IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 14, 2020, accepted July 24, 2020, date of publication July 29, 2020, date of current version August 10, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3012722

Efficient Biomedical Image Segmentation on EdgeTPUs at Point of Care

ANDREAS M. KIST AND MICHAEL DÖLLINGER
Division of Phoniatrics and Pediatric Audiology, Department of Otorhinolaryngology, Head and Neck Surgery, University Hospital Erlangen, Friedrich-Alexander-University Erlangen-Nürnberg, 91054 Erlangen, Germany

Corresponding author: Andreas M. Kist (andreas.kist@uk-erlangen.de)

The work of Andreas M. Kist was supported in part by Bundesministerium für Wirtschaft und Energie (BMWi) under Grant ZF4010105/BA8; and in part by the Joachim-Herz-StiftungAdd-On Fellowship. The work of Michael Döllinger was supported in part by BMWi under Grant ZF4010105/BA8; and in part by the German Research Foundation Deutsche Forschungsgemeinschaft (DFG) under Grant DO1247/8-1 323308998.

ABSTRACT The U-Net architecture is a state-of-the-art neural network for semantic image segmentation that is widely used in biomedical research. It is based on an encoder-decoder framework and its vanilla version shows already high performance in terms of segmentation quality. Due to its large parameter space, however, it has high computational costs on both, CPUs and GPUs. In a research setting, inference time is relevant, but not crucial for the results. However, especially in mobile, clinical applications a light and fast variant would allow deep-learning assisted, objective diagnosis at the point of care. In this work, we suggest an optimized, tiny-weight U-Net for an inexpensive hardware accelerator. We first mined the U-Net architecture to reduce computational complexity to increase runtime performance by simultaneously keeping the accuracy on a high level. Using an open, biomedical dataset for high-speed videoendoscopy (BAGLS), we show that we can dramatically reduce the parameter space and computations by over 99.8% while keeping the segmentation performance at 95% of our baseline. Using a custom upscaling routine, we further successfully deployed our optimized U-Net to an EdgeTPU hardware accelerator to gain cost-effective speed improvements on conventional computers and to showcase the applicability of EdgeTPUs for biomedical imaging processing of large images on portable devices. Combining the optimized architecture and the EdgeTPU, we gain a speedup of >79-times compared to our initial baseline while keeping high accuracy. This combination allows to provide immediate results to the clinician, especially in constrained computational environments, and an objective diagnosis at the point of care.

INDEX TERMS EdgeTPU, convolutional neural network, coral, semantic segmentation.

I. INTRODUCTION

Image processing and analysis becomes increasingly important to be performed efficiently and accurately, and ideally physically close to the device where the image is acquired. For example, in autonomous driving, camera images need to be processed in real-time in the car to detect dangerous situations, such as a passing subject, in order to react with almost no latency. Especially mobile environments have strong limitations in power consumption and computational efficiency, and mostly lack any on-site sophisticated image processing. This is also true for portable biomedical imaging devices, such as laryngeal videoendoscopy systems. Here, latest convolutional neural networks for image analysis and classification are rarely applied, as they need special environments, such as graphical processing units (GPUs), to operate decently. The classical modus operandi is to transfer the data to a specific data processing unit, analyze the data and report the result back. This is not only potentially affecting privacy and exhibits data protection issues, but also involves transferring large amounts of data, which is almost impossible in remote settings. Therefore, there is an urgent need to bring efficient biomedical image processing to the point of care.

Laryngeal high-speed videoendoscopy (HSV) is a clinical imaging procedure using a mobile imaging unit that has since decades a bottleneck in immediate processing of large, acquired data. In a typical examination, several thousands of frames are acquired using a high-speed camera connected to an oral or nasal endoscope (Figure 1) resulting in a view from above onto the oscillating vocal folds. The standard

The associate editor coordinating the review of this manuscript and approving it for publication was Larbi Boubchir.

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

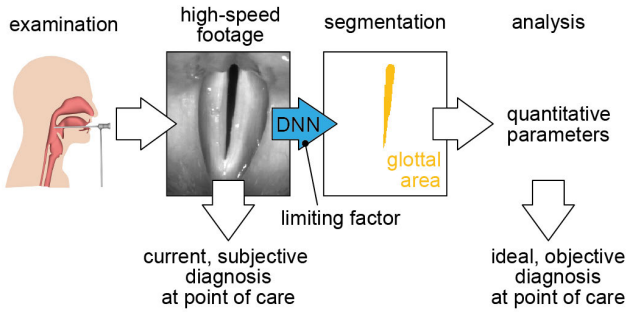

FIGURE 1. High-speed video endoscopy and segmentation task, with current and ideal diagnosis at point of care. A patient examination results in thousands of endoscopic video frames. For every frame, the glottal area is segmented by a deep neural network for further clinical parameter computation to ideally result in an objective diagnosis.

analysis procedure is the frame by frame segmentation of the glottal area, i.e. the opening between the vocal folds (Figure 1). With this procedure, we are able to quantify vocal fold oscillations directly from a HSV recording. The resulting oscillation behaviors are capable to describe and diagnose voice disorders, such as functional dysphonias [1]–[3]. However, HSV is rarely applied in a clinical setting, because the current state-of-the-art data analysis is cumbersome and tedious, performed by specialists and not directly at the point of care [4]–[6]. As data cannot be analyzed in time, the rich and quantifiable, objective HSV data is not considered at all in the diagnosis, rendering it purely subjective.

We and others recently showed that training and applying deep neural networks to segment the glottal area is feasible and became the state-of-the-art segmentation technique [7]–[10], outperforming previously described methods for automatic glottis segmentations, such as 3D active contours [11] or threshold-based techniques combined with level set methods [12]. Nevertheless, the inference time/latency spent per frame on a consumer-grade CPU, as normally available at the point of care, is rather large (around 2 s per frame), rendering it almost impossible to analyze a 1,000 frame long video (an around 250 ms long recording) in a reasonable amount of time (ca. 33 min). The rationale would be to provide a neural network that has increased performance on a consumer-grade CPU or affordable external accessories, without losing significantly segmentation accuracy to ensure a reliable diagnosis.

In this study, we perform an in-depth analysis of the U-Net architecture in terms of inference speed and computational load by reducing the amount of computations and parameters in the model. Our main contributions are summarized as follows:

- We found that the second convolutional layer in encoder and decoder is not required in the U-Net architecture for high accuracy segmentations and only residual propagation is relevant, not its type. Replacing the activation function improves the network accuracy.
- We were able to reduce the total parameter (and thus computational) space by over 99.8 %, while keeping 95 % of the baseline accuracy as measured by the intersection over union.
- We introduced a custom upscaling routine that ensures large images are upscaled on the EdgeTPU and that is relevant across multiple architectures
- By porting our modified, optimized architecture to an EdgeTPU hardware accelerator, we were able to analyze a typical 1,000 frame long dataset in less than 25 s resulting in a 79× speed improvement compared to our initial baseline.

Taken together, this allows the immediate, local data analysis, and thus, a diagnosis based on objective parameters directly at the point of care.

II. RELATED WORK

Increasing neural network performance is of general interest. Especially on mobile devices, where computational power is limited, computationally efficient networks are desired [13], [14]. We utilize several of these key ideas here, such as separable convolutions [15], to reduce the computational load and the parameters dramatically.

Despite the fact that neural architecture search (NAS) is a common way of optimizing architectures, even for mobile applications [16]–[19], we consciously decided to mine the U-Net to identify hot-spots and operations that are crucial to maintain U-Net accuracy. We therefore thought to optimize the U-Net itself, with features known from different previous works, and to systematically investigate each part of the network. Improving U-Nets in terms of efficiency is rarely performed, compared to accuracy, but has been proposed recently by coupling stacked U-Nets for landmark detection together with evaluating several quantization approaches [20] or to use a very reduced U-Net for low power satellite segmentation [21].

Other encoder-decoder networks similar to, derived from or inspired by the U-Net with superior performance in biomedical imaging have been proposed, such as Linknet [22], V-Net [23], nnU-Net [24], U-Net++ [25] and Attention U-Net [26]. In Linknet, the encoding layers feed forward the residuals, and use special convolutional layer structures in encoder and decoder. Further, instead of concatenate the residuals, Linknet adds the residuals from encoder to decoder block, thus, removing another source for trainable parameters. In U-Net++, the skip connections were intensively mined yielding an architecture outperforming the classical U-Net in 2D and 3D medical datasets. In our study, we investigate different ideas derived from these works, such as residual propagation.

Porting a neural network to hardware accelerators has been performed for various architectures [27]. This is especially common for FPGAs [28], [29], and used for example in hand tracking [30] or language processing [31]. Special attention is also on adapting key principles in neural network architectures, such as depth-wise convolutions for FPGAs [32] or quantized-based operations, such as binary neural networks [33]. In contrast to general,

"all purpose" GPUs, tensor-processing units (TPUs) are specialised on matrix operations, such as multiplications and additions, as massively used in neural networks. EdgeTPUs use int8 and int16 data types for computations, can compute up to 4 trillion operations per second (TOPS), and has a power efficiency of 2 TOPS per Watt. Recently, EdgeTPUs are available as additional hardware accelerators that can easily be added to any computer using conventional USB 3.1 ports. Similar to other hardware architectures, neural networks have to be adjusted to successfully deploy it. This can lead to main changes in the neural network architecture, as shown for the EfficientNet [34] in this blog post [35]. Still, the deployment procedure is more streamlined and efficient compared to deploying a neural network to a FPGA (see also Methods).

III. METHODS

We trained deep neural networks using TensorFlow 1.14 [36] using the high-level Keras package. All layers used are found in the naive TensorFlow library or are derived thereof. For training, we used the Dice loss [23] and optimized the network using the Adam classifier [37] using a cyclic learning rate between 10^{-3} and 10^{-6} [38]. The BAGLS dataset [7] served as training, validation and test image set. We re-scaled all images to 512 × 256 px for training, to avoid losing segmentation accuracy by shrinking the image, as previous works have shown that image-derived parameters are highly resolution dependent [39], and to allow the combination of several image sources in the same batch. We normalized images in general to a range between −1 and 1, and for EdgeTPU-deployed networks to a range between 0 and 1. We trained models on a Titan RTX for maximal 25 epochs and chose the network that performed best on the validation data. All networks were k-fold cross-validated (k=3) by shuffling training and validation set, and all networks were evaluated on both, validation and test data set. As we see similar behavior of validation and test data, we decided to show validation data for visualization purposes. Figures showing the test data are nevertheless provided in the Supplementary Material. CPU performance was tested on a Xenon Silver 4116 at 2.10 GHz. To utilize the EdgeTPU built into the Coral USB Accelerator (Google), we enabled quantization aware training (ready for uint8 conversion). Next, we saved our Keras models to TensorFlow protobuf format and further converted these files to TensorFlow Lite file format with 8 bit unsigned integer quantization using TensorFlow 1.15. Then, we compiled the TensorFlow Lite files with the EdgeTPU compiler to map operations to the EdgeTPU. Notably, the EdgeTPU does not yet support all TensorFlow operations at time of publication, such as dilation rate and index unpooling, therefore we adapted accordingly models to allow successful deployment, if possible. As evaluation metric we used the intersection over union [40] (IoU) that compares two segmentation maps. The IoU ranges from 0 (no overlap) to 1 (perfect overlap), suggesting that higher IoU scores indicate better network performance. If not otherwise stated, we refer with IoU to the IoU on the validation set.

Clinical parameters for validation, i.e. fundamental frequency (F_0), jitter and shimmer were calculated as described elsewhere [41]. Data used for the validation section were acquired using a commercially available system (KayPentax HSV 9710) that provides a maximal resolution of 512 × 256 px at 4,000 frames per second and were already analyzed in another study [39]. We selected a 1,000 frame long subset of the recordings, segmented the glottal area using deep neural networks developed in this study, and analyzed the resulting glottal area waveform (GAW, sum of segmented pixels per frame). First, individual cycles were detected in the GAW by finding maxima in the periodic signal [42]. Next, F_0 was determined by the average period duration of the detected maxima T (eq. 1).

$$F_0 = \frac{1}{N}\sum_{i=1}^{N}\frac{1000}{T_i} \tag{1}$$

Jitter and shimmer in percent were computed as follows:

$$\text{Jitter}[\%] = \frac{\frac{1}{N-1}\sum_{i=1}^{N-1}|T_i - T_{i-1}|}{\frac{1}{N}\sum_{i=0}^{N-1}T_i}\cdot 100, \tag{2}$$

$$\text{Shimmer}[\%] = \frac{\frac{20}{N-1}\sum_{i=0}^{N-2}|\log_{10}\left[\frac{A_i}{A_{i+1}}\right]|}{\frac{20}{N}\sum_{i=0}^{N-1}|\log_{10}A_i|}\cdot 100, \tag{3}$$

where T_i are the detected maxima in time, and A_i their respective amplitude.

IV. RESULTS

We started with a vanilla U-Net as described in [43] (Figure 2). We changed the padding option to "same", to easily down- and upscale our image. The transposed convolutional layers were replaced by simple upsampling operations followed by a normal convolutional layer to further decrease

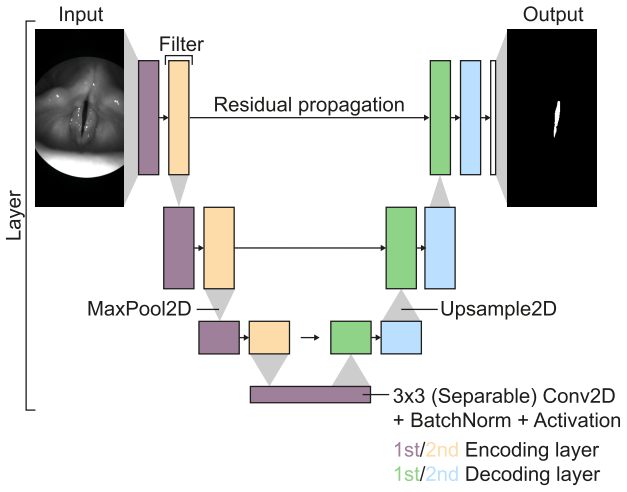

FIGURE 2. Our U-Net baseline variant is based on an encoder/decoder architecture that contains two convolutional layers in each main layer, followed by a MaxPool2D and Upsample2D operation in the encoder and decoder, respectively. Residuals are propagated from encoder to decoder by default via concatenation.

the parameter space. We further added Batch Normalization before the activation in each convolutional layer, similar to [44], and thus, removing the need of a bias term in each convolutional layer. Throughout the network, we used 3×3 convolutions similar to [43].

A. REDUCING LAYERS AND FILTERS INCREASES PERFORMANCE

The U-Net consists of an encoder-decoder structure with by default four depth layers L in encoder and decoder (Figure 2). In the vanilla U-Net, the filter number for each convolutional layer is calculated using the following equation:

$$f_{layer} = f_{base} \cdot 2^{L-1}, \quad \text{with} f_{base} = 64$$

That means, that the two convolutional layers in depth layer $L = 1$ have $f_{layer} = 64$ filters, in depth layer $L = 2$ 128 filters, and so on. An obvious way to downscale the network is by reducing the amount of depth layers L in the encoding and decoding network (in the vanilla U-Net four), as well as changing the base filter count f_{base} as introduced above. Further, one can adjust each filter number in each filter individually, however, in this study we focused on changing solely f_{base}. In the following, we are using the U-Net with $L = 4$ and $f_{base} = 64$ with the settings described above as baseline.

By alternating the number of layers and the base filter count f_{base}, we see a strong dependence of the IoU, our evaluation metric, on the depth of the network (Figure 3, Supplementary Figure 1a). Three layers perform slightly worse than four layers, however, only two layers dramatically perform worse. As depth is in line with detection of high-level features [45], we expected that relationship. However, there are three aspects we noted: first, already a shallow network with only two depth layers $L = 2$ and $f_{base} = 8$, consisting of only 21,088 parameters (0.1% of our baseline U-Net, Table 1), is able to gain a high IoU score of 0.792 on the validation set. Even more striking is that a neural network that consists of only 1,384 trainable parameters

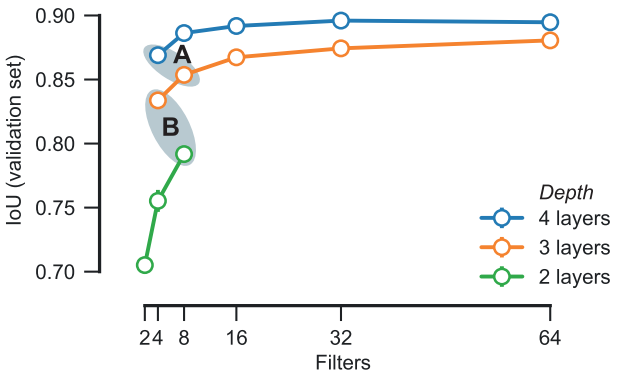

FIGURE 3. Segmentation accuracy is stable across most layer and filter settings. IoU: Intersection over Union. Gray ellipses (noted A and B) indicate two settings with almost identical number of parameters.

TABLE 1. Trainable parameters with different layer L and filter f_{base} configurations.

Layers (L)	f_{base}	Parameters (% of 4L/64F)	IoU (val)	IoU (test)
4	64	21,986,304 (100.0%)	0.895	0.770
	32	5,497,984 (25.0%)	0.896	0.764
	16	1,377,024 (6.3%)	0.892	0.773
	8	345,216 (1.6%)	0.887	0.763
	4	86,480 (0.4%)	0.869	0.760
3	64	5,463,552 (24.8%)	0.881	0.748
	32	1,366,656 (6.2%)	0.874	0.753
	16	342,912 (1.6%)	0.867	0.751
	8	86,208 (0.4%)	0.854	0.753
	4	21,648 (0.1%)	0.834	0.739
2	8	21,088 (0.1%)	0.792	0.684
	4	5,360 (0.02%)	0.755	0.643
	2	1,384 (0.006%)	0.705	0.615

(0.006% of baseline) is still able to converge to a decent IoU score on the BAGLS dataset.

Second, when keeping the number of layers stable and varying f_{base}, the IoU score slowly declines with an abrupt effect at around $f_{base} = 8$. When comparing the four f_{base} network to the 64 f_{base} network with $L = 4$, the validation IoU is only slightly reduced (IoU = 0.887 and 0.895, respectively), but we save 98.4% of the parameters. Further, the training time per epoch is greatly reduced from 40.57 min (64 f_{base}) to 9.37 min (8 f_{base}).

Third, the effect of depth is especially obvious when comparing these two setting combinations: 4L/4F, 3L/8F and 3L/4F, 2L/8F (xL/yF in short notation for $L = x$ and $f_{base} = y$). Both combinations feature around the same amount of parameters (ca. 86,000 parameters, 0.4% of baseline, and ca. 21,000 parameters, 0.1% of baseline, respectively), but have notably deviations in their IoU (0.869 vs 0.854, and 0.834 vs. 0.792, respectively). This is also highlighted in Figure 3 with grey ellipses labelled with A and B. We found that the architecture consisting of four main layers together with eight f_{base} (4L/8F) has the best trade-off between speed, accuracy and number of parameters involved. Thus, we focus on investigating the architecture with this setting, and as a comparison we use 3L/8F, 4L/4F and 3L/4F.

B. STACKED CONVOLUTIONS ARE NOT A NECESSITY

We next tested if the second convolutional layer in the encoding and decoding pathway is necessary (Figure 2, orange and light-blue convolutional blocks). We abbreviate convolutional layers in encoder as "E" and in decoder as "D", their removal with "x". By removing either one (Ex/DD or EE/Dx for encoding or decoding layer, respectively) or both (Ex/Dx), we do observe a small decline, but overall a rather stable level of accuracy (Figure 4, Supplementary Figure 1b, exemplary for 4L/8F) when using ordinary two-dimensional convolutions. This suggests, that additional layers on the same depth level are slightly beneficial for the network, however, we do not see a necessity of the second encoder or decoder layer in glottis segmentation. In training time, we see improvements ranging from 8.37 min (EE/DD), over 7.73 min (Ex/DD, EE/Dx) to 6.26 min (Ex/Dx) for the 4L/8F configuration.

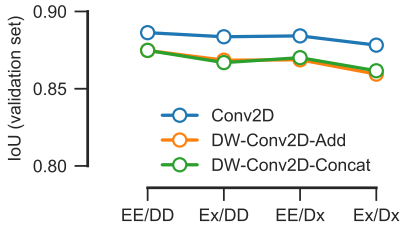

FIGURE 4. Effect of second convolutional block in encoder (EE and Ex) and decoder (DD and Dx), for ordinary convolutions and depth-wise (DW) convolutional layers.

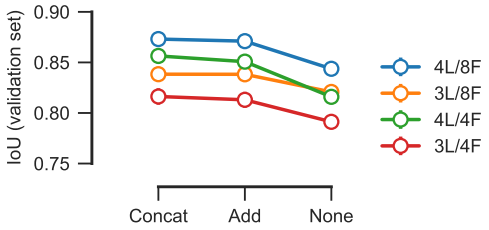

FIGURE 5. Residual propagation.

We also observe similar results for the other configurations (3L/8F, 4L/4F, 3L/4F).

C. DEPTH-WISE CONVOLUTIONS ONLY AFFECT LITTLE PERFORMANCE

Replacing ordinary convolutional layers with separable convolutional layers (a depth-wise convolution followed by a point-wise convolution) is a common computational trick to enhance efficiency. First described by Sifre [15] and widely used in MobileNets [13], separable or depth-wise convolutions are now part of various architectures [46]–[49].

We therefore replaced each ordinary convolutional layer by a separable convolutional layer except of the very first convolutional block in the encoder and the last convolutional layer that results in the segmentation mask. We found that we further reduce massively our parameter space (Table 2), as expected. Notably, the segmentation accuracy was only little affected, so we decided to keep the separable convolutions in our optimized architecture (Figure 4). Interestingly, using separable convolutions we do not observe an effect in removing either of the second encoder/decoder convolutional layer. There is only a slight decrease in performance when removing both of the second convolutional layers (Figure 4).

TABLE 2. Parameters in separable convolutions.

L	f_{base}	Parameters (% of Conv2D equiv)	IoU (val)	IoU (test)
4	8	55,384 (16,0%)	0.875	0.761
	4	15,596 (18.0%)	0.856	0.747
3	8	22,904 (26,6%)	0.838	0.743
	4	6,380 (29.5%)	0.816	0.719

D. RESIDUAL PROPAGATION IS IMPORTANT, NOT THE PROPAGATION METHOD

We next investigated if any residual propagation is important, and if so, which residual propagation method performs better. We found that no residual propagation worsens the results dramatically (Figure 5, Supplementary Figure 1c). Next, we compared the classical propagation, i.e. concatenation, [43] to simply adding the encoder residuals to the corresponding decoder block as used in the LinkNet architecture [22]. We found that both, adding or concatenating residuals, are improving the network performance. Notably, in some cases we even achieved better results by adding the residuals (Figure 5). Additionally, by using addition instead of concatenation we reduced the parameter space from 55,384 to 46,520 parameters, i.e. by 16.0 % in the 4L/8F architecture.

E. SWISH ACTIVATION FUNCTION IMPROVES PERFORMANCE

Changing the activation function in convolutional layers has shown dramatic training and performance improvements [50]. As default, we were using the ReLU activation function (eq. 4). As the ultimate goal is to deploy our neural network to a hardware accelerator that operates on uint8 models, we tested if ReLU6 (eq. 5), a clipped variant of the ReLU function often used in neural networks optimized for mobile phones, performs equally to the unconstrained ReLU function. We further tested the h-swish function (eq. 6) [50], as it becomes popular in the use of MobileNets [47], EfficientNets [34] and semantic segmentation [51]. For comparison, all activation functions are shown in Figure 6a.

$$\text{ReLU}(x) = \max(0, x) \tag{4}$$

$$\text{ReLU6}(x) = \min(\max(0, x), 6) \tag{5}$$

$$\text{h-swish}(x) = x \cdot \frac{\text{ReLU6}(x+3)}{6} \tag{6}$$

When changing the activation function in each convolutional layer to ReLU6, we did not determine large differences, suggesting that our architectures are able to perform the segmentation task in this constrained boundary (Figure 6). However, when training the network with h-swish, we observed that h-swish performs slightly better than ReLU and ReLU6 (0.877 h-swish vs 0.871 and 0.874 for ReLU and ReLU6, respectively), similar to previous reports [51], most likely

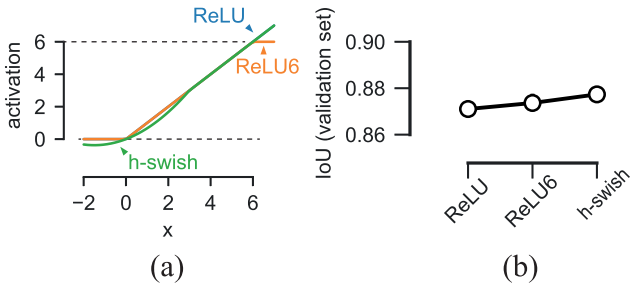

FIGURE 6. ReLU vs ReLU6 vs h-swish. a) comparison of activation functions with different ranges. ReLU is partially behind ReLU6 and swish. b) performance as mIoU of ReLU, ReLU6 and h-swish.

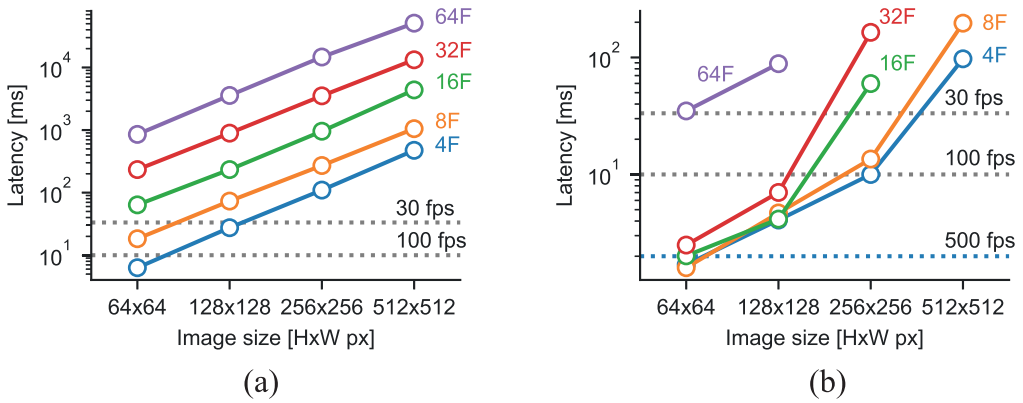

FIGURE 7. Latency for different U-Net configurations (4L, 4, 8, 16, 32 or 64F) and image sizes. a) performance as fully quantized model mapped to CPU. b) Performance of models that compiled successfully for EdgeTPU. Dashed lines indicate 30 frames per second (fps), i.e. video rate, 100 and 500 fps.

due to the small, but existing hysteresis close to the origin (Figure 6a).

F. PORTING TO EdgeTPU

GPUs are highly optimized to perform computations using floating point variables. However, mobile CPUs and TPUs can increase their performance if the computations are based on 8-bit integer values [52], [53]. To reduce the load of the main CPU and accelerate the neural network execution, we thought to utilize the inexpensive Coral platform [54] to perform fast and high quality segmentations. We further focused on the Coral USB Accelerator that contains a dedicated EdgeTPU and is connected via a USB 3.1 interface.

The EdgeTPU is able to perform all operations in our optimized U-Net architecture, including separable convolutions and resizing using nearest neighbour (default Upsample2D behavior). Unfortunately, the h-swish function is currently not supported, and we use the h-swish function only for better CPU inference if the USB Accelerator is not available, but use the ReLU6 activation function for the EdgeTPU variant (similar to deployed MobileNetV3 and EfficientNets on EdgeTPUs). We further tested if separable convolutions have another gain in speed compared to ordinary convolutions, as it has been shown that on the EdgeTPU ordinary convolutions perform faster [35].

To create a baseline, we first deployed a range of basic U-Net configurations (similar to Figure 3 and Table 1), fully quantized to uint8 and converted to TFLITE, and measured the performance in terms of latency across several image sizes (Figure 7a). The reason to use different image sizes is to run inference using a region of interest of the original image and to determine speed effects depending on image size. Only three configurations are able to perform in video rate, i.e. 30 frames per second (fps), only one configuration was able to run at 100 fps (4F/64 × 64, Figure 7a). Next, we recompiled all models using the EdgeTPU compiler to map operations to the EdgeTPU. Large U-Net configurations did not compile for the EdgeTPU, i.e. 64F/256 × 256, 64F/512 × 512, 32F/512 × 512 and 16F/512 × 512, because of model size limitations on the hardware accelerator (Figure 7b). We observed a dramatic speedup running inference on the EdgeTPU ranging from 3.7 to 127.2× compared to CPU-bound inference (Table 3 and Figure 7). Detailed latencies are given in Table 3. Nine of the twenty models tested are now able to process images at 100 fps (Figure 7b), using very small images (64 × 64 px) we are able to exceed the 500 fps margin for some models.

TABLE 3. Speed improvements on EdgeTPU compared to CPU when model compiled as TFLITE.

F	Image size	Latency [CPU]	Latency [EdgeTPU]	Speedup
4	64x64	6.31 ms	1.70 ms	3.72x
	128x128	27.51 ms	4.08 ms	6.75x
	256x256	109.81 ms	10.00 ms	10.98x
	512x512	474.34 ms	97.38 ms	4.87x
8	64x64	18.35 ms	1.60 ms	11.50x
	128x128	73.28 ms	4.69 ms	15.63x
	256x256	271.18 ms	13.55 ms	20.02x
	512x512	1051.63 ms	196.40 ms	5.35x
16	64x64	63.78 ms	2.02 ms	31.65x
	128x128	233.60 ms	4.15 ms	56.23x
	256x256	958.14 ms	59.83 ms	16.01x
32	64x64	233.69 ms	2.49 ms	93.78x
	128x128	892.64 ms	7.02 ms	127.18x
	256x256	3521.82 ms	164.00 ms	21.47x
64	64x64	852.74 ms	34.91 ms	24.42x
	128x128	3573.14 ms	88.48 ms	40.38x

The EdgeTPU compiler decides, which operations are mapped to the EdgeTPU or to the CPU. As we are operating with relatively large images (512 × 256), the compiler is mapping some Upsample2D operations to the CPU because of precision reasons. However, when operations are performed on both, CPU and EdgeTPU, the performance drops and results in lower throughput, contrary to our main goal in this study. Therefore, we changed the classic Upsample2D block to a custom upsampling routine that splits the image into tiles, upsamples the tiles and then merges the tiles back together (Figure 8). We provide an example implementation for Keras in the Supplementary Material (Supplementary Code). Our upscaling routine is computationally rather expensive, however, with this routine we can map 100% of the network operations to the EdgeTPU and thus, circumventing the even

TABLE 4. Performance across platforms. Architectures are abbreviated similar to previous Figures. EE/Ex (Encoder with and without second convolution), DD, Dx (Decoder with and without second convolution), C (Convolution), SC (Separable Convolution).

Model	Architecture	Params	Naive (Keras)	TFLite (mapped to CPU)	TFLite (mapped to EdgeTPU)	IoU (val)	IoU (test)
NN1	4L/16F, EE, DD, C, Concat	1.3M	180 ms	2365 ms	195 ms	0.858	0.769
NN2	4L/8F, EE, DD, C, Add	0.3M	104 ms	559 ms	31 ms	0.854	0.776
NN3	4L/8F, Ex, Dx, C, Add	0.09M	67 ms	249 ms	25 ms	0.832	0.746
NN4	4L/8F, Ex, Dx, SC, Add	0.03M	57 ms	189 ms	27 ms	0.807	0.741

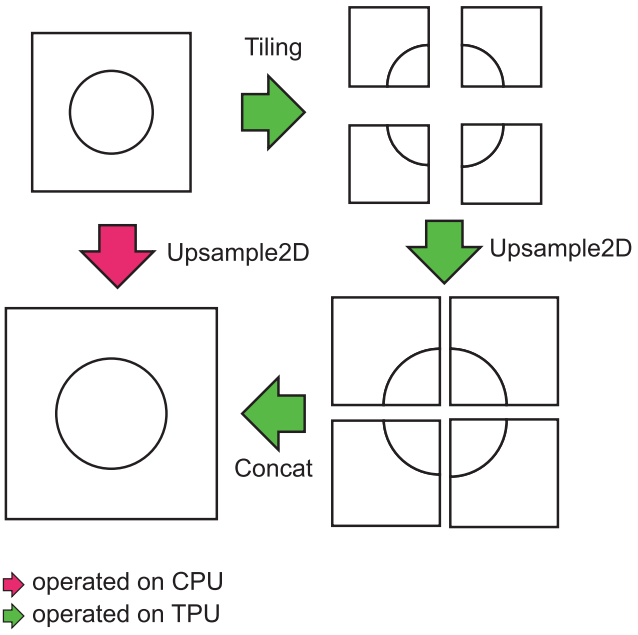

FIGURE 8. Upsampling operation on CPU vs. EdgeTPU for large images. See sample code in the Supplementary Material.

more expensive switch between CPU and EdgeTPU (see also later paragraphs). We compared the latency of four interesting configurations as gained from our in-depth analysis above in the default Keras inference mode (float32) on a CPU, converted to TFLITE mapped to the CPU, and converted to TFLITE and mapped to the EdgeTPU (Table 4). The low parametric models mapped to the EdgeTPU outperform significantly the same models executed on the CPU in the naive Keras environment (Table 4, Models 2-4). With latencies as low as 25 ms per frame, we yield a processing speed of 40 fps for a full 512 × 256 image, indicating that a typical 1,000 frame recording is analyzed in 25 s. Overall, we gain a speed-up 2.1× - 3.4× using solely the EdgeTPU for computation on Models 2-4. The larger model Model 1 (4L/16F) performs slightly worse on the EdgeTPU compared to the CPU (180 vs 195 ms, 0.9x). In line with previous studies [35], separable convolutions are faster on CPUs, whereas ordinary convolutions are slightly faster on the EdgeTPU (Table 4, compare Model 3 and 4). Taken together, all models tested are able to execute fully on the EdgeTPU and especially models with a low parameter space, as gained from our optimization procedure, outperform largely a consumer CPU.

G. TRANSFER ABILITY OF TECHNIQUES

To investigate if our approaches are applicable to other semantic segmentation neural network architectures, we tested different approaches to improve the SegNet [55] and Deeplabv3+ [56] architecture.

First, we tested the performance of the SegNet architecture when reducing the base filter count similar to Figure 3. Interestingly, we found that reducing the base filter count does even improve the performance in terms of IoU score (64 f_{base}: IoU = 0.704 vs 4 f_{base}: IoU = 0.772) as shown in Supplementary Figure 2a, however, performed consistently worse on the BAGLS dataset compared to our U-Net implementations (SegNet best IoU=0.798, U-Net best IoU=0.895). Inference speed was strongly improved (64 f_{base} 4293 ms to 4 f_{base} 165 ms on CPU, Supplementary Figure 2b), similar fast as our first proposed neural network in Table 4. However, due to technical limitations in the TFLITE and EdgeTPU compiler network in respect to the index unpooling, we were not able to port SegNet to the EdgeTPU.

Second, we tested the DeeplabV3+ architecture. We found that both common backbones, Xception [46] and MobileNetV2 [49], result in high IoU (validation) scores (0.861 and 0.849, respectively). Because both IoU scores are very close to each other, but the MobileNetV2 backbone has only 5% of the Xception parameter space, we opted to further optimize the MobileNetV2 backbone. The alpha-parameter in the MobileNetV2 architecture scales basically the base filter count (f_{base}). We investigated alpha values of 1 (full backbone), 0.75, 0.5 and 0.25, resulting in an f_{base} of 32, 24, 16 and 8, respectively. We found that reducing the alpha value does not largely impact the mean IoU scores (Supplementary Figure 3a,b, alpha=1 IoU=0.850 vs. alpha=.25 IoU=0.840 (validation set)). However, by reducing f_{base} using lower alpha values, we could save up to 85.22% of parameters (Supplementary Figure 3b). This leads to a large inference time speed-up in the naive Keras environment (CPU, Supplementary Figure 3c). When porting the architectures to the EdgeTPU (see Methods), the inference latency is already lower compared to the naive Keras environment (alpha=1, 166 ms vs. 478 ms for EdgeTPU and naive Keras, respectively). However, when reducing alpha, the EdgeTPU latency remained stable, whereas the naive Keras environment gained lower latencies (168 ms vs. 107 ms). We found that this phenomenon is due to the upscaling layers that remain partially on the CPU and are not fully mapped to the EdgeTPU. By using our introduced custom upscaling routine

(Figure 8), we were able to fully port the networks to the EdgeTPU and outperform largely the naive Keras environment gaining almost video rate inference times (alpha=.25, fully mapped to EdgeTPU, 36 ms, Supplementary Figure 3d). Thus, a 1,000 frame long video will be processed in roughly 34 s. Similar to the U-Net, the relatively expensive upscaling algorithm worsens the performance in the naive Keras environment. However, due to the complete mapping of all layers to the EdgeTPU, we gain significant speedups in the EdgeTPU environment (Supplementary Figure 3d,e). Still, with all optimizations we neither gain lower latencies nor better performance as available in our optimized U-Net architecture (compare Supplementary Figure 3b,e and Table 4).

In summary, our proposed methods are also applicable on other semantic segmentation architectures. Still, the U-Net is outperforms other state-of-the-art networks in terms of performance (IoU and latency), as well as portability to the EdgeTPU.

H. CLINICAL RELEVANCE

To show the segmentation performance of the architectures provided in Table 4, we tested images from the BAGLS test dataset that have not been presented during training. All networks reliably produce segmentation maps with high IoU scores (Figure 9).

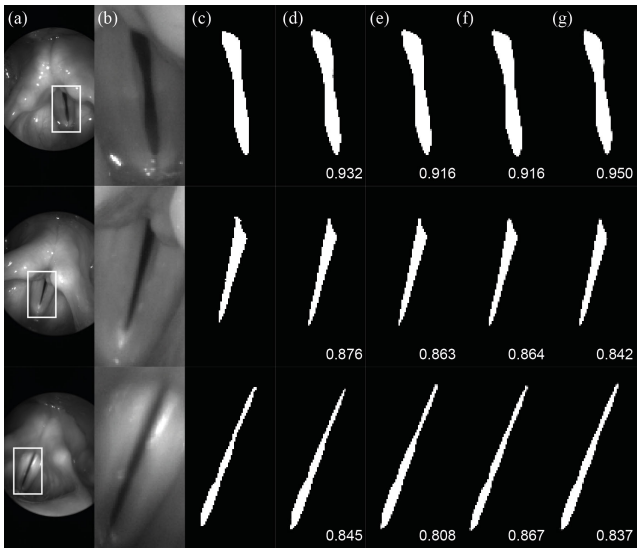

FIGURE 9. Segmentation performance across tested neural networks on images of BAGLS benchmark dataset. (a) original endoscopy image used as input for neural network, (b) inset of (a), white box, (c) ground-truth, (d) first, (e) second, (f) third, and (g) fourth neural network configuration as shown in Table 4.

To clinically validate our reduced, EdgeTPU-mapped architectures, we compared commonly used, clinical relevant, GAW extracted parameters, such as fundamental frequency (F_0), jitter and shimmer [41] (see also Methods). We analyzed 57 recordings from young, healthy individuals, a random subset of a recording collection segmented and analyzed in a previous study [39]. There, segmentations were generated by the current gold-standard, semi-automatic threshold-based region growing [6], and are used in the following as ground truth.

After fully automatically segmenting the same 1,000 frames as analyzed in [39], we computed the glottal area waveforms (GAW), i.e. the sum of segmented pixels across frames. All GAWs do not show large differences across architectures, only slight differences in amplitude (an example is shown in Figure 10a), potentially affecting clinical parameters dependent on the amplitude, such as shimmer (see eq. 3). In Figure 10b-d, we show, however, that these

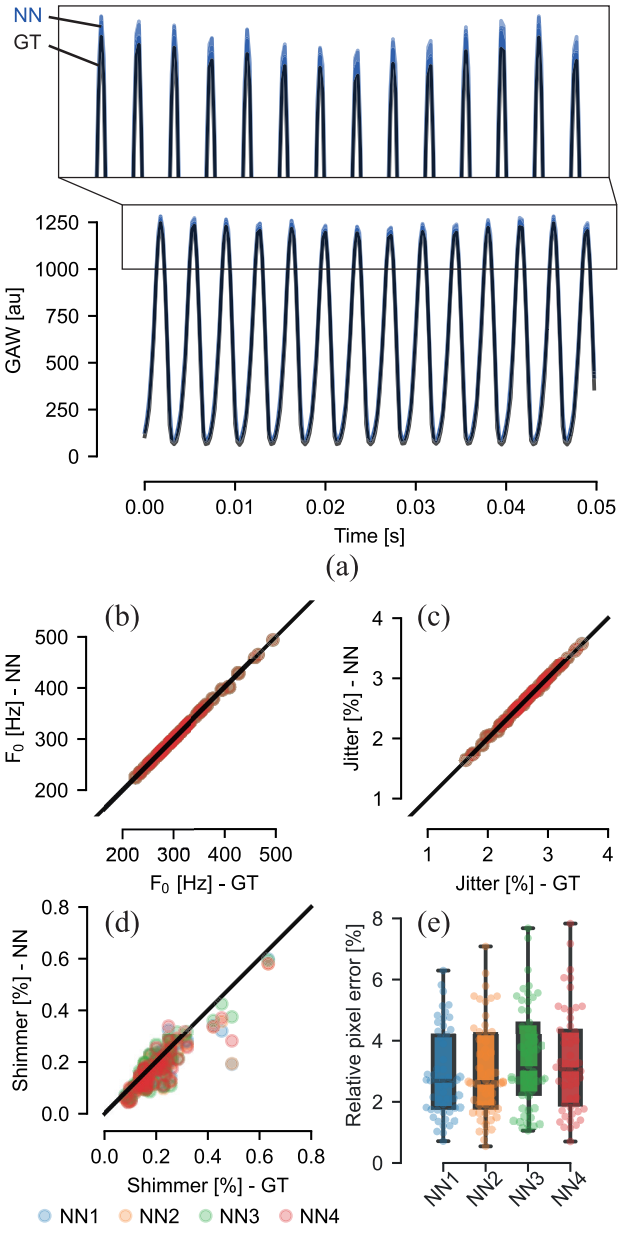

FIGURE 10. (a) Glottal Area Waveform comparison. Note that mainly peaks differ. (b) Fundamental frequency (F_0) ground-truth (GT) vs neural networks (NN1-4) provided in Table 4. (c) Jitter-%, (d) Shimmer-% (e) Differences in px relative to the ground-truth for each recordings.

clinical relevant parameters are barely affected and still retain their validity. Even though shimmer seems to be slightly affected (Figure 10d), the absolute error is less than 0.5% on a scale that ranges between 0 and 300% as shown in [41]. These differences in amplitude are due to small changes at border pixels in the segmentation that are on average 2.5% of the total segmented pixels across architectures as shown in Figure 10e. Taken together, these differences in amplitude are therefore negligible. We therefore conclude that our EdgeTPU-mapped, optimized U-Net architecture is suitable for glottal area segmentation at the point of care and provides clinically valid, objective parameters.

V. DISCUSSION AND CONCLUSION

In this study, we have shown that the U-Net architecture is a powerful architecture to perform glottis segmentation. We reduced the parameter space by changing the filter size, the network depth, the residual propagation, the encoder and decoder network and the activation function. In terms of glottis segmentation, we can show that a small, but well-chosen network is capable to produce reliable segmentation results, without the need of convolutional LSTMs as suggested elsewhere [8]. Our results lie in the range of inter-segmenter variability common to glottis segmentation [57], and thus, are performing as good as experts in the field. Our method is also superior to other (fully automatic) glottis segmentation algorithms available [8], [9], [12], as it is validated on a diverse dataset, computationally highly efficient and also highly portable. However, by constraining the network capacity, the network may generalize less and thus, produce potentially erroneous segmentations depending on the input data. We therefore propose a potential solution by training multiple segmentation networks with different features, such as capacity or training data, and introducing a classifier that selects the most appropriate network for the given input data. These network selection classifiers also ensures that networks can specialize, but not overfit to a given data. We also suggest that implementing other common architecture features, such as (inverted) residual layers [47], [49] or Inception modules [58] are able to improve segmentation performance by simultaneously not decreasing the inference speed dramatically.

In our medical use case, we used full endoscopic images (512×256 px) in the inference and gained a rather slow processing speed of 40 fps. By cropping the full endoscopic image to a region of interest (ROI) containing the glottis (similar to the inset in Figure 9), one can gain even lower latencies, i.e. speed improvements, as shown in Table 3 and Figure 7. As many medical image datasets rely on adjacent image features, cropping might not be feasible. By introducing our custom upscaling routine (Figure 8), we allow the inference of large images on EdgeTPUs. For a future study, we propose an architecture that evaluates and therefore combines both, relevant ROI detection and efficient segmentation.

We further envision that other optimization factors, such as pruning [59]–[62], neural architecture search [35], [63]–[66], neural unit optimization [67], [68], or the use of MixConvs [18] may lead to more efficient architectures in general (such as the EfficientNet-EdgeTPU [35]) or to more specialized architectures for a medical use case, e.g. glottal area segmentation. Notably, current pruning algorithms implemented in TensorFlow are only creating sparsity, allowing efficient model compression, however, do not yield any speed improvments. Interestingly, by using quantization aware training and converting the neural networks from float32 to uint8, we are still able to produce very accurate segmentation maps. Previously, quantization strategies that lower the bit depth showed that networks are still able to produce astonishing results, for example shown by [69], where the authors used binary convolutional neural networks, achieved a 69.2 % Top-5 accuracy on the ImageNet dataset and due to the efficient implementation of binary operations gained a (theoretical) $58\times$ speed-up compared to conventional convolutions. Here we show a real $79\times$ speedup on commercial available hardware and environments compared to our initial baseline.

The EdgeTPU is a specialized integrated circuit for deep learning purposes. In our study we show that EdgeTPUs are a cost-effective alternative to GPUs for optimized neural architectures, and even show that they outperform costly CPUs. EdgeTPUs can be used as external hardware accelerator (this study), permanent attached to the system using PCIe or M.2 interfaces, or directly attached on embedded solution as ASICs. Although EdgeTPUs are not covering yet all TensorFlow operations to port every neural architecture, such as dilation rate that is important for atrous convolutions in the Deeplabv3+ architecture [56], adapted, EdgeTPU-optimized architectures can reach similar performance levels [35].

To our knowledge, this is the first application of a medical semantic segmentation task mapped completely to an EdgeTPU and ready for clinical deployment at the point of care. While we focus here on glottis segmentation, the general concepts of this study, neural network optimization and EdgeTPU deployment, are transferable to other architectures and semantic segmantion tasks. We envision that biomedical imaging tasks, e.g. optical coherence tomography (OCT) or medical ultrasound techniques, such as echocardiography, will benefit from this real-time segmentation approach. Especially mobile and remote imaging platforms with highly limited resources would be able to use EdgeTPUs to implement end-to-end artificial intelligence applications. Our concept can also be extended to the automotive and industrial sector, where fast, remote and online semantic segmentation is important.

ACKNOWLEDGMENT

Andreas M. Kist would like to thank P. Gómez for discussions and initial implementations of U-Nets with lower filter count.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

REFERENCES

[1] T. Wittenberg, P. Mergell, M. Tigges, and U. Eysholdt, "Quantitative characterization of functional voice disorders using motion analysis of high-speed video and modeling," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 3, Apr. 1997, pp. 1663–1666.

[2] E. C. Inwald, M. Döllinger, M. Schuster, U. Eysholdt, and C. Bohr, "Multiparametric analysis of vocal fold vibrations in healthy and disordered voices in high-speed imaging," *J. Voice*, vol. 25, no. 5, pp. 576–590, Sep. 2011.

[3] T. Braunschweig, J. Flaschka, P. Schelhorn-Neise, and M. Döllinger, "High-speed video analysis of the phonation onset, with an application to the diagnosis of functional dysphonias," *Med. Eng. Phys.*, vol. 30, no. 1, pp. 59–66, Jan. 2008.

[4] A. Skalski, T. Zielinki, and D. Deliyski, "Analysis of vocal folds movement in high speed videoendoscopy based on level set segmentation and image registration," in *Proc. Int. Conf. Signals Electron. Syst.*, Sep. 2008, pp. 223–226.

[5] S.-Z. Karakozoglou, N. Henrich, C. d'Alessandro, and Y. Stylianou, "Automatic glottal segmentation using local-based active contours and application to glottovibrography," *Speech Commun.*, vol. 54, no. 5, pp. 641–654, Jun. 2012.

[6] J. Lohscheller, U. Eysholdt, H. Toy, and M. Dollinger, "Phonovibrography: Mapping high-speed movies of vocal fold vibrations into 2-D diagrams for visualizing and analyzing the underlying laryngeal dynamics," *IEEE Trans. Med. Imag.*, vol. 27, no. 3, pp. 300–309, Mar. 2008.

[7] P. Gómez, A. M. Kist, P. Schlegel, D. A. Berry, D. K. Chhetri, S. Dürr, M. Echternach, A. M. Johnson, S. Kniesburges, M. Kunduk, Y. Maryn, A. Schützenberger, M. Verguts, and M. Döllinger, "BAGLS, a multihospital benchmark for automatic glottis segmentation," *Sci. Data*, vol. 7, no. 1, pp. 1–12, Dec. 2020.

[8] M. K. Fehling, F. Grosch, M. E. Schuster, B. Schick, and J. Lohscheller, "Fully automatic segmentation of glottis and vocal folds in endoscopic laryngeal high-speed videos using a deep convolutional LSTM network," *PLoS ONE*, vol. 15, no. 2, Feb. 2020, Art. no. e0227791.

[9] M.-H. Laves, J. Bicker, L. A. Kahrs, and T. Ortmaier, "A dataset of laryngeal endoscopic images with comparative study on convolution neural network-based semantic segmentation," *Int. J. Comput. Assist. Radiol. Surg.*, vol. 14, no. 3, pp. 483–492, Mar. 2019.

[10] A. Hamad, M. Haney, T. E. Lever, and F. Bunyak, "Automated segmentation of the vocal folds in laryngeal endoscopy videos using deep convolutional regression networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2019, pp. 1–9.

[11] F. Schenk, M. Urschler, C. Aigner, I. Roesner, P. Aichinger, and H. Bischof, "Automatic glottis segmentation from laryngeal high-speed videos using 3D active contours," in *Proc. MIUA*, 2014, pp. 111–116.

[12] O. Gloger, B. Lehnert, A. Schrade, and H. Volzke, "Fully automated glottis segmentation in endoscopic videos using local color and shape features of glottal regions," *IEEE Trans. Biomed. Eng.*, vol. 62, no. 3, pp. 795–806, Mar. 2015.

[13] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*. [Online]. Available: http://arxiv.org/abs/1704.04861

[14] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size," 2016, *arXiv:1602.07360*. [Online]. Available: http://arxiv.org/abs/1602.07360

[15] L. Sifre, "Rigid-motion scattering for image classification," Ph.D. dissertation, Centre De Mathematiques Appliquees, École Polytechnique, Palaiseau, France, 2014.

[16] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 19–34.

[17] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," 2018, *arXiv:1802.03268*. [Online]. Available: http://arxiv.org/abs/1802.03268

[18] M. Tan and Q. V. Le, "MixConv: Mixed depthwise convolutional kernels," 2019, *arXiv:1907.09595*. [Online]. Available: https://arxiv.org/abs/1907.09595

[19] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2016, *arXiv:1611.01578*. [Online]. Available: http://arxiv.org/abs/1611.01578

[20] Z. Tang, X. Peng, K. Li, and D. N. Metaxas, "Towards efficient U-nets: A coupled and quantized approach," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 8, pp. 2038–2050, Aug. 2020.

[21] G. Bahl, L. Daniel, M. Moretti, and F. Lafarge, "Low-power neural networks for semantic segmentation of satellite images," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Seoul, South Korea, Oct. 2019, pp. 2469–2476. [Online]. Available: https://ieeexplore.ieee.org/document/9022273/

[22] A. Chaurasia and E. Culurciello, "LinkNet: Exploiting encoder representations for efficient semantic segmentation," 2017, *arXiv:1707.03718*. [Online]. Available: http://arxiv.org/abs/1707.03718

[23] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," 2016, *arXiv:1606.04797*. [Online]. Available: http://arxiv.org/abs/1606.04797

[24] F. Isensee, P. F. Jäger, S. A. A. Kohl, J. Petersen, and K. H. Maier-Hein, "Automated design of deep learning methods for biomedical image segmentation," 2019, *arXiv:1904.08128*. [Online]. Available: http://arxiv.org/abs/1904.08128

[25] Z. Zhou, M. Mahfuzur Rahman Siddiquee, N. Tajbakhsh, and J. Liang, "UNet++: Redesigning skip connections to exploit multiscale features in image segmentation," 2019, *arXiv:1912.05074*. [Online]. Available: http://arxiv.org/abs/1912.05074

[26] O. Oktay, J. Schlemper, L. Le Folgoc, M. Lee, M. Heinrich, K. Misawa, K. Mori, S. McDonagh, N. Y Hammerla, B. Kainz, B. Glocker, and D. Rueckert, "Attention U-net: Learning where to look for the pancreas," 2018, *arXiv:1804.03999*. [Online]. Available: http://arxiv.org/abs/1804.03999

[27] K. Ovtcharov, O. Ruwase, J.-Y. Kim, J. Fowers, K. Strauss, and E. S. Chung, "Accelerating deep convolutional neural networks using specialized hardware," *Microsoft Res. Whitepaper*, vol. 2, no. 11, pp. 1–4, 2015.

[28] S. Liu and W. Luk, "Towards an efficient accelerator for DNN-based remote sensing image segmentation on FPGAs," in *Proc. 29th Int. Conf. Field Program. Log. Appl. (FPL)*, Sep. 2019, pp. 187–193.

[29] S. Liu, H. Fan, X. Niu, H.-C. Ng, Y. Chu, and W. Luk, "Optimizing CNN-based segmentation with deeply customized convolutional and deconvolutional architectures on FPGA," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 11, no. 3, pp. 1–22, Dec. 2018.

[30] M. Krips, T. Lammert, and A. Kummert, "FPGA implementation of a neural network for a real-time hand tracking system," in *Proc. 1st IEEE Int. Workshop Electron. Design, Test Appl.*, Jan. 2002, pp. 313–317.

[31] S. Li, C. Wu, H. Li, B. Li, Y. Wang, and Q. Qiu, "FPGA acceleration of recurrent neural network based language model," in *Proc. IEEE 23rd Annu. Int. Symp. Field-Programmable Custom Comput. Mach.*, May 2015, pp. 111–118.

[32] B. Liu, D. Zou, L. Feng, S. Feng, P. Fu, and J. Li, "An FPGA-based CNN accelerator integrating depthwise separable convolution," *Electronics*, vol. 8, no. 3, p. 281, Mar. 2019. [Online]. Available: https://www.mdpi.com/2079-9292/8/3/281

[33] S. Liang, S. Yin, L. Liu, W. Luk, and S. Wei, "FP-BNN: Binarized neural network on FPGA," *Neurocomputing*, vol. 275, pp. 1072–1086, Jan. 2018.

[34] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," 2019, *arXiv:1905.11946*. [Online]. Available: http://arxiv.org/abs/1905.11946

[35] S. Gupta and M. Tan. *Efficientnet-Edgetpu: Creating Accelerator-Optimized Neural Networks With Automl*. Accessed: Mar. 29, 2020. [Online]. Available: https://ai.googleblog.com/2019/08/efficientnet-edgetpu-creating.html

[36] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," 2016, *arXiv:1603.04467*. [Online]. Available: http://arxiv.org/abs/1603.04467

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

[38] L. N. Smith, "Cyclical learning rates for training neural networks," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2017, pp. 464–472.

[39] P. Schlegel, M. Kunduk, M. Stingl, M. Semmler, M. Döllinger, C. Bohr, and A. Schützenberger, "Influence of spatial camera resolution in high-speed videoendoscopy on laryngeal parameters," *PLoS ONE*, vol. 14, no. 4, Apr. 2019, Art. no. e0215168.

[40] P. Jaccard, "Étude comparative de la distribution florale dans une portion des Alpes et des Jura," *Bull. Soc. Vaudoise Sci. Nat.*, vol. 37, pp. 547–579, 1901.

[41] P. Schlegel, M. Stingl, M. Kunduk, S. Kniesburges, C. Bohr, and M. Döllinger, "Dependencies and ill-designed parameters within high-speed videoendoscopy and acoustic signal analysis," *J. Voice*, vol. 33, no. 5, pp. 811.e1–811.e12, Sep. 2019.
[42] P. Virtanen *et al.*, "SciPy 1.0: Fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, pp. 261–272, Feb. 2020.
[43] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015, *arXiv:1505.04597*. [Online]. Available: http://arxiv.org/abs/1505.04597
[44] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3D U-net: Learning dense volumetric segmentation from sparse annotation," 2016, *arXiv:1606.06650*. [Online]. Available: http://arxiv.org/abs/1606.06650
[45] M. D Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," 2013, *arXiv:1311.2901*. [Online]. Available: http://arxiv.org/abs/1311.2901
[46] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 1251–1258.
[47] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for MobileNetV3," 2019, *arXiv:1905.02244*. [Online]. Available: http://arxiv.org/abs/1905.02244
[48] L. Kaiser, A. N. Gomez, and F. Chollet, "Depthwise separable convolutions for neural machine translation," 2017, *arXiv:1706.03059*. [Online]. Available: http://arxiv.org/abs/1706.03059
[49] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobilenetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.
[50] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," 2017, *arXiv:1710.05941*. [Online]. Available: http://arxiv.org/abs/1710.05941
[51] R. Avenash and P. Viswanath, "Semantic segmentation of satellite images using a modified CNN with hard-swish activation function," in *Proc. 14th Int. Joint Conf. Comput. Vis., Imag. Comput. Graph. Theory Appl.*, 2019, pp. 169–173.
[52] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient Integer-Arithmetic-Only inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2704–2713.
[53] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proc. 44th Annu. Int. Symp. Comput. Archit.*, 2017, pp. 1–12.
[54] S. Cass, "Taking AI to the edge: Google's TPU now comes in a maker-friendly package," *IEEE Spectr.*, vol. 56, no. 5, pp. 16–17, May 2019.
[55] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.
[56] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017, *arXiv:1706.05587*. [Online]. Available: http://arxiv.org/abs/1706.05587
[57] Y. Maryn, M. Verguts, H. Demarsin, J. van Dinther, P. Gomez, P. Schlegel, and M. Döllinger, "Intersegmenter variability in high-speed laryngoscopy-based glottal area waveform measures," *Laryngoscope*, Dec. 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/lary.28475, doi: 10.1002/lary.28475.
[58] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.
[59] J.-H. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5058–5066.
[60] K. Suzuki, I. Horiba, and N. Sugie, "A simple neural network pruning algorithm with application to filter synthesis," *Neural Process. Lett.*, vol. 13, no. 1, pp. 43–53, 2001.
[61] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," 2016, *arXiv:1611.06440*. [Online]. Available: http://arxiv.org/abs/1611.06440
[62] J. Lin, Y. Rao, J. Lu, and J. Zhou, "Runtime neural pruning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 2181–2191.
[63] H. Jin, Q. Song, and X. Hu, "Auto-Keras: An efficient neural architecture search system," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 1946–1956.
[64] C. Wong, N. Houlsby, Y. Lu, and A. Gesmundo, "Transfer learning with neural automl," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 8356–8365.
[65] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei, "Auto-DeepLab: Hierarchical neural architecture search for semantic image segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 82–92.
[66] X. He, K. Zhao, and X. Chu, "AutoML: A survey of the State-of-the-Art," 2019, *arXiv:1908.00709*. [Online]. Available: http://arxiv.org/abs/1908.00709
[67] K. G. Sheela and S. N. Deepa, "Review on methods to fix number of hidden neurons in neural networks," *Math. Problems Eng.*, vol. 2013, Jun. 2013, Art. no. 425740.
[68] K. Z. Mao and G.-B. Huang, "Neuron selection for RBF neural network classifier based on data structure preserving criterion," *IEEE Trans. Neural Netw.*, vol. 16, no. 6, pp. 1531–1540, Nov. 2005.
[69] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: Imagenet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 525–542.

ANDREAS M. KIST received the B.S. and M.S. degrees in molecular medicine from Friedrich-Alexander-University Erlangen-Nuremberg, Germany, in 2012 and 2014, respectively, and the Ph.D. degree from the Max-Planck-Institute of Neurobiology, Germany, in 2019. Since 2018, he has been a Research Scientist with the University Hospital Erlangen. His research interests include image acquisition and processing using computer vision and advanced machine learning methods. He received several highly competitive fellowships, such as the Joachim-Herz-foundation Addon Fellowship and funding from the Boehringer-Ingelheim fonds.

MICHAEL DÖLLINGER studied in mathematics at University Erlangen-Nürnberg, Germany. He received the Diploma (M.Sc.) degree in February 2000, and the Ph.D. degree in computer science from University Erlangen-Nürnberg, in November 2002. From 2003 to 2005, he was a Postdoctoral Fellow with the University of California, Los Angeles. Then, he returned to Erlangen, Germany, and became an Assistant Professor and an Associate Professor, from 2005 to 2008. In 2008, he became a Professor and the Head of Research. He was a Scientific Head of the DFG funded Research Group FOR894 Fundamentals on Flow Dynamics in Voice Production, from 2008 to 2013. Since 2008, he has been an Adjunct Professor with Louisiana State University, Baton Rouge.