compromise of approximately 5%. We find that the optimal Dice performance is given by the architecture with *m* = 6.

*E. Depth-Wise Separable Convolutions*

We further reduce model size and CPU inference time by using depth-wise separable convolutions [10] (Fig. 1(c)). This technique factorises the three-dimensional convolutions required for CNN computation into two distinct steps, thereby parametrising the process efficiently. Intuitively, the method is akin to factorising a rank-deficient *n* *×* *n* matrix into a *n* *×* 1 vector and a 1 *×* *n* vector.

In CNNs which operate on 2D inputs, intermediate feature maps have three dimensions: two spatial and one channel-wise. Thus, regular convolution kernels are represented as 3D tensors with square spatial dimensions, *K*2, and a channel-wise dimen-sion equal to the number of input channels, *N* . As each of the

* filters in a given convolutional layer requires one kernel, the number of parameters in a single regular convolutional layer is described by Equation (3). The process is illustrated in Fig. 4(a).

Depth-wise separable convolutions break this process down. They first compute one spatial convolution per input channel (producing an intermediate set of activations) and then com-bine the information from the various input channels using *point-wise* convolutions. Point-wise convolutions are identicalto the regular variant with *K* = 1. As such, the number of parameters required to represent a single separable convolutional

1064 IEEE JOURNAL OF BIOMEDICAL AND HEALTH INFORMATICS, VOL. 24, NO. 4, APRIL 2020

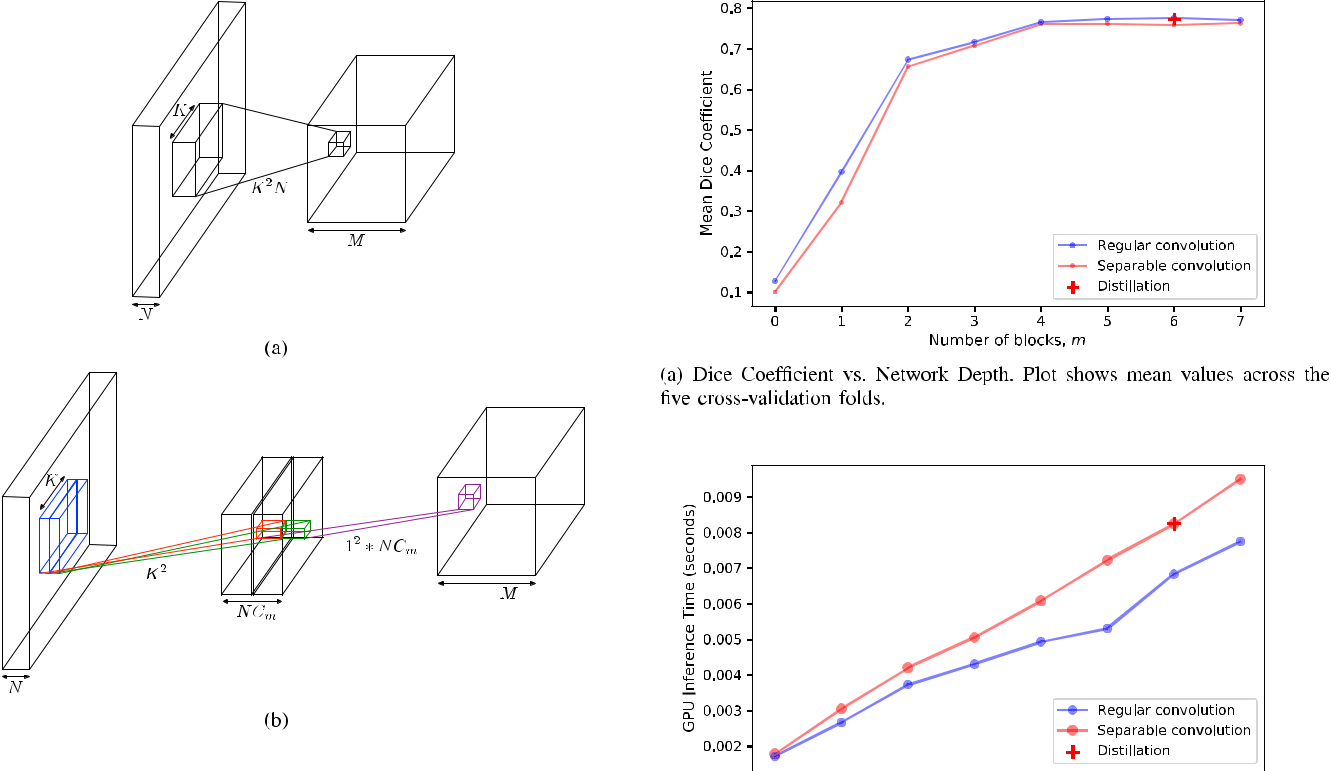


Fig. 4. (a) Regular convolutional layer: Each of the *M* output channels



requires a convolution tensor with *K*2*N* parameters. (b) Depth-wise



separable convolutional layer: Convolution process is decomposed into

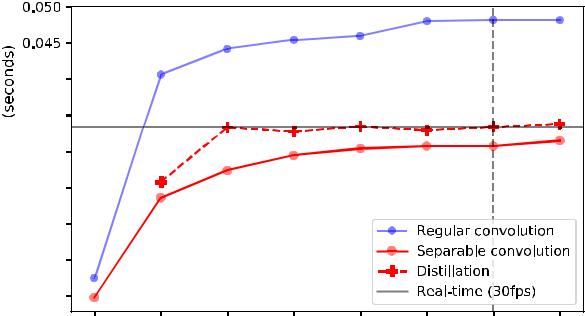


*Cm* spatial convolutions for each of the *N* input channels, followed by *M*



point-wise convolutions (1 *×* 1 spatial dimension) to produce the output.

In this work, *Cm* = 1. [15].



layer is described by Equation (4), and the process illustrated in Fig. 4(b).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2 | ¯ 2 | 2 | *N* +*NM* | (4) |  |
| *|θl|* = *K* | *N* + *K* | *NM⇒K* |  |

¯ = 1

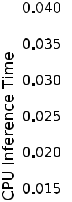
Here, the variable *K* is introduced to highlight that the

second stage of convolutions is identical to a regular convolution with a spatial dimension of 1. Equation (4) demonstrates that separable convolutions reduce the parameter count by breaking the multiplicative interaction between the squared spatial kernel size, *K*2, and the number of output layers, *M* . The technique has been used in the computer vision community to reduce model size and speed up inference [5], [11], [22].

In large architectures, it has been noted that similar accuracies can be achieved for a lower computational cost [11]. Separable convolutions are a low-rank approximation to the regular variant, with previous literature showing that the sum of a number of separable convolutions can losslessly replicate a standard convolution [30]. When using a single separable convolutional layer, however, one must leverage redundancy between feature channels to achieve comparable accuracy. Therefore, in large networks, which inherently have greater potential to accommo-date redundancy, little salient information is lost when using separable convolutions.

Here, we trial the effect of using depth-wise separable convo-lutions in a light-weight architecture, with only *M* = 32 filters per layer. We carry out analogous tests to those in Section IV-B

Fig. 5. Five-fold cross-validation experiments on the effect of network depth (parametrised by the number of blocks, *m*) on the validation accu-racy and inference times for thin regular and separable convolutional networks. Effects of knowledge distillation are shown for *m* = 6. We include CPU inference times for the distilled model at 1 *≤* *m* *≤* 7 to contextualise the 30 fps result at *m* = 6.



(experimenting with a range of architecture depths) with the results shown by the red lines in Fig. 5. Fig. 5(a) shows that even when separable convolutions are used in an already small archi-tecture, they can achieve comparable performance to regular convolutions. Here, the best performing separable architecture performs within 1.8% of the best regular model.

The inference times on GPU and CPU are demonstrated in Figs. 5(b) and 5(c). Interestingly, on a GPU, the regular

|  |  |
| --- | --- |
| VAZE *et al.*: LOW-MEMORY CNNs ENABLING REAL-TIME ULTRASOUND SEGMENTATION TOWARDS MOBILE DEPLOYMENT | 1065 |
|  |  |

convolution network consistently requires less time to compute than the separable architecture, despite the markedly fewer FLOPs required for the latter. There are two reasons for this. Firstly, there exist highly optimised cuDNN kernels (low level CUDA libraries for implementing deep learning on GPUs) for regular 3 *×* 3 convolutions, as they are the most common CNN operation. Secondly, convolutions are implemented on a GPU as a vectorised dot product, with the dot product process being fast and highly parallelised. Thus, for small input tensors (as is the case here with few feature channels) a significant proportion of the computation time is the vectorisation process itself, which must be performed twice for a separable convolution but only once for a regular convolution.

The CPU inference times are more intuitive (Fig. 5(c)), as computation on CPUs is more serialised and thus more closely related to the number of FLOPs required. It is notable that adding additional blocks to an already deep network causes only a small increase in inference time. This is because the later layers are highly down-sampled with small feature maps, and hence do not come with a significant computational cost. Most importantly, however, we note that the separable convolutions require significantly less CPU compute time than the regular variant, with a forward pass completing approximately 35% (17 ms) faster in the setup with *m* *≥* 1. This is advantageous for the clinical setting, which generally relies on CPU hardware (see Section V-D1).

*F. Knowledge Distillation*

Although the performances of the separable convolution mod-els were close to those of the regular architectures, it is clear that the introduction of the technique comes with some compromise in accuracy. Specifically, the separable performance is consis-tently worse across the depth parameter space, with an average reduction in accuracy of 2.2%.

This section describes the efforts made to boost the per-formance of the thin, separable convolution architecture while maintaining the inference speed-up (Fig. 1(d)). The best per-forming regular convolution model occurs with six blocks in the encoder and decoder paths. Hence we aim to boost the performance of the separable convolution model with six blocks to that of its regular convolution counterpart.

The method proposed is *knowledge distillation*, which uses a larger *teacher* model (the thin, regular convolution network) to supervise the training of a lighter *student* network (the thin, sep-arable convolution network). The distillation method involves optimising the student network with respect to the sum of two losses: a hard target loss (*L*1, defined by Equation (1)); and a distillation loss, which encodes information on the teacher’s internal representation.

For classification problems, the distillation loss often uses the teacher’s outputs as ‘soft targets’, in an identical fashion to the manual labels [23]. This works well, as the output of a classification network likely encodes abstract class information useful to the student’s training process. For example, a prediction of ‘car’ may also contain a high score for ‘lorry,’ suggesting their visually similar nature.

However, in the case of segmentation, the teacher’s outputs offer negligible additional information over the manual labels. As an alternative, we create a distillation loss using the teacher network’s intermediate activations, incentivising the student network to recreate the feature maps of the larger architecture. The assumption here is: the optimal parameter configuration in the separable convolution (student) network is that of a low-rank approximation to the regular convolution (teacher) architecture.

To achieve this we optimised the student network with respect to *L*1 + *L*2, where *L*2 is defined as follows for the *ith* training input, *xi*.

|  |  |  |  |
| --- | --- | --- | --- |
| *L*2*,i*(*xi*) = | ˆ | (5) |  |
| *ws*Δ(*φs*(*xi*)*, φs*(*xi*)) |  |

*s∈S*

Here, Δ(*·*) refers to the mean absolute difference (L1-distance) and *S* is the set of all layers supervised by the dis-tillation loss. Intuitively, each term inside the summation takes the mean L1-distance between the *sth* activation of the teacher

network, *s*(*·*), and the corresponding student activation,ˆ*s*(*·*),

*φ* *φ*

weighted by a parameter *ws*. In this work, *S* contains the final convolutional output in each network block.

In order to ensure that every supervised activation contributed equally to the distillation loss, the weightings were set such that all *ws* were inversely proportional to the number of neurons in the *sth* activation. The weightings were then fixed by defining an additional hyper-parameter (*λ*2 = 10), as the sum of all *ws*.

Furthermore, two adjustments were made to the student ar-chitecture to increase its capacity and make it more amenable to distillation (see Fig. 3 for context):

1. The first two separable convolutional layers in the net-work (those in the first block of the encoder path) were replaced with regular convolutional layers.
2. An extra separable convolutional layer was added to the most down-sampled blocks of the network.

The first change was necessary as having separable con-volutions in the initial stages of the network severely limits representational power. For instance, separable convolutions in the first layer (that is, acting directly on the input image) form only one spatial filter (*N* = 1), which is multiplied by *M* = 32 distinct scalars. Conversely, a regular convolutional layer has the capacity to form 32 distinct spatial filters on the input image. This change is common practise when using separable convolutional architectures [5], [11]. Interestingly, however, in preliminary ex-periments without distillation supervision, incorporating regular convolutions in the first two layers generally gave poorer results (see Section V-C).

The second change was used as a way to increase the represen-tational capacity of the student without damaging the inference speed significantly. Extra layers in the most down-sampled blocks act on small feature maps, thus contributing few FLOPs to the inference process.

The effects of the distillation training are shown by the red crosses in Fig. 5. Fig. 5(a) shows that the distilled networks perform almost identically to the best thin regular convolution model (*m* = 6), with the mean Dice coefficient for the net-works being 0.773 and 0.776, respectively. We note here that the proposed distilled network performs within 0.3% of the

1066 IEEE JOURNAL OF BIOMEDICAL AND HEALTH INFORMATICS, VOL. 24, NO. 4, APRIL 2020

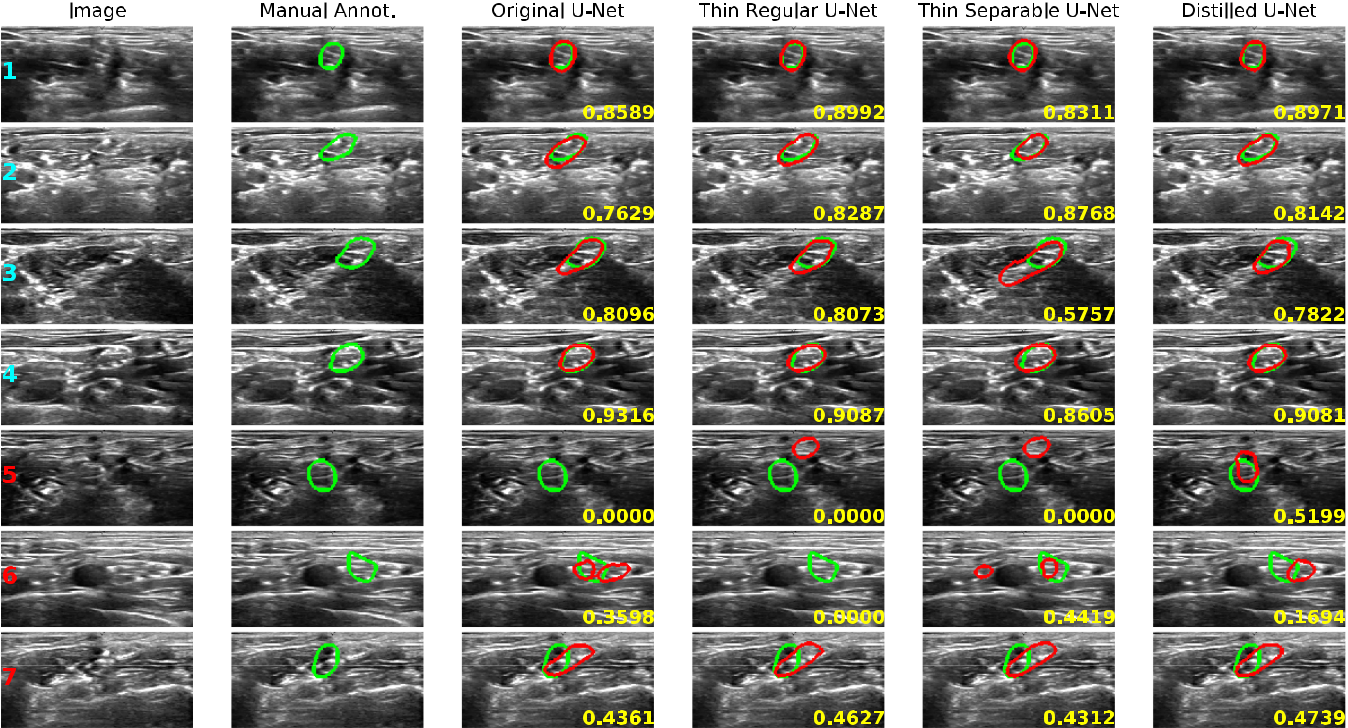


Fig. 6. Sample results of the original U-Net; the thin regular convolution U-Net; the entirely separable convolution network; the separable convolution network adapted for distillation. Green outlines indicate manual annotations, while red outlines indicate CNN predictions. Rows with red labels indicate failure modes (Dice coefficient *<* 0.5) and Dice coefficients of network predictions are given in lower right corners of images.

best regular convolution architecture, as compared to the best separable model which performed within 1.8%.

Furthermore, the architectural changes do not seem to affect the GPU inference times when compared with the separable architecture (Fig. 5(b)), but do slightly increase CPU times (Fig. 5(c)). Specifically, the CPU inference time at *m* = 6 is increased by 2.64 ms (averaged over 270 trials); however we note that the distilled network still desirably operates at 30 fps, meeting our real-time target. We show the CPU inference times of the distilled architecture at other depths to contextualise the 30 fps time in a trend and demonstrate that the result at *m* = 6 is not a chance occurrence (though it is a function of input image size, see Section V-D2).

1. RESULTS

In this section we compare the performance of four models on a held-out test set:

1. *Original U-Net:* Network as presented in [4] with theinclusion of batch-normalisation layers.
2. *Thin Regular U-Net:* A thin U-Net with 32 regular convo-lutional filters per layer, and *six* down-sampling blocks.
3. *Thin Separable U-Net:* A thin U-Net with separable con-volutional layers throughout the architecture.
4. *Distilled U-Net:* A separable convolution U-Net adapted

for knowledge distillation, using the thin regular U-Net as a teacher.

The test set contains 465 slices, with the Dice coefficient selected as the key evaluation metric. We found that the original U-Net required longer to converge than the lighter models,

and so was trained for 160 epochs rather than 100. In this analysis, we focus on the Distilled U-Net, which we propose as the optimal compromise between accuracy and computational complexity.

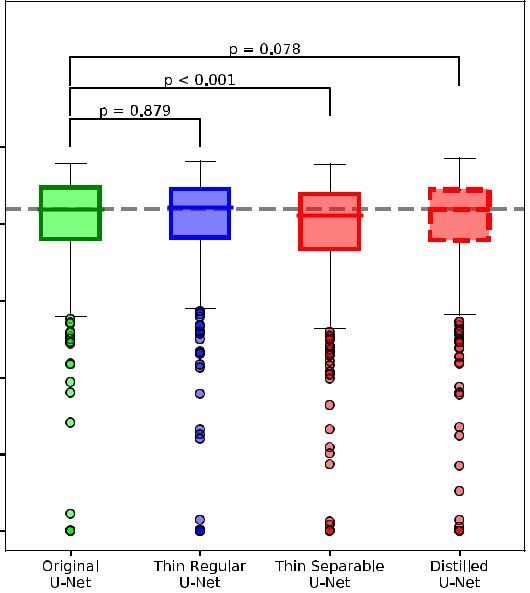
*A. Qualitative Assessment*

Example predictions of the four networks, along with the manual annotations, can be seen in Fig. 6. The top four rows show good results, in which the predictions match the manual annotations well. They show that even in the presence of chal-lenging input images, the networks can produce a reasonable result. The fourth row shows an image with a confounding shape of similar appearance below the target region, while the second row shows an image with an extremely sparse boundary signal.

The benefit of distillation is best demonstrated in the third row, wherein the distilled network significantly outperforms the purely separable architecture. Here, the distilled network (like its teacher) makes a prediction of similar size to the manual annotation; meanwhile, the purely separable network signifi-cantly oversegments due to misleading edges in the input. We also include the fifth row to show that the distilled architecture’s accuracy is not necessarily bounded by that of its teacher. This is because the proposed distillation procedure also optimises the network parameters with respect to the *IoU* score. However, as a consequence, there are cases in which the Thin Separable ar-chitecture outperforms its distilled counterpart, as demonstrated in row 2 and row 6.

|  |  |
| --- | --- |
| VAZE *et al.*: LOW-MEMORY CNNs ENABLING REAL-TIME ULTRASOUND SEGMENTATION TOWARDS MOBILE DEPLOYMENT | 1067 |
|  |  |

TABLE I



PERFORMANCE OF THE FOUR NETWORKS ON A HELD-OUT TEST SET.

HAUSDORFF DISTANCE FIGURES ARE GIVEN NORMALIZED BY THE

PRODUCT OF THE INPUT DIMENSIONS. HERE, ‘ACCURACY’ SPECIFICALLY

REFERS TO THE RAND INDEX

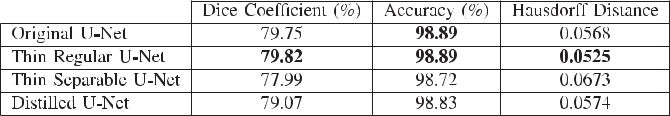


Fig. 7. Box plot of the Dice coefficient of all four networks on a held out test set. Also shown are the results of paired sample t-tests between the original U-Net and our proposed models. We note that the similarity in performance of the proposed Distilled U-Net and the original U-Net.



The bottom three rows show examples of failure modes of the Distilled U-Net (Dice coefficient *<* 0.5). The fifth and sixth rows show examples in which the networks struggle to localise the target region, either due to confounding objects of similar appearance (row 5) or poorly defined edges (row 6). We note with interest that row 6 induces predictions of multiple, distinct structures in the original and Thin Separable U-Nets. This is despite the training set only containing samples with one target region.

Due to the wide range of appearances in the predictions from the four networks, it is difficult to conclude the effects of our proposed methods from qualitative analysis alone. We investigate these effects quantitatively in the next subsection.

*B. Quantitative Assessment*

The Dice performance of the four considered networks on the 465 slices in the test set is illustrated in Fig. 7. The plot shows how the Thin Regular U-Net produces similar results to the original U-Net, both in terms of quartile markers and in the distribution of outliers. Furthermore, the plot nicely demon-strates the compromise in performance of the Thin Separable U-Net compared to its regular convolution counterpart, with the median Dice 2.1% lower. It then also shows the benefits of knowledge distillation, with the median Dice of the Distilled U-Net virtually identical to that of the original U-Net (83.76% and 83.73%, respectively).

Fig. 7 also shows the p-values for paired sample t-tests conducted between the results of our proposed models and the original U-Net. Firstly, it is evident that there is little difference between the performance of the original U-Net and the Thin Regular U-Net. Furthermore, the p-values also show that, at

both the 5% and 1% significance levels, there is evidence to suggest that the predictions from the Thin Separable U-Net are different to those of the original U-Net. However, for the Distilled U-Net, there is insufficient evidence to suggest this. We therefore observe that although the accuracy improvement through the distillation process is small in terms of absolute percentage, it is statistically significant.

Table I codifies the test set performance of the four networks. In these figures, ‘Accuracy’ specifically refers to the Rand Index, summarising the detection rates as (*T P* + *T N* )*/*(*T P* + *F P* + *T N* + *F N* ). Here we note that the Thin Regular U-Net actuallyperforms better by most metrics when compared to the original U-Net. While this falls within the bounds of statistical variation (*p* = 0*.*879 for the Dice score), our experience also suggests that the original U-Net is more prone to overfitting, due to its significantly larger parameter count and capacity. Though this problem could be ameliorated by more careful tuning of the regularisation hyper-parameters (we performed basic grid-search for the models’ hyper-parameters), we note the utility of easy-to-train models for practical clinical deployment.

We highlight that the purpose of this paper is to propose meth-ods which can reduce the memory costs and inference times of neural network models in the context of ultrasound. We propose the distillation method as a way of boosting the performance of these lightweight models in accuracy-critical settings. We suggest that future work selects the most relevant components of this research according to the task under consideration.

*C. Distillation Loss Study*

Our proposed distillation method has two components. We first make minor architectural changes to the Thin Separable U-Net, which make the model more amenable to distillation, followed by training this model with respect to a distillation loss (see Section IV-D for details). Here we conduct an experiment to determine the effect of the loss component.

We do this by implementing the architectural changes to the Thin Separable U-Net and training it solely with respect to the regularised *IoU* score (without the distillation loss, Equation

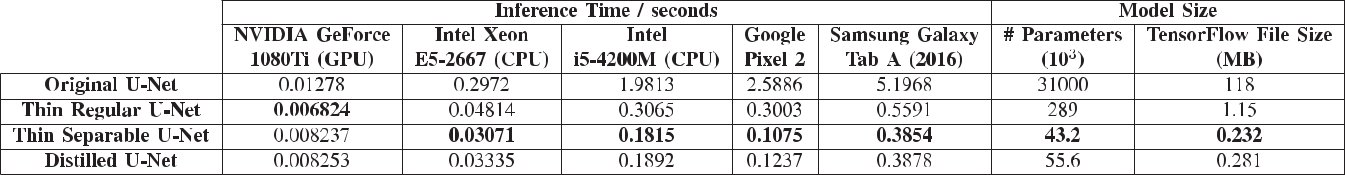
(1)). The mean Dice coefficient on the 465 test slices in this ex-periment is 77.86%, which is comparable to the Thin Separable U-Net’s accuracy of 77.99% (within statistical bounds indicated by a paired sample t-test, *p* = 0*.*801). It is thus clear that the larger contributor to the accuracy boost from the proposed distillation procedure is the extra component in the loss function.

In general, we observe that simply adding extra parame-ters does not necessarily improve accuracy. This is most evi-dently shown by the similar performances of the original U-Net

1068 IEEE JOURNAL OF BIOMEDICAL AND HEALTH INFORMATICS, VOL. 24, NO. 4, APRIL 2020

TABLE II

INFERENCE TIMES AND MODEL SIZES OF THE DISCUSSED CNNS. TIMES SHOWN ARE AVERAGED OVER MULTIPLE TRIALS. RESULTS SHOW THAT THE THIN REGULAR U-NET IS THE FASTEST MODEL ON THE DESKTOP GPU, WHILE THE THIN SEPARABLE U-NET IS THE FASTEST ON ALL OTHER DEVICES



(*≈* 30 M parameters) and Thin Regular U-Net (*≈* 300 K pa-rameters). However, our experience of training these mod-els suggested that the judicious addition of parameters (in the manner detailed in Section IV-D) can aid the distillation training.

*D. Inference Times and Mobile Deployment*

Finally, in this section, we analyse the computational cost of our proposed methods. We analyse the inference times of the four networks on PC hardware, before investigating the performance of the models on mobile devices.

The three leftmost columns of Table II give the inference times of the networks on: a high-end GPU (NVIDIA GeForce 1080Ti); a modern desktop CPU (six-core Intel Xeon E5-2667); and an older CPU chip designed for a laptop (dual-core Intel i5-4200 M). It is evident that the original U-Net is significantly slower than all three proposed networks on all devices. For instance, it is slower than the Distilled U-Net by a factor of 10.5 on the Intel i5 CPU, and by a factor of 8.9 on the Intel Xeon CPU.

We also reiterate the observation that the Thin Regular U-Net actually has a shorter run-time on a GPU than the smaller Thin Separable and Distilled models. However, on the CPU architectures, which are likely more comparable to the hardware available in a clinical setting (see subsection on GPU Avail-ability, Section V-D1), both the Thin Separable and Distilled U-Nets make predictions significantly faster than their regular convolution counterpart. Specifically, on a high-end CPU, the Distilled U-Net operates in real-time at 30 fps, computing 30.8% faster than the Thin Regular U-Net, and only 8.6% slower than the Thin Separable model.

The models were also deployed on two demonstrative An-droid devices: the Google Pixel 2 and the Samsung Galaxy Tab A. The model sizes and inference times are given in the four rightmost columns of Table II. The model size is given both as the number of network parameters, as well as the size of the TensorFlow weight file. It is interesting to note that the Google Pixel 2 yields comparable inference times to the laptop CPU. This is because many modern mobile devices come equipped with dedicated cores for efficient CNN inference, and the latest Android APIs contain libraries for fast neural network deployment [31]. We again highlight the significant inference time gains made by our proposed methods, with the Distilled U-Net predicting 20.9*×* faster than the original U-Net on the Google device, and 13.4*×* faster on the Samsung device.

Furthermore, we note the memory savings afforded by our proposed methods: the Distilled U-Net requires 420*×* less space in memory than the original U-Net. This could be especially important for practical mobile deployment, in which the memory budget is often limited.

1. *GPU Availability:* It should be noted that most moderndevices do come with integrated GPUs, though these are smaller and less powerful than dedicated graphics cards (which still remain relatively uncommon). Thus, on most consumer devices, CNN inference must still be conducted on the CPU. On PC hardware, for instance, TensorFlow’s GPU accelerated package requires a dedicated CUDA-enabled graphics card to run. Con-sequently, the utility of CPU catered algorithms still remains high.

The conditions are slightly different on mobile devices. Most mobile devices now come with integrated graphics chips and even neural processing units (NPUs), along with operating system (OS) functionality to accelerate deep learning on this hardware. However, these chips must share RAM with the rest of the OS and consume significant power. Therefore, under the hood, the OS (such as the TensorFlow mobile library) distributes the workload among all available hardware [32].

It is difficult to know exactly how this is done without inspect-ing the software compilers themselves. However, our empirical results show that all of our proposed models give a significant inference time boost over the original U-Net on mobile devices. Furthermore, we show that our separable models also run signifi-cantly faster than their regular convolution counterpart on mobile devices (for instance, roughly 3*×* faster on the Google Pixel 2). As we observed that separable convolutions did not offer inference time gains when tested on desktop GPU hardware, this suggests that much of the computation still occurs on the mobile devices’ CPU.

1. *Effect of Image Size:* Finally, we comment on the imagesize selected for these experiments. The size of 128 *×* 256 pixels was chosen to best approximate the original aspect ratio of the data (420 *×* 580) while also allowing a reasonable batch size (32) for CNN optimisation, and for having ‘power of two’ dimensions to facilitate repeated downsampling in the models. Downsampling the input images further would speed up infer-ence as the number of FLOPs required for a CNN’s forward pass (and thus the CPU inference time) scales linearly with the number of pixels in the image. However, this also comes with accuracy degradation.

To demonstrate this, we downsampled the input images to

64 *×* 128 pixels and trained the original U-Net architecture on the resultant dataset. The accuracy of the model was degraded, with

|  |  |
| --- | --- |
| VAZE *et al.*: LOW-MEMORY CNNs ENABLING REAL-TIME ULTRASOUND SEGMENTATION TOWARDS MOBILE DEPLOYMENT | 1069 |
|  |  |

a mean Dice coefficient of 77.51% on the 465 test slices. Notably, this is lower than the least accurate of our proposed models (the Thin Separable U-Net). Furthermore, the inference time on the Intel Xeon CPU was predictably reduced by a factor of four (to 0.0681 s), still making it slower than all of our proposed models. Thus we can demonstrate that, though downsampling is a quick and easy way of boosting inference times, the gains are significantly less than with our proposed methods, with greater performance compromise. Specifically, downsampling the input data led to 1.56% worse Dice performance than the Distilled U-Net, while also requiring over twice as long to make predic-tions. We also note that downsampling the images has no effect on the number of model parameters, making the original U-Net architecture still have a large memory footprint.

VI. CONCLUSION

In this paper, we have tackled the problem of real-time ultra-sound analysis on CPUs, thereby catering our CNN architectures to the clinical setting. Our methods involve computationally efficient neural networks, adapting the popular U-Net for fast inference by using a small number of filters and employing depth-wise separable convolutions.

Furthermore, we demonstrate the accuracy compromise with separable convolutions in smaller architectures, and remedy the issue with a novel knowledge distillation technique. We show that, for a negligible compromise in test set segmentation accu-racy, our final proposed model runs 9*×* faster than the standard U-Net on a CPU (at a rate of 30 fps) and requires 420*×* less space in memory.

Finally, we conclude by noting that the methods and architec-ture alterations presented in this work have been demonstrated on the segmentation of nerve clusters in ultrasound, but could be readily integrated into many existing CNN architectures, enabling fast inference and low memory costs for a range of tasks. We hope that future work will build on these methods to facilitate real-time ultrasound analysis at point-of-care, without sacrificing accuracy.

ACKNOWLEDGMENT

The authors are grateful for support from the Royal Academy of Engineering under the Engineering for Development Research Fellowship scheme.

REFERENCES

1. A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Proc. Conf. Neural Inf.* *Process. Syst.*, 2012, pp. 1097–1105.
2. D. Ciregan, U. Meier, and J. Schmidhuber, “Multi-column deep neural networks for image classification,” in *Proc. IEEE Conf. Comput. Vision* *Pattern Recognit.*, 2012, pp. 3642–3649.
3. G. Litjens *et al.*, “A survey on deep learning in medical image analysis,” *Med. Image Anal.*, vol. 42, pp. 60–88, 2017.
4. O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical image segmentation,” in *Proc. Int. Conf. Med. Image* *Comput. Comput.-Assisted Intervention*, 2015, pp. 234–241.
5. A. G. Howard *et al.*, “MobileNets: Efficient convolutional neural networks

[for mobile vision applications,” 2017. [Online]. Available: https://arxiv.](https://arxiv.org/abs/1704.04861) [org/abs/1704.04861](https://arxiv.org/abs/1704.04861)

1. C. F. Baumgartner *et al.*, “SonoNet: Real-time detection and localisation of fetal standard scan planes in freehand ultrasound,” *IEEE Trans. Med.* *Imag.*, vol. 36, no. 11, pp. 2204–2215, Nov. 2017.
2. “Basic physical principles of ultrasound,” *LCMC Pre-reading*,

2013.[Online].Available:[http://www.dcconferences.com.au/](http://www.dcconferences.com.au/lcmc2012/pdf/Phlebology_PrereadingFriday.pdf)

[lcmc2012/pdf/Phlebology\_PrereadingFriday.pdf, Accessed on: 2019.](http://www.dcconferences.com.au/lcmc2012/pdf/Phlebology_PrereadingFriday.pdf)

1. D. Hyun, “Customizable ultrasound imaging in real-time using a GPU-accelerated beamformer,” in *Lecture Slides*. Stanford, CA, USA: Stanford Univ. Press.
2. M. Havaei *et al.*, “Brain tumor segmentation with deep neural networks,” *Med. Image Anal.*, vol. 35, pp. 18–31, 2017.
3. L. Sifre and S. Mallat, “CMAP rigid-motion scattering For image classifi-cation,” Ph.D. dissertation, Ecole Polytechnique, Palaiseau, France, 2014.
4. F. Chollet, “Xception: Deep learning with depthwise separable convo-lutions,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2016,
   1. 1800–1807.
5. “Kaggle competitions, ultrasound nerve segmentation,” 2016. [Online]. Available: <https://www.kaggle.com/c/ultrasound-nerve-segmentation>
6. Y. Wang, C. Wei, Z. Wang, Q. Lu, and C. Wang, “A more streamlined u-net for nerve segmentation in ultrasound images,” in *Proc. Chin. Autom.* *Congr.*, 2018, pp. 101–104.
7. Y. Weng, T. Zhou, Y. Li, and X. Qiu, “Nas-unet: Neural architecture search for medical image segmentation,” *IEEE Access*, vol. 7, pp. 44 247–44 257, 2019.
8. S. Vaze and A. I. L. Namburete, *Segmentation Of Fetal Adipose Tissue Us-ing Efficient Cnns For Portable Ultrasound*. Cham, Switzerland: Springer,2018, pp. 55–65.
9. J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proc. IEEE Conf. Comput. Vision Pattern* *Recognit.*, 2015, pp. 3431–3440.
10. M. Lin, Q. Chen, and S. Yan, “Network in network,” in *Proc.* [*Int. Conf. Learn. Representations*, 2014. [Online]. Available: https://](https://dblp.org/rec/bibtex/journals/corr/LinCY13)[dblp.org/rec/bibtex/journals/corr/LinCY13](https://dblp.org/rec/bibtex/journals/corr/LinCY13)
11. K. Simonyan and A. Zisserman, “Very deep convolutional net-works for large-scale image recognition,” in *Proc. Int. Conf.* [*Learn. Representations*, 2015. [Online]. Available: https://dblp.](https://dblp.org/rec/bibtex/journals/corr/SimonyanZ14a)[org/rec/bibtex/journals/corr/SimonyanZ14a](https://dblp.org/rec/bibtex/journals/corr/SimonyanZ14a)
12. W. Chen *et al.*, “Compressing neural networks with the hashing trick,” in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 2285–2294.
13. J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, “Quantized convolutional neural networks for mobile devices,” in *Proc. IEEE Conf. Comput. Vision* *Pattern Recognit.*, 2016, pp. 4820–4828.
14. I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, “Binarized neural networks,” in *Proc. Conf. Neural Inf. Process. Syst.*, 2016, pp. 4107–4115.
15. L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-decoder with atrous separable convolution for semantic image segmenta-tion,” in *Proc. Eur. Conf. Comput. Vision*, 2018, pp. 833–851.
16. G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” in *Proc. Conf. Neural Inf. Process. Syst. Deep Learning Work-shop*, 2015. [Online]. Available: <https://research.google/pubs/pub44873/>
17. J. Johnson, A. Alahi, and L. F.-Fei, “Perceptual losses for real-time style transfer and super-resolution,” in *Proc. Eur. Conf. Comput. Vision*, 2016.
18. K. Mangalam and M. Salzmann, “On compressing u-net using knowledge distillation,” 2018. [Online]. Available: <https://arxiv.org/abs/1812.00249>
19. I. Loshchilov and F. Hutter, “SGDR: Stochastic gradient descent with warm restarts,” in *Proc. Int. Conf. Learn. Representations*, 2017. [Online]. Available: <https://dblp.org/rec/bibtex/conf/iclr/LoshchilovH17>
20. S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Proc. Mach. Learn. Res.*, 2015, vol. 37, pp. 448–456.
21. K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2016,
    1. 770–778.
22. C. Szegedy *et al.*, “Going deeper with convolutions,” in *Proc. IEEE* *Conf. Comput. Vision Pattern Recognit.*, 2015. [Online]. Available:<https://research.google/pubs/pub43022/>
23. J. Guo, Y. Li, W. Lin, Y. Chen, and J. Li, “Network decoupling: From regular to depthwise separable convolutions,” Tech. Rep., Inst. Interdisci-plinary Inf. Sci., Tsinghua Uni., China, 2018.
24. [“Neural Networks API -Android NDK,” [Online]. Available: https://](https://developer.android.com/ndk/guides/neuralnetworks/)

[developer.android.com/ndk/guides/neuralnetworks/, Accessed on: 2019.](https://developer.android.com/ndk/guides/neuralnetworks/)

1. [“TensorFlow Lite Converter,” [Online]. Available: https://www.](https://www.tensorflow.org/lite/convert)

[tensorflow.org/lite/convert, Accessed on: 2019.](https://www.tensorflow.org/lite/convert)