ACCURATE AND RESOURCE-EFFICIENT LIPREADING WITH EFFICIENTNETV2 AND TRANSFORMERS

Alexandros Koumparoulis, Gerasimos Potamianos

Electrical and Computer Engineering Department, University of Thessaly, Volos, Greece

ABSTRACT

We present a novel resource-efficient end-to-end architecture for lipreading that achieves state-of-the-art results on a popular and challenging benchmark. In particular, we make the following contributions: First, inspired by the recent success of the EfficientNet architecture in image classification and our earlier work on resource-efficient lipreading models (MobiLipNet), we introduce EfficientNets to the lipreading task. Second, we show that the currently most popular in the literature 3D front-end contains a max-pool layer that prohibits networks from reaching superior performance and propose its removal. Finally, we improve our system's back-end robustness by including a Transformer encoder. We evaluate our proposed system on the "Lipreading In-The-Wild" (LRW) corpus, a database containing short video segments from BBC TV broadcasts. The proposed network (T-variant) attains 88.53% word accuracy, a 0.17% absolute improvement over the current state-of-the-art, while being five times less computationally intensive. Further, an up-scaled version of our model (L-variant) achieves 89.52%, a new state-of-the-art result on the LRW corpus.

Index Terms— EfficientNet, Transformers, Lipreading.

1. INTRODUCTION

Visual speech recognition (VSR) models have progressed significantly, thanks to advances in deep learning algorithms. Typically, these are based on accurate but computationally intensive architectures, such as convolutional neural networks (CNNs) and recurrent or self-attention layers. For example, the 2D CNN of [1] requires 11.22×10^9 floating-point operations (FLOPs) to process a single video frame. In resource-constrained scenarios, such deep learning-based models are impractical due to their computational profile.

Recently, a small number of works [2–5] have focused on improving VSR model efficiency, in order to enable broader application of this technology. Proposed architectures reduce computational requirements by replacing standard convolutions with grouped ones, such as depthwise and pointwise. In these cases, the gains in FLOPs are significant: for example, our MobiLipNetV2 [3] is 37 times more efficient than a 3D-ResNet. However, when presented with challenging real-world data, efficient VSR models still lag in recognition performance. For example, the best ShuffleNetV2 model in [5] achieves a word accuracy (WAcc) of 85.5% on the LRW corpus [6], trailing the current state-of-the-art of 88.36% [7] on the task.

Motivated by the above, in this paper we attempt to eliminate the performance discrepancy between conventional and resource-efficient VSR systems. To this end, we propose a novel neural network architecture, focusing on both recognition accuracy and resource efficiency. In particular, we make the following contributions:

- First, we introduce a new VSR model based on the architecture of EfficientNet [8, 9]. Such resource-efficient model relies on pointwise/depthwise convolutions as well. However, its main departure from similar architectures like MobileNetV3 [10] is compound scaling, where the model's depth/width/resolution are scaled together, resulting in a family of models with different accuracy/efficiency trade-off. Here we present three configurations: a tiny one (denoted by "T"), targeting resource-constrained applications, and two larger ones (denoted by "M" and "L"). To differentiate between them, we append the variant-type letter at the end of the model name, e.g. EfficientNetV2-T. Our experiments show that these models yield superior performance over all other VSR architectures. To our knowledge, this represents the first use of EfficientNets in VSR.
- Second, we systematically study the 3D front-end of the VSR model. This is a crucial module, as it captures the short-term dynamics of the mouth region, and it has been proven advantageous over simpler 2D front-ends [3, 11, 12]. Specifically, we assume a 3D front-end with a single convolution layer, and we focus on the dimensions of the 3D convolution kernel and whether a max-pooling layer should be used downstream. Our experiments show that smaller kernels are as effective as larger ones, however max-pooling combined with non-unit convolution stride hurts recognition, thus adversely affecting the majority of recent VSR systems that employ it within their 3D front-ends.
- Third, we propose a robust back-end that combines a Transformer encoder with a temporal convolutional network (TCN). TCNs are known to perform on-par with recurrent architectures [5, 7], while being easier to train. However, one disadvantage is their fixed receptive field. For this reason, we pre-process the input features using a Transformer encoder, allowing us to better handle sequences of different size. This yields a nearly 1% absolute WAcc improvement on the LRW corpus.

We evaluate our proposed system for speaker-independent word VSR on the LRW corpus [6], a very popular lipreading dataset [2, 5–7, 12–25], thus allowing extensive comparisons. We report that our introduced EfficientNetV2-T model is more accurate than the currently best CNN-based network [7], while having a significantly smaller computational cost. Further, our larger EfficientNetV2-L configuration achieves the new state-of-the-art of 89.52% WAcc on LRW, namely a 1.16% absolute improvement over [7].

2. THE PROPOSED VSR SYSTEM AND ITS MODULES

This section introduces our proposed VSR system and details its modules. A schematic system overview is provided in Fig. 1a.

2.1. 3D Front-end

The mouth-region video frames are first processed by a 3D front-end in order to capture short-term spatio-temporal motion. The 3D

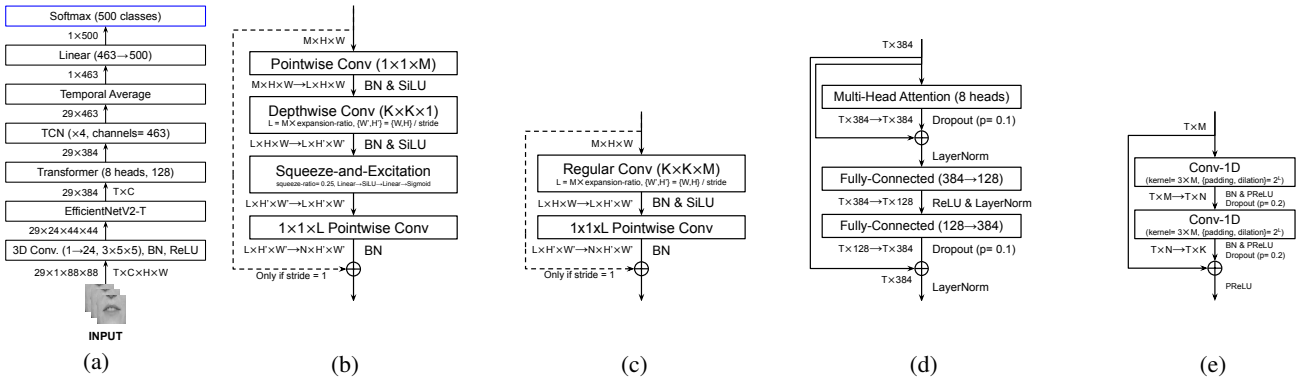

Fig. 1: Overview of the proposed VSR architecture, employing the EfficientNetV2-T model: (a) Entire system; (b) Inverted-Bottleneck module with SiLU activation; (c) Fused-MBConv module, where the first pointwise and depthwise convolutions have been merged into a single regular convolution layer; (d) Transformer-encoder layer [26], along with the hyper-parameters employed; (e) TCN-module. In our architecture we have used $M = N = K$ and dilation 2^L, where L is the index of the TCN module (0-3).

front-end complements the back-end temporal classifier [11], which primarily captures long-term dynamics.

The dominant 3D front-end in the literature [12, 21, 22, 27] consists of a convolutional layer with 3-dimensional (3D) kernels of $5\times7\times7$ size (time/width/height) and stride $1\times2\times2$, followed by batch normalization (BN) and rectified linear units (ReLU). The extracted feature maps are passed through a spatio-temporal max-pool layer with kernel of size $1\times3\times3$ and stride $1\times2\times2$.

Due to the non-unit stride in convolution and max-pool layers, the output spatial dimensions are four times smaller than the input. We postulate this design decision was mostly driven by practical constrains: by reducing significantly the output dimensions, the network could fit in the GPU memory and train faster. However, excessive and early downsampling discards crucial information in the feature extraction process that is not retained by the large 3D convolution kernel. For this reason, we remove the max-pool layer and decrease the convolution's kernel size to $3\times5\times5$.

2.2. EfficientNetV2

Most resource-efficient CNNs [10, 28–30] rely on introducing new basic-block modules that are computationally leaner. EfficientNets deviate from this pattern. First, neural architecture search (NAS) is utilized to obtain a baseline model that has good trade-off on accuracy and FLOPs. The baseline model is then scaled up with a compound scaling strategy (input resolution/width/depth), obtaining a family of models with different efficiency/accuracy ratios. In this work, we retain the input resolution fixed (88×88 pixels) and scale the networks in the other two dimensions. We proceed with describing the two basic modules used in EfficientNetV2-based models.

Inverted Residual Bottleneck (MBConv): The inverted residual with linear bottleneck module was first introduced in [29] (shown in Fig. 1b). Here, we describe the variant used in the EfficientNet architectures. First, a low-dimensional compressed representation 2D input is expanded (increasing the number of channels, $M \rightarrow L$) with pointwise (PW) convolution according to ratio ρ ($L = M \cdot \rho$), then filtered with a depthwise (DW) $K \times K$ convolution kernel ($M \rightarrow M$). Channel-wise attention is applied using the squeeze-and-excitation (SE) mechanism [31] (reduction ratio $r = 1/24$) and finally compressed back with pointwise convolution ($L \rightarrow N$). The depthwise convolution may have stride greater than one, decreasing the output dimensions. In case of unit stride, a residual connection is also applied (shown with a dashed line). EfficientNetV2 uses only $K = 3$. The SiLU activation [32] is also applied on the first two convolution layers and inside the SE mechanism.

To provide numerical examples of required FLOPs, we assume input of size $M = 16$, $L = 128$ ($\rho = 8$), $N = 32$, $H = W = 32$, DW kernel size $K = 3$ and unit stride. The first PW convolution layer requires $M\,L\,H\,W$ multiplications (2.09M FLOPs, 2048 parameters), the DW one costs $K^2\,L\,H\,W$ (1.17M FLOPs, 1152 parameters), and the second PW convolution costs $L\,N\,H\,W$ (4.19M FLOPs, 4096 parameters). The SE mechanism contains two anti-symmetrical fully-connected layers, each requiring $L\,(L{\cdot}r)$ multiplications (1.3K FLOPs, 1376 parameters). The total cost of a single such module is therefore 7.45M FLOPs and 8K parameters.

Fused-MBConv: The MBConv module relies on depthwise convolution for spatial filtering. Depthwise convolutions have fewer parameters and FLOPs than regular convolutions, but they often cannot fully utilize modern accelerators, especially in the early layers where spatial dimensions are large. To better utilize mobile or server accelerators, Fused-MBConv was recently proposed [9]. It fuses the expansion pointwise and depthwise convolution MBConv with a single regular convolution and lacks an SE mechanism, as shown in Fig. 1c. When applied in early stages, Fused-MBConv improves training speed with a small overhead on parameters and FLOPs, however it is not as effective at later stages [9].

For the same numerical setup as in MBConv, the regular convolution costs $M\,K^2\,L\,H\,W$ multiplications (18M FLOPs, 18K parameters), and the projection pointwise costs $L\,N\,H\,W$ (4.19M FLOPs, 4096 parameters). In total, the module requires 22.19M FLOPs and 23K parameters. While the cost is roughly three times that of MBConv, for smaller numbers of channels and larger spatial dimensions MBConv is more efficient on parallel hardware, where

Table 1: Hyper-parameters of the Baseline and T/M/L EfficientNet variants. Operators MBConv and Fused-MBConv are explained in Section 2.2. All modules use spatial kernels with size $1\times3\times3$. For all SE layers, $r = 1/24$. For variants T ($\alpha = 0.66$, $\beta = 1.0$) and M ($\alpha = 1.0$, $\beta = 1.1$) the *#Layers* and *#Channels* columns, respectively, are not shown, as they equal those of the Baseline.

Stage	Module	Stride	**Baseline**		**T** ($\alpha=0.66$)	**M** ($\beta=1.1$)	**L** ($\alpha=1.1$, $\beta=1.2$)	
			#Channels	#Layers	#Channels	#Layers	#Channels	#Layers
1	Fused-MBConv (ρ=1)	1	16	1	8	2	24	2
2	Fused-MBConv (ρ=4)	2	32	2	16	3	48	4
3	Fused-MBConv (ρ=4)	2	48	2	32	3	64	4
4	MBConv (ρ=4, SE)	2	96	3	56	4	128	6
5	MBConv (ρ=6, SE)	1	112	5	64	6	120	6
6	MBConv (ρ=6, SE)	2	192	8	112	9	208	10
7	Conv1×1→Pooling→GLU	1	768→384	1	768→384	1	768→384	1

Table 2: Comparison of ShuffleNetV2 ($0.5\times$) [5] and our EfficientNetV2-T VSR systems with four 3D front-end variants, in terms of recognition performance (in WAcc, %, on the LRW test set) and efficiency (in per-frame FLOPs and parameters).

3D Front-end			ShuffleNetV2 (0.5×) [5]				EfficientNetV2-T			
Conv-kernel size	FLOPs ($\times 10^9$)	Max Pool	WAcc (%)	FLOPs ($\times 10^9$) Total	CNN	Params ($\times 10^6$)	WAcc (%)	FLOPs ($\times 10^9$) Total	CNN	Params ($\times 10^6$)
3×3×3	0.04	✗	80.82	0.66	0.46	5.012	88.38	1.56	1.12	8.96
		✓	79.50	0.32	0.12		86.19	0.80	0.36	
3×5×5	0.10	✗	81.04	0.72	0.46	5.013	88.52	1.62	1.12	8.96
		✓	79.51	0.39	0.12		86.67	0.86	0.36	

MBConv is bottlenecked by memory bandwidth instead of compute.

Network Topology and Scaling: For our EfficientNet-based VSR systems we start with the Baseline model of EfficientNetV2, shown in the fourth column of Table 1. In contrast to the original Baseline, we remove the first layer (Stage = 0), also known as stem layer, since its functionality has been replaced by the 3D front-end (Section 2.1). Further, all network variants share the same last Stage (7): a PW convolution expands the number of channels ($X \rightarrow 768$, where the size of X depends on the previous stage, 6), a spatial averaging operation aggregates spatial dimensions, and finally a GLU [33] reduces the final number of channels to 384. Apart from these changes, our Baseline model is the same as the original [9].

All network variants apply the same modules as the Baseline. However, each variant is a scaled version of the baseline model. Each module is parameterized by the channel width (α) that controls how many channels will be used, as well as by the depth multiplier (β) that controls how many times the number of layers will be scaled in each stage. We retain a fixed input size (88×88 pixels) for all variants. We consider three scaling setups. The T (from tiny) variant ($\alpha=0.66$, $\beta=1.0$) is a configuration with a reduced number of channels and our main network, aiming at a computational cost of roughly one Giga-FLOP per frame. In addition, we create two more variants, M (medium, $\alpha=1.0$, $\beta=1.1$) and L (large, $\alpha=1.1$, $\beta=1.2$), to study the performance effect of scaling each axis (module channels and module depth).

2.3. Transformer

After feeding the video through the 3D front-end and the 2D EfficientNet, the output features are further processed by a Transformer encoder [26] (shown in Fig. 1d). The Transformer encoder first passes the input sequence through a multihead-attention layer [26] and subsequently through two fully-connected layers. The attention mechanism in the Transformer allows the network to dynamically discard irrelevant information, for example the end of the previous word or the start of the next one. Further, because it is fully parallelizable it is fast to compute, a desirable but missing property in recurrent architectures. The Transformer retains the number of input channels (384).

2.4. TCN and Final Classifier

Finally, the output of the Transformer is fed to a 4-layer TCN, similar to the ones used in [5]. Recently, TCNs have been used with great success in VSR systems [5, 7, 21]. They are used as a drop-in replacement to recurrent architectures such as the GRU or LSTM. They are easier to train and for tasks like ours (LRW) have proved as accurate as recurrent models. Each TCN module (shown in Fig. 1e) applies two 1D convolutions with a 3×1 kernel and dilation $\delta=2^L$, where L is the module index (0-3). Further, each convolution layer pads the input appropriately in order to maintain the input length.

Table 3: Comparison of back-end classifier variations. The first two row-groups investigate a leaner TCN (named TCN-S) and whether a Transformer encoder is beneficial. The last row-group investigates fewer heads in the multi-head attention mechanism.

Model	WAcc (%)	Params ($\times 10^6$)	FLOPs ($\times 10^9$)	Inference-time (ms / frame)
EfficientNetV2-T + TCN-S	87.27	6.73	1.49	2.90
EfficientNetV2-T + TCN-S + Transformer (8 heads)	88.14	7.42	1.51	2.93
EfficientNetV2-T + TCN	87.52	8.89	1.60	2.98
EfficientNetV2-T + TCN + Transformer (8 heads)	88.52	8.96	1.62	3.01
EfficientNetV2-T + TCN + Transformer (1 head)	88.33	8.96	1.62	3.01
EfficientNetV2-T + TCN + Transformer (4 heads)	88.47	8.96	1.62	3.01

Each convolution is followed by BN and PReLU. Finally, the hidden output of the last TCN module ($L=3$) is temporally averaged, fed to the final classification head (fully-connected layer with $463\rightarrow500$), and a softmax normalizes it into a probability distribution.

3. EXPERIMENTAL SETUP

3.1. Database

We train and evaluate our proposed architecture on the challenging LRW database [6]. It consists of short audiovisual speech segments, extracted automatically from BBC TV broadcasts. The task is to recognize 500 distinct words from continuous speech. The database is challenging due to its high variability with respect to the number of speakers, naturally varying head-pose (near-frontal), and noisy background (real-world conditions). Words are not pre-segmented, and there may be co-articulation from preceding and subsequent ones. Moreover, there exist word pairs that share most visemes, for example nouns in singular and plural forms (e.g. thing / things), verbs in both present and past tenses (e.g. happen / happened / happening), and homophones (e.g. whether / weather). All clips have a fixed duration of 1.16 sec (29 frames at a 25 Hz rate). The training set consists of up to 1000 utterances per target-word, while the validation and testing sets both contain 50 utterances per word. Samples are shown in Fig. 2.

3.2. Visual Front-end

We use the same visual front-end as the one in [5], so we only briefly overview its operation here. LRW videos contain tightly cropped face images, thus the face detection step is skipped. First, 68 facial landmarks are detected and tracked using [34]. These are interpolated in case of a detection failure in a frame. Consequently, using the detected landmarks, the faces are aligned to a neutral reference frame, and a region-of-interest (ROI) of size 88×88 pixels containing the mouth area is extracted from each frame.

3.3. Training Setup

We follow the training hyper-parameters of [9]. We use two GPUs, each with a batch size of 80 videos for EfficientNetV2-T and 40

Fig. 2: Top row: example frames from the LRW corpus. Bottom row: corresponding ROIs after applying the visual front-end.

videos for variants EfficientNetV2-M and EfficientNetV2-L. We employ the RMSProp optimizer with decay 0.9 and momentum 0.9; BN momentum 0.99; weight decay 1e-5. Learning is first warmed up from 1e-6 to 0.18 for three epochs, and then decayed by 0.97 every 2.4 epochs. We use exponential moving average with 0.9999 decay rate, dropout (p=0.3), and stochastic depth [35] with 0.8 survival probability. During training, we apply data augmentations at the segment level by employing RandAugment [36].

4. RESULTS

We investigate a number of variations of our EfficientNetV2-T VSR architecture. First, we consider the role of the max-pool layer in the 3D front-end. Second, we investigate how the width of the TCN module affects model accuracy, whether the presence of the Transformer encoder is beneficial, and which number of attention heads leads to the best performance. Finally, we experiment with larger EfficientNets (M and L variants).

3D Front-end: We perform controlled experiments where the hyper-parameters of the 3D front-end are varied. In particular, we test two different convolution kernel sizes ($3\times3\times3$ and $3\times5\times5$), and, for each, two variants, with or without the max-pool layer. To ensure that our results transfer to other architectures too, we include results for the ShuffleNetV2 ($0.5\times$) [5] network as well. As we can see from the results in Table 2, adding a max-pool layer to the 3D front-end has detrimental effect on network performance, since all networks without it outperform those with it. One negative side-effect though is the increased computational cost. Further, from our experience, networks without the max-pool layer begin to converge faster than those with it.

Transformer and TCN: Transformers have been tried before on the LRW database without much success [22]. However, when combined with other components (such as TCN), they are effective. We perform three experiments to understand the effect of TCN width, the effect of Transformer presence, and finally whether a smaller number of attention heads leads to better performance. We summarize our results in Table 3. The first two rows contain results for the TCN-S variant, which has 386 channels, instead of 463 used in TCN. As we can see, the wider TCN shows a 0.25% absolute word accuracy improvement. Further, in both cases (TCN-S and TCN), the presence of the Transformer is beneficial, leading to significant

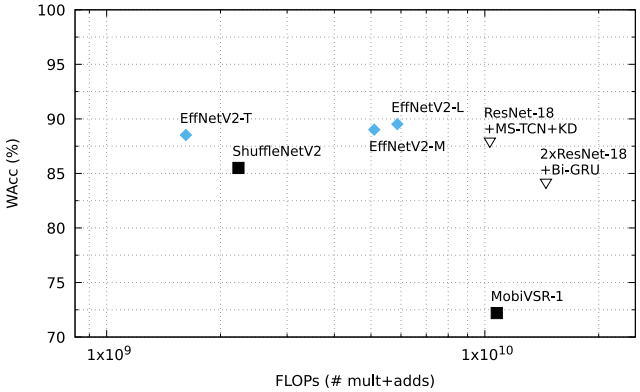

Fig. 3: Recognition performance on the LRW test set (in WAcc, %) vs. efficiency (in per-frame FLOPs) of the proposed EfficientNet-based VSR system (three variants), other resource-efficient models (ShuffleNetV2 and MobiVSR) and ResNet-based VSR systems (see also Table 4).

Table 4: EfficientNetV2 results for all three variants (T, M, L) on the LRW test set (lower part of the table). Inference time is measured on a single-core of a i7-8750H CPU using PyTorch. Other literature results on the same benchmark are summarized at the upper part.

Model	WAcc (%)	Params ($\times10^6$)	FLOPs ($\times10^9$)	Inference-time (ms / frame)
VGG [6]	61.10	–	–	–
ResNet-34 + BiLSTM [12]	83.00	–	–	–
ResNet-18 + 2× BiLSTM + word-boundary [13]	88.08	–	–	–
Multi-Grained ResNet-34 + Conv + BiConvLSTM [14]	83.34	–	–	–
Two-Stream ResNet-18 + BiLSTM [15]	84.07	–	–	–
ResNet-18 + Bi-GRU + Policy Gradient [16]	83.60	–	–	–
ResNet-18 + STFM [17]	83.70	–	–	–
2× ResNet-18 + Bi-GRU [18]	84.13	7.95	14.5	–
ResNet-18 + 3× Bi-GRU + MI [19]	84.41	–	–	–
ResNet-18 + BiGRU + Face-cutout [20]	85.02	–	–	–
ResNet-18 + Multi-Scale TCN [21]	85.30	–	–	–
MobiVSR-1 [2]	72.20	4.5	10.75	–
ResNet-18 + MS-TCN + MS-KD [5]	87.90	36.4	10.31	–
ResNet-18 + MS-TCN + MS-KD (ensemble) [5]	88.50	36.4	10.31	–
ShuffleNetV2 + MS-TCN + KD [5]	85.50	28.8	2.23	–
SE-ResNet-18 + BiGRU [22]	85.00	–	–	–
SE-ResNet-18 + BiGRU + word-boundary [22]	88.40	–	–	–
ResNet-18 + HPConv [23]	86.83	–	–	–
ResNet-18 + HPConv + word-boundary [23]	88.60	–	–	–
2× ResNet-50 + SlowFast [37]	84.40	–	–	–
ResNet-34 + GCN + BiGRU [38]	84.25	–	–	–
ResNet-18 + Transformer [24]	87.32	–	–	–
ResNet-18 + Dense-TCN [7]	88.36	–	–	–
ResNet-18 + Conformer + Bimodal-KD [39]	88.10	–	–	–
ALSOS-ResNet-18 + MS-TCN [40]	87.01	–	–	–
3D-ResNet18 + BiGRU [41]	86.23	–	–	–
Spatio-Temporal Attention + KD + word-boundary [25]	88.64	–	–	–
EfficientNetV2-T + TCN + Transformer (8 heads)	88.52	8.96	1.62	3.01
EfficientNetV2-M + TCN + Transformer (8 heads)	89.01	13.68	5.10	7.07
EfficientNetV2-L + TCN + Transformer (8 heads)	**89.52**	15.47	5.87	7.89

WAcc improvement (by 0.87% and 1.0% absolute, respectively). In contrast, a smaller number of heads (1 and 4) degrades WAcc very slightly, compared to the 8-head variant.

EfficientNet: We summarize literature results in the upper section of Table 4, while in the lower section we present results for our three EfficientNetV2-based variants (T/M/L). Along with WAcc (%), we also include model size (number of parameters) and computational requirements (FLOPs and inference time per frame), where available. We consider as the state-of-the-art on LRW the 88.36% WAcc reported in [7]. Note that some systems in the literature exceed this performance, however these either exploit word-boundary information (which must be presented at test-time), or are ensembles of multiple networks, or are trained on additional data resources.

Our EfficientNetV2-T outperforms all tabulated models of the literature, while at the same time being 4 times leaner than the ResNet-18 of [5]. Compared to other resource-efficient VSR models like ShuffleNetV2 [5], it yields a 3% absolute WAcc improvement, while also being slightly faster. Scaling the width or depth of the network yields further improvements. Indeed, the EfficientNetV2-M variant achieves 0.49% absolute improvement over the T-variant, and EfficientNetV2-L achieves a new state-of-the-art result of 89.52%, a 1.16% absolute WAcc improvement over [7]. A summary of such VSR system comparisons can also be viewed in Fig. 3.

5. CONCLUSIONS

In this paper, we focused on resource-efficient end-to-end deep-learning based lipreading, making three contributions on efficient VSR architectures. First, we presented EfficientNetV2-T, a lean model that outperforms other resource-efficient VSR systems, and subsequently EfficientNetV2-L, an up-scaled version that improves the state-of-the-art WAcc result on LRW by 1.16% absolute. Finally, we identified and resolved a deficiency in the leading 3D front-end of the literature, and we showed that combining a Transformer encoder with a TCN is beneficial.

6. REFERENCES

[1] J. S. Chung, A. Senior, O. Vinyals, and A. Zisserman, "Lip reading sentences in the wild," in *Proc. CVPR*, 2017.

[2] N. Shrivastava, A. Saxena, Y. Kumar, R. R. Shah, D. Mahata, and A. Stent, "MobiVSR: A visual speech recognition solution for mobile devices," *CoRR*, arXiv:1905.03968v3, 2019.

[3] A. Koumparoulis and G. Potamianos, "MobiLipNet: Resource-efficient deep learning based lipreading," in *Proc. Interspeech*, 2019.

[4] A. Koumparoulis, G. Potamianos, S. Thomas, and E. S. Morais, "Resource-adaptive deep learning for visual speech recognition," in *Proc. Interspeech*, 2020.

[5] P. Ma, B. Martinez, S. Petridis, and M. Pantic, "Towards practical lipreading with distilled and efficient models," in *Proc. ICASSP*, 2021.

[6] J. S. Chung and A. Zisserman, "Lip reading in the wild," in *Proc. ACCV*, 2016.

[7] P. Ma, Y. Wang, J. Shen, S. Petridis, and M. Pantic, "Lipreading with densely connected temporal convolutional networks," in *Proc. WACV*, 2021.

[8] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. ICML*, 2019.

[9] M. Tan and Q. V. Le, "EfficientNetV2: Smaller models and faster training," in *Proc. ICML*, 2021.

[10] 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," in *Proc. ICCV*, 2019.

[11] Y. M. Assael, B. Shillingford, S. Whiteson, and N. de Freitas, "LipNet: End-to-end sentence-level lipreading," *CoRR*, arXiv:1611.01599v2, 2016.

[12] T. Stafylakis and G. Tzimiropoulos, "Combining residual networks with LSTMs for lipreading," in *Proc. Interspeech*, 2017.

[13] T. Stafylakis, M. H. Khan, and G. Tzimiropoulos, "Pushing the boundaries of audiovisual word recognition using residual networks and LSTMs," *Comp. Vision Image Unders.*, 176-177: 22–32, 2018.

[14] C. Wang, "Multi-grained spatio-temporal modeling for lipreading," in *Proc. BMVC*, 2019.

[15] X. Weng and K. Kitani, "Learning spatio-temporal features with two-stream deep 3D CNNs for lipreading," in *Proc. BMVC*, 2019.

[16] M. Luo, S. Yang, S. Shan, and X. Chen, "Pseudo-convolutional policy gradient for sequence-to-sequence lip-reading," in *Proc. FG*, 2020.

[17] X. Zhang, F. Cheng, and S. Wang, "Spatio-temporal fusion based convolutional sequence learning for lip reading," in *Proc. ICCV*, 2019.

[18] J. Xiao, S. Yang, Y. Zhang, S. Shan, and X. Chen, "Deformation flow based two-stream network for lip reading," in *Proc. FG*, 2020.

[19] X. Zhao, S. Yang, S. Shan, and X. Chen, "Mutual information maximization for effective lip reading," in *Proc. FG*, 2020.

[20] Y. Zhang, S. Yang, J. Xiao, S. Shan, and X. Chen, "Can we read speech beyond the lips? Rethinking RoI selection for deep visual speech recognition," in *Proc. FG*, 2020.

[21] B. Martinez, P. Ma, S. Petridis, and M. Pantic, "Lipreading using temporal convolutional networks," in *Proc. ICASSP*, 2020.

[22] D. Feng, S. Yang, S. Shan, and X. Chen, "Learn an effective lip reading model without pains," *CoRR*, arXiv:2011.07557, 2020.

[23] H. Chen, J. Du, Y. Hu, L.-R. Dai, B.-C. Yin, and C.-H. Lee, "Automatic lip-reading with hierarchical pyramidal convolution and self-attention for image sequences with no word boundaries," in *Proc. Interspeech*, 2021.

[24] M. Luo, S. Yang, X. Chen, Z. Liu, and S. Shan, "Synchronous bidirectional learning for multilingual lip reading," in *Proc. BMVC*, 2020.

[25] S. Elashmawy, M. Ramsis, H. M. Eraqi, F. Eldeshnawy, H. Mabrouk, O. Abugabal, and N. Sakr, "Spatio-temporal attention mechanism and knowledge distillation for lip reading," *CoRR*, arXiv:2108.03543, 2021.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NeurIPS*, 2017.

[27] S. Petridis, T. Stafylakis, P. Ma, F. Cai, G. Tzimiropoulos, and M. Pantic, "End-to-end audiovisual speech recognition," in *Proc. ICASSP*, 2018.

[28] 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," *CoRR*, arXiv:1704.04861v1, 2017.

[29] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. CVPR*, 2018.

[30] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. CVPR*, 2018.

[31] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. CVPR*, 2018.

[32] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," *Neural Networks*, 107: 3–11, 2018.

[33] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *Proc. ICML*, 2017.

[34] S. Ren, X. Cao, Y. Wei, and J. Sun, "Face alignment at 3000 fps via regressing local binary features," in *Proc. CVPR*, 2014.

[35] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *Proc. ECCV*, 2016.

[36] E. D. Cubuk, B. Zoph, J. Shlens, and Q. Le, "RandAugment: Practical automated data augmentation with a reduced search space," in *Proc. NeurIPS*, 2020.

[37] P. Wiriyathammabhum, "SpotFast networks with memory augmented lateral transformers for lipreading," in *Proc. ICONIP*, 2020.

[38] H. Liu, Z.Chen, and B. Yang, "Lip graph assisted audio-visual speech recognition using bidirectional synchronous fusion," in *Proc. Interspeech*, 2020.

[39] P. Ma, R. Mira, S. Petridis, B. W. Schuller, and M. Pantic, "LiRA: Learning visual speech representations from audio through self-supervision," in *Proc. Interspeech*, 2021.

[40] D. Tsourounis, D. Kastaniotis, and S. Fotopoulos, "Lip reading by alternating between spatiotemporal and spatial convolutions," *J. Imaging*, 7(5):91, 2021.

[41] M. Hao, M. Mamut, N. Yadikar, A. Aysa, and K. Ubul, "How to use time information effectively? Combining with time shift module for lipreading," in *Proc. ICASSP*, 2021.