Deep Lip Reading: a comparison of models and an online application

Triantafyllos Afouras, Joon Son Chung, Andrew Zisserman

Visual Geometry Group, Department of Engineering Science,
University of Oxford, UK

{afourast,joon,az}@robots.ox.ac.uk

Abstract

The goal of this paper is to develop state-of-the-art models for lip reading – visual speech recognition. We develop three architectures and compare their accuracy and training times: (i) a recurrent model using LSTMs; (ii) a fully convolutional model; and (iii) the recently proposed transformer model. The recurrent and fully convolutional models are trained with a Connectionist Temporal Classification loss and use an explicit language model for decoding, the transformer is a sequence-to-sequence model. Our best performing model improves the state-of-the-art word error rate on the challenging BBC-Oxford Lip Reading Sentences 2 (LRS2) benchmark dataset by over 20 percent.

As a further contribution we investigate the fully convolutional model when used for online (real time) lip reading of continuous speech, and show that it achieves high performance with low latency.

Index Terms: lip reading, visual speech recognition

1. Introduction

In recent years, there has been a quantum leap in the performance of visual speech recognition systems, thanks to the advances in deep learning techniques [1, 2, 3, 4] and the availability of large-scale datasets [5, 6].

In this paper, we propose three new lip reading neural network models based on recently proposed sequence learning methods that have been used successfully for machine translation and automatic speech recognition (ASR). There are two main strands in sequence modelling, namely using an encoder-decoder architecture with soft-attention [7, 8, 9] ('sequence to sequence'), or using CTC [10, 11]. We select two models that use CTC – a recurrent model with LSTMs, and a fully convolutional model, and from the family of attention-based methods, we use the recently proposed Transformer [12] which is the current state-of-the-art in machine translation.

We make the following four contributions: first, we propose three complementary new models for lip reading. For two of these, we adapt architectures developed for other domains, namely machine translation and ASR, and repurpose them for lip reading for the first time. Second, we compare the strengths and weaknesses of these architectures in terms of performance accuracy, training time, generalization at test time, and ease of use; third, we achieve a new state-of-the-art on the public BBC-Oxford Lip Reading Sentences 2 (LRS2) benchmark dataset; finally, we consider modifications that enable on-line lip reading, so that transcriptions are available immediately, and not restricted to utterance-in, utterance-out.

On-line lip reading opens up a host of new applications, such as real-time speech captioning in noisy environments.

1.1. Related works

Research on lip reading has a long history, and has received an increasing amount of attention in recent years. Large scale datasets for lip reading are now available such as the Lip Reading in the Wild (LRW) [5, 13] and LRS2 [6].

For character-level recognition of visual sequences, the prior work can be divided into two strands. The first strand uses CTC, where the model predicts frame-wise labels and then looks for the optimal alignment between the frame-wise predictions and the output sequence. An example based on this approach is LipNet [14], which uses a spatio-temporal front-end, with 3D and 2D convolutions for generating the features, followed by two layers of BLSTM.

The second strand is sequence-to-sequence models that first read the input sequence before predicting the output sentence. An example of this is the LSTM based encoder-decoder architecture with attention of [6], where the model can also combine the audio and visual input streams. This work is extended in [15], where a wider variety of poses is added to the dataset and multi-view models are trained.

A deeper architecture than LipNet [14] is used by [16], who propose a residual network with 3D convolutions to extract more powerful representations. The network is trained with a cross-entropy loss to recognise words from the LRW dataset. Here, the standard ResNet architecture [3] is modified to process 3D image sequences by changing the first convolutional and pooling blocks from 2D to 3D. An extended version of this architecture is used for jointly modeling audio and video by [17].

While both encoder-decoder and CTC based approaches initially relied on recurrent networks, recently there has been a shift towards purely convolutional models [18]. For machine translation, [19] replace the encoder and [20] the whole pipeline with a fully-convolutional model. Encoder-decoder architectures based on dilated convolutions have been also used for translation [21] and speech synthesis [22], while [23] suggests using depth-separable convolutions [24] instead. Fully convolutional networks have been recently proposed for ASR with CTC [25, 26] or a simplified variant [27, 28, 29].

For online sequence-to-sequence prediction, [30] uses attention but constrains it to be monotonic, which allows the alignment to be computed online, while [31] replaces soft with hard attention, which is trained with a policy gradient method and does not require the whole input sequence to be available in order to start decoding. For training online models with CTC, [32] use a teacher-student approach, where an offline BLSTM based model transfers its knowledge to a unidirectional LSTM student, while [33] use unidirectional RNNs and an expectation-maximization algorithm dealing with long sequence lengths. Alternatively, [34] propose a method trained with dynamic programming that conditions on the partially observed input and allows the model to produce output online.
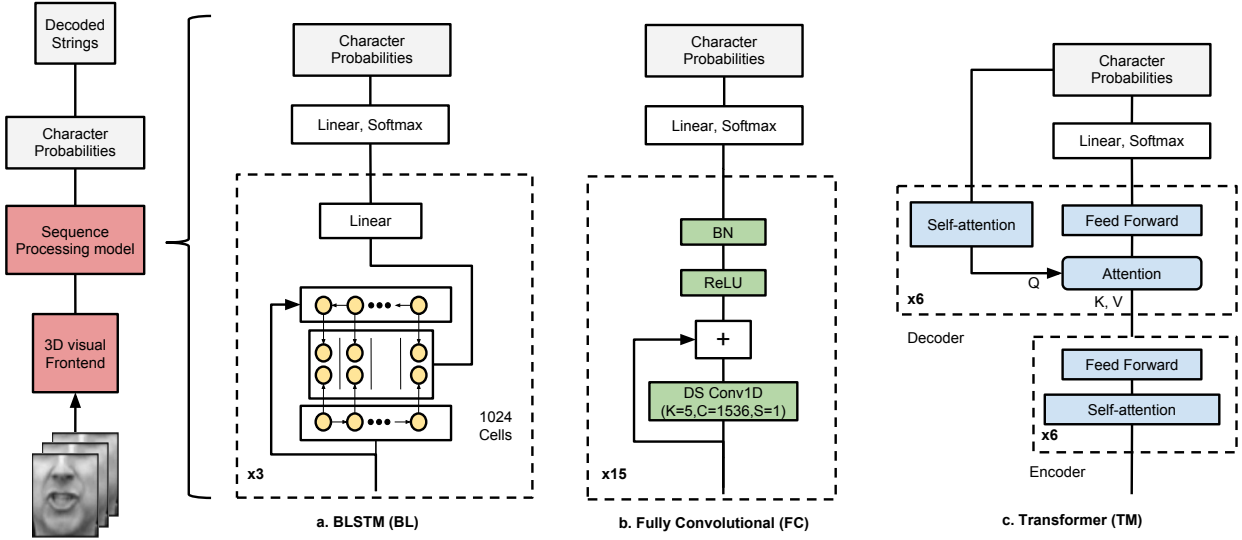
Figure 1: *Lip reading models. The image sequence is first processed by a spatio-temporal ResNet that is common to all models. The visual features are then processed by one of three architectures. (a) BL: The recurrent model consists of a stack of Bidirectional LSTM layers; (b) FC: The fully convolutional model is a deep network formed of depth-separable convolutions. (c) TM: a Transformer model. K, V and Q denote the Key, Value and Query tensors for the multi-head attention.*

2. Architectures

Given a silent video of a talking face, our task is to predict the sentences being spoken. In this section, we propose three deep neural network models for it. In each case the model consists of two modules (or sub-networks): a spatio-temporal visual front-end that inputs a sequence of images of loosely cropped lip regions, and outputs one feature vector per frame; and a sequence processing module that inputs the sequence of per-frame feature vectors and outputs a sentence character by character. The visual front-end is common across the three models, they only differ in the sequence transcription. We briefly describe each of these modules in the following, and illustrate them in Figure 1.

2.1. Vision Module (VM)

The spatio-temporal visual front-end is based on [16]. The network applies a spatio-temporal (3D) convolution on the input image sequence, with a filter width of five frames, followed by a 2D ResNet that gradually decreases the spatial dimensions with depth (for full detail please refer to the supplementary material). For an input sequence of $T \times H \times W$ frames, the output is a $T \times \frac{H}{32} \times \frac{W}{32} \times 512$ tensor (*i.e.* the temporal resolution is preserved) that is then average-pooled over the spatial dimensions, yielding a 512-dimensional feature vector for every input video frame.

2.2. Bidirectional LSTM (BL)

This is the first of the three sequence transcription modules that we compare. It consists of three stacked bidirectional LSTM (BLSTM) recurrent layers. The first BLSTM layer ingests the vision feature vectors, and the final BLSTM layer emits a character probability for every input frame. The BLSTM have 1024 cells each. The implementation of the BL network is similar to the one used by LipNet [14]. The network is trained with CTC. The output alphabet is therefore augmented with the CTC blank character, and the decoding is performed with a beam search that incorporates prior information from an external language model [35, 36].

2.3. Fully Convolutional (FC)

The network consists of a number of temporal convolutional layers. We use depth-wise separable convolution layers [24], that consist of a separate convolution along the time dimen-

sion for every channel, followed by a projection along the channel dimensions (a position-wise convolution with filter width 1). After each convolution we add a shortcut connection, followed by Batch Normalization, and ReLU. The FC network is also trained with a CTC loss, with sequences decoded by using a beam search that incorporates the external language model (above). We consider two variants: one with 10 convolutional layers (FC-10), and a deeper one with 15 convolutional layers (FC-15).

2.4. Transformer model (TM)

The Transformer [12] model has an encoder-decoder structure with multi-head attention layers used as building blocks. The encoder is a stack of self-attention layers, where the input tensor serves as the attention queries, keys and values at the same time. Every decoder layer attends on the embeddings produced by the encoder using common soft-attention: the encoder outputs are the attention keys and values and the previous decoding layer outputs are the queries. The information about the sequence order of the encoder and decoder inputs is fed to the model via fixed positional embeddings in the form of sinusoid functions. The decoder produces character probabilities which are directly matched to the ground truth labels and trained with a cross-entropy loss. We use the base model [12] as is, with 6 encoder and 6 decoder layers, model size 512, 8 attention heads and dropout with $p = 0.1$. The TM does not require an explicit language model for decoding, since it learns an implicit one during training on the visual sequences. However, integrating an external language model in the decoding process has been shown to be beneficial [37].

2.5. External Language Model (LM)

During inference we use a character-level language model, which is a recurrent network with 4 unidirectional layers of 1024 LSTM cells each. The LM is trained to predict one character at a time. Decoding is performed with a left-to-right beam search where the LM log-probabilities are combined with the model's outputs via shallow fusion [37]. This is common for all models, however the beam search is slightly more complicated in the CTC case. For more details refer to the appendix.

Net	Method	# p	CER Greedy	CER T2	WER Greedy	WER T1	WER T2	t/b (s)	time
B	MV-WAS [15]	-	-	-	-	70.4%	-	-	-
BL	BLSTM + CTC	67M	40.6%	38.0%	76.5%	62.9%	62.2%	0.76	4.5d
FC-10	FC×10 + CTC	24M	37.1%	35.0%	69.1%	58.2%	57.1%	0.23	2.4d
FC-15	FC×15 + CTC	35M	35.3%	33.9%	64.8%	56.3%	55.0%	0.34	3.4d
TM	Transformer	40M	38.6%	34.0%	58.0%	51.2%	**50.0%**	0.41	13d

Table 1: *Character error rates (CER) and word error rates (WER) on the LRS2 dataset (lower is better). In the case of T1, we use a LM trained on the corpus explicitly to decode the CTC models, whereas the TM model learns the corpus implicitly during training. For T2, the external LM is explicitly integrated at inference time for all models. Greedy denotes decoding without beam search. #p denotes the total number of parameters of the model (excluding the visual front-end), t/b the processing time for a single batch of 100 samples of 60 frames, and time the total time for completing the training curriculum on a single GPU (d=days). The time to train the visual front-end (2 weeks) is excluded from the statistics.*

3. Experiments & Results

3.1. Datasets and evaluation measures

For training and evaluation, we use the Lip Reading in the Wild (LRW) and the Lip Reading Sentences 2 (LRS2) datasets. LRW consists of approximately 489K samples, each containing the utterance of a single word out of a vocabulary of 500. The videos have a fixed length of 29 frames, the target word occurring in the middle of the clip and surrounded by co-articulation. All of the videos are either frontal or near-frontal. The LRS2 dataset contains sentences of up to 100 characters from BBC videos, with a range of viewpoints from frontal to profile. The dataset is extremely challenging due to the variety in viewpoint, lighting conditions, genres and the number of speakers. The training data contains over 2M word instances and a vocabulary of over 40K.

We also make use of the MV-LRS dataset used in [6], from which we extract individual words to obtain additional word-level pre-training data. This auxiliary word-level set will be referred to as MV-LRS(w). Both MV-LRS and LRS2 have "pre-train" sets that contain sentence excerpts which may be shorter than the full sentences included in the train sets and are annotated with the alignment boundaries of every word.

The statistics on these datasets are summarised in Table 3 of the appendix.

Datasets for training external language models. We use two different text corpora to train the language models. The first, $T1$, only contains the transcriptions of the LRS2 pre-train and main train data ($2M$ words), and therefore the same information that is provided with teacher forcing via the decoder inputs to the **TM** model during training. The second set, $T2$, of $26M$ words, contains the full subtitles of all the videos from which the LRS2 training set is generated (*i.e.* $T1$ is a subset of $T2$).

Evaluation measures. We evaluate the models on the LRS2 test set that consists of 1,243 utterances. We report Character Error Rates (CER) and Word Error Rates (WER) on the LRS2 test set, along with the number of parameters, the computation time for a single mini-batch and the total training time for each model. The error rates are defined as the normalized edit distance between the ground truth and predicted sentences.

3.2. Training protocol

The training proceeds in three stages: first, the visual front-end module is trained; second, visual features are generated for all the training data using the vision module; third, the sequence processing module is trained.

Pre-training visual features. For the first stage, we pre-train the visual front-end on the word-level datasets (LRW and MV-LRS(w)) following [16], where a 2 layer temporal convolution network is used to classify every talking head with a word label. The input video frames are converted to greyscale, scaled and centrally cropped. We also perform data augmentation in the form of horizontal flipping, removal of random frames [14, 16], and random shifts of up to ±5 pixels in the spatial dimension

and of ±2 frames in the temporal dimension.

Curriculum learning. After pre-training the visual module, we proceed with training the sequence processing networks. We first pass all the videos through the pre-trained front-end to obtain the visual features. We then train the sequence models directly on the features, using a strategy similar to [6], that starts with utterances of 2 words then of 2 and 3 words then {2, 3, 4} etc. Since the position of every word in the input video is known, we can choose any continuous sentence excerpt contained in the dataset, calculate the corresponding indices in the visual features sequence and load the features extracted from the video frames containing the utterance. This approach helps to accelerate the training procedure. We first train the network on the MV-LRS and the "pre-train" part of the LRS2 dataset, and finally fine-tune on the "train" set of LRS2. We deal with the difference in utterance lengths by zero-padding them to a maximum sequence length, which we gradually increase along with the maximum number of words used at every step of the curriculum.

Training details. The **TM** is trained using teacher forcing – we supply the ground truth of the previous decoding step as the input to the decoder, while during inference we feed back the decoder prediction. The network is trained with dropout [38] with probability 0.3 on the inputs and the recurrent units of the BLSTM layers. The **FC** uses dropout with probability 0.8 after each every batch normalisation layer. For the **BL** architecture we use SGD with a fixed momentum of 0.9 and learning rate starting at 10^{-2} and reducing it every time the error plateaus, down to 10^{-4}. For the **FC** and **TM** we use the ADAM optimiser [39] with the default parameters and initial learning rate 10^{-3}, reducing it on plateau down to 10^{-4}. All the models are implemented in TensorFlow and trained on a single GeForce GTX 1080 Ti GPU with 11GB memory.

3.3. Results and Model Comparison

The results are summarized in Table 1. The best performing network is the Transformer, which achieves a WER of 50% when decoded with a language model trained on $T2$, an improvement of over 20% compared to the previous 70.4% state-of-the-art [6].

The FC model. The fully convolutional model has a smaller number of parameters and trains faster than **BL** and **TM**, achieving 55% WER. Comparing to the 10-layer architecture **FC-10**, the 5 additional layers contribute a 2% reduction in WER. We believe this improvement to be mostly due to the wider total receptive field which gives the model more context for every prediction. Using depth-separable convolutions doubles the network training speed, without negatively affecting the accuracy. With the **FC** architecture, we have fine-grained control over the amount of future and past context by adjusting the receptive field. We cannot constrain this in the same way when using either the **BL** or **TM** models, since for both the entire input sequence needs to be available at inference time. This en-

frame #	Decoded string	frame #	Decoded string
02	i	02	one
04	he re	07	to
07	on	10	it in
08	a	11	on it
09	we what	12	to on
10	we we	13	to how
11	we have	14	at home
12	we do	26	at home
13	we did	27	at home and
15	we did	28	home
17	we did	29	home to
18	we did it	32	home to
20	we did it	33	home to your
21	we didn't have	34	home to your
22	we didn't have	38	home you
23	we did live	40	home you are
24	we didn't have	41	home you and
25	we did different	45	home to you and
27	we did different	46	home to you and had
gt	we did a different	**gt**	home to an animal

Table 2: *Online decoding examples. Red color denotes the completions of words by the language model. The last line contains the ground truth transcriptions of the excerpt.*

ables us to perform online decoding with **FC**, as described in more detail in the next section.

The BL model. We obtain worse performance with **BL** compared to **FC-10**, even though the recurrent model has full context on every decoding timestep compared to the convolutional that only looks at a limited time-window of the input. We suspect that this is in part due to the CTC loss having a local nature: the output labels are not conditioned on each other and a monotonic alignment is enforced. Therefore the capacity of the BLSTM to learn long-term, non-linear dependences cannot be fully exploited for modelling complex grammar rules.

Language modelling. For all models we get an improvement of 0.7 - 1.3 % in WER when decoding with T2 compared to T1.

Training time. **TM** and **FC-15** both take approximately the same amount of time to complete a batch. Every layer of both models has a $O(td^2)$ complexity (for $t < d$), where d is the layer's width (number of channels). **TM**'s layers have smaller width (every self-attention block has a base width of 512 channels and it is followed by two position-wise fully connected layers with 2048 and 512, compared to 1536 for the **FC**), but it is effectively a deeper model, with $3(6 + 6) = 24$ layers in total. However **FC-15** takes fewer iterations to train, completing the full curriculum in 3.5 days, compared to 13 days for **TM**. We hypothesize that this is due to the Transformer model being tasked with learning the self-attention weights, the encoder-decoder attention, and an implicit language model. In contrast, the **FC**'s task of learning the character-emission probabilities given a fixed context is simpler. The **BL** naturally takes more time for processing one batch, since the computations within its layers have to be run sequentially, in contrast to the other two models. However it converges in fewer epochs, consequently even though the time per iteration for **BL** is almost double that of **FC-10**, it takes only one extra day to train in total.

Generalization to longer sequences. The **FC** model generalises well to longer sequences once it has been trained on sentences that are long enough to cover its full receptive field. We start observing diminishing returns in terms of accuracy gains when training on sequences longer than 80 frames. We had similar findings with **BL**. We could not get the **TM** model to generalize as well when evaluating on longer sequences than seen during training and, therefore we continued the curriculum in order to cover the length up to the longest sample in the validation set.

4. Online lip reading

In this section we describe how the **FC** model can be used for online lip reading with low latency. One advantage of using the temporal convolutions is that we can control how much future context we want to allow the model to see. In contrast, when using bidirectional recurrent networks, or any model with vanilla attention, the entire input sequence needs to be available at the start of the inference. Every temporal convolution with filter width K contributes $\frac{K-1}{2}$ future frames to the overall receptive field. The total receptive field of a network with L similar layers is $R = L \times \frac{K-1}{2} \times 2 + 1$ frames, which allows it to peek up to $r = L \times \frac{K-1}{2}$ frames into the future. In our setting with $K = 5$, r is equal to 22 and 32 frames for the 11 and 16 layer models respectively (here we also take into account the contribution of the front-end's 3D convolutions).

Training with CTC is known to result in peaky distributions [40, 41, 42]. In practice we find that the network emits a character with high probability when the frames that trigger it are under the center of its receptive field. In an online setting we would receive one input video frame at a time. To obtain the same decodings as when running offline, it is sufficient to apply the convolutions on the incoming frames with a time lag of r frames: At the decoding time step t the network's receptive field is centred at frame $t - r$ and emits a distribution p_t^{ctc}, peeking r frames into the future. The beam search step can be run iteratively on the probabilities p_t^{ctc}, scoring them with the language model and accumulating them into the running hypotheses. The final prediction is the same as the offline case.

However, since the network is trained on variable length inputs, it is able to handle partial sentences. For every decoding time-step of the loop described above, we can run additional r beam search steps as if the sentence would end at the current frame. In this manner, we can make predictions in real time on every time step with an additional computation overhead proportional to the size of the receptive field. Using convolutions requires only $O(r)$ new computations for the network forward pass to obtain the CTC emission probabilities and then an extra $O(rW|A|)$ to run the Beam Search, where $|A|$ is the alphabet size and W the beam search width, overall resulting in linear time complexity, $O(TrW|A|)$. We summarize the procedure in Algorithm 2 in the appendix.

Finally, on every decoding time step we can predict further into the future by querying the language model. We show examples of online decoding in Table 2, where the endings of incomplete words of the current beam state are filled in by the language model.

5. Conclusion

We have proposed and compared three new neural network architectures for lip reading, and exceeded the previous state-of-the-art by a large margin. The networks will be publicly released. We have also carried out a preliminary investigation of on-line lip reading and proposed a decoding algorithm for this. Future work could include varying the activations (e.g. Maxout or PReLU as in [26]). Another strand to investigate is whether outputting phonemes and byte-pairs rather than characters, as is now standard for ASR, would lead to a boost in performance.

6. References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *NIPS*, 2012. 1

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015. 1

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015. 1, 6

[4] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. ICML*, 2015. 1

[5] J. S. Chung and A. Zisserman, "Lip reading in the wild," in *Proc. ACCV*, 2016. 1

[6] J. S. Chung, A. Senior, O. Vinyals, and A. Zisserman, "Lip reading sentences in the wild," in *Proc. CVPR*, 2017. 1, 3

[7] I. Sutskever, O. Vinyals, and Q. Le, "Sequence to sequence learning with neural networks," in *NIPS*, 2014. 1

[8] K. Cho, B. van Merriënboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," in *EMNLP*, Oct 2014. 1

[9] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2014. 1

[10] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*. ACM, 2006. 1

[11] A. Graves, "Sequence transduction with recurrent neural networks," *CoRR*, vol. abs/1211.3711, 2012. 1

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," in *NIPS*, 2017. 1, 2

[13] J. S. Chung and A. Zisserman, "Learning to lip read words by watching videos," *CVIU*, 2018. 1

[14] Y. M. Assael, B. Shillingford, S. Whiteson, and N. de Freitas, "Lipnet: Sentence-level lipreading," *arXiv preprint arXiv:1611.01599*, 2016. 1, 2, 3

[15] J. S. Chung and A. Zisserman, "Lip reading in profile," in *Proc. BMVC.*, 2017. 1, 3

[16] T. Stafylakis and G. Tzimiropoulos, "Combining residual networks with lstms for lipreading," in *Interspeech*, 2017. 1, 2, 3, 6

[17] S. Petridis, T. Stafylakis, P. Ma, F. Cai, G. Tzimiropoulos, and M. Pantic, "End-to-end audiovisual speech recognition," *CoRR*, vol. abs/1802.06424, 2018. 1

[18] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018. 1

[19] J. Gehring, M. Auli, D. Grangier, and Y. Dauphin, "A convolutional encoder model for neural machine translation," in *ACL*, 2017. 1

[20] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *ICML*, 2017. 1

[21] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. van den Oord, A. Graves, and K. Kavukcuoglu, "Neural machine translation in linear time," *CoRR*, vol. abs/1610.10099, 2016. 1

[22] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in *ISCA Speech Synthesis Workshop*, 2016. 1

[23] L. Kaiser, A. N. Gomez, and F. Chollet, "Depthwise separable convolutions for neural machine translation," *arXiv preprint arXiv:1706.03059*, 2017. 1

[24] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. CVPR*, 2017. 1, 2

[25] Y. Wang, X. Deng, S. Pu, and Z. Huang, "Residual Convolutional CTC Networks for Automatic Speech Recognition," *arXiv preprint arXiv:1702.07793*, 2017. 1

[26] Y. Zhang, M. Pezeshki, P. Brakel, S. Zhang, C. Laurent, Y. Bengio, and A. C. Courville, "Towards end-to-end speech recognition with deep convolutional neural networks," *CoRR*, vol. abs/1701.02720, 2017. 1, 4

[27] R. Collobert, C. Puhrsch, and G. Synnaeve, "Wav2letter: an end-to-end convnet-based speech recognition system," *CoRR*, vol. abs/1609.03193, 2016. 1

[28] V. Liptchinsky, G. Synnaeve, and R. Collobert, "Letter-based speech recognition with gated convnets," *CoRR*, vol. abs/1712.09444, 2017. 1

[29] N. Zeghidour, N. Usunier, I. Kokkinos, T. Schatz, G. Synnaeve, and E. Dupoux, "Learning filterbanks from raw speech for phone recognition," *CoRR*, vol. abs/1711.01161, 2017. 1

[30] C. Raffel, T. Luong, P. J. Liu, R. J. Weiss, and D. Eck, "Online and linear-time attention by enforcing monotonic alignments," *CoRR*, vol. abs/1704.00784, 2017. 1

[31] Y. Luo, C.-C. Chiu, N. Jaitly, and I. Sutskever, "Learning online alignments with continuous rewards policy gradient," *arXiv preprint arXiv:1608.01281*, 2017. 1

[32] S. Kim, M. L. Seltzer, J. Li, and R. Zhao, "Improved training for online end-to-end speech recognition systems," *arXiv preprint arXiv:1711.02212*, 2017. 1

[33] K. Hwang and W. Sung, "Online sequence training of recurrent neural networks with connectionist temporal classification," *arXiv preprint arXiv:1511.06841*, 2017. 1

[34] N. Jaitly, Q. V. Le, O. Vinyals, I. Sutskever, D. Sussillo, and S. Bengio, "An online sequence-to-sequence model using partial conditioning," in *NIPS*, 2016. 1

[35] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ser. ICML, 2014. 2

[36] A. L. Maas, Z. Xie, D. Jurafsky, and A. Y. Ng, "Lexicon-free conversational speech recognition with neural networks," in *Proceedings the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2015. 2, 6

[37] A. Kannan, Y. Wu, P. Nguyen, T. N. Sainath, Z. Chen, and R. Prabhavalkar, "An analysis of incorporating an external language model into a sequence-to-sequence model," *arXiv preprint arXiv:1712.01996*, 2017. 2, 6

[38] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, 2014. 3

[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015. 3

[40] A. Zeyer, E. Beck, R. Schlüter, and H. Ney, "CTC in the Context of Generalized Full-Sum HMM Training," in *INTERSPEECH*, 2017. 4

[41] Z. Chen, Y. Zhuang, Y. Qian, K. Yu, Z. Chen, Y. Zhuang, Y. Qian, K. Yu, K. Yu, Y. Zhuang, Z. Chen, and Y. Qian, "Phone synchronous speech recognition with ctc lattices," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 2017. 4

[42] A. Rosenberg, K. Audhkhasi, A. Sethy, B. Ramabhadran, and M. Picheny, "End-to-end speech recognition and keyword search on low-resource languages," in *ICASSP*, 2017. 4

[43] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's neural machine translation system: Bridging the gap between human and machine translation," *CoRR*, vol. abs/1609.08144, 2016. 6

A. Appendix

A.1. Dataset statistics

The statistics of the datasets used in this paper is given in Table 3.

Name	Type	Vocab	#Utter.	#Words
LRW	word	500	-	489K
MV-LRS(w) ∗	word	480	-	1,9M
MV-LRS ∗	sent.	30K	430K	5M
LRS2	sent.	41K	142K	2M
T1	text	41K	142K	2M
T2 ∗	text	60K	8M	26M

Table 3: *Description of the datasets used for training and testing. We formed MV-LRS(w) by isolating individual word excerpts of the 480 most frequent words, all of which have a count of at least 1000 samples. The statistics for the MV-LRS and the LRS2 datasets include the noisy "pre-train" sets in addition to the main dataset. T1 consists of the transcriptions of the samples in LRS2. We form T2 by collecting the full transcripts of all the subtitles of the shows used in the making of LRS2. The sets marked with ∗ are not publicly available.*

A.2. Visual front-end architecture

The details of the spatio-temporal front-end are given in Table 4.

Layer Type	Filters	Output dimensions
Conv 3D	$5 \times 7 \times 7, 64, /[1, 2, 2]$	$T \times \frac{H}{2} \times \frac{W}{2} \times 64$
Max Pool 3D	$/[1, 2, 2]$	$T \times \frac{H}{4} \times \frac{W}{4} \times 64$
Residual Conv 2D	$[3 \times 3, 64] \times 2 \ /1$	$T \times \frac{H}{4} \times \frac{W}{4} \times 64$
Residual Conv 2D	$[3 \times 3, 64] \times 2 \ /1$	$T \times \frac{H}{4} \times \frac{W}{4} \times 64$
Residual Conv 2D	$[3 \times 3, 128] \times 2 \ /2$	$T \times \frac{H}{8} \times \frac{W}{8} \times 128$
Residual Conv 2D	$[3 \times 3, 128] \times 2 \ /1$	$T \times \frac{H}{8} \times \frac{W}{8} \times 128$
Residual Conv 2D	$[3 \times 3, 256] \times 2 \ /2$	$T \times \frac{H}{16} \times \frac{W}{16} \times 256$
Residual Conv 2D	$[3 \times 3, 256] \times 2 \ /1$	$T \times \frac{H}{16} \times \frac{W}{16} \times 256$
Residual Conv 2D	$[3 \times 3, 512] \times 2 \ /2$	$T \times \frac{H}{32} \times \frac{W}{32} \times 512$
Residual Conv 2D	$[3 \times 3, 512] \times 2 \ /1$	$T \times \frac{H}{32} \times \frac{W}{32} \times 512$

Table 4: *Architecture details for the spatio-temporal visual front-end [16]. The strides for the residual 2D convolutional blocks apply to the first layer of the block only (i.e. the total down-sampling factor in the network is 32). A short cut connection is added after every pair of 2D convolutions [3]. The 2D convolutions are applied separately on every time-frame.*

A.3. Seq2Seq decoding with external language model

For decoding with the TM model, we use a left-to right beam search with width W as in [37, 43], with the hypotheses y being scored as follows:

$$score(x,y) = \frac{log\ p(y|x) + \alpha\ log\ p_{LM}(y)}{LP(y)}$$

where $p(y|x)$ and $p_{LM}(y)$ are the probabilities obtained from the visual and language models respectively and LP is a length normalization factor LP(y) = $\left(\frac{5+|y|}{6}\right)^{\beta}$ [43]. We did not experiment with a coverage penalty. The best values for the hyperparameters were determined via grid search on the validation set: for decoding without the external language model (T1) they

were set to $W = 5$, $\alpha = 0.0$, $\beta = 0.6$ and for decoding with the LM (T2) to $W = 15$, $\alpha = 0.1$ $\beta = 0.7$.

A.4. CTC decoding algorithm with external language model

Algorithm 1 describes the CTC decoding procedure with an external language model. It is also a beam search with width W and hyperparameters α and β that control the relative weight given to the LM and the length penalty. The beam search is similar to the one described for seq2seq above, with some additional bookkeeping required to handle the emission of repeated and blank characters and normalization LP(y) = $|y|^{\beta}$. We obtain the best results with $W = 100$, $\alpha = 0.5$, $\beta = 0.1$.

Algorithm 1 CTC Beam search decoding with Language Model adapted from [36]. Notation: A is the alphabet; $p_b(s,t)$ and $p_{nb}(s,t)$ are the probabilities of partial output transcription s resulting from paths ending in blank and non-blank token respectively, given the input sequence up to time t; $p(s,t) = p_b(s,t) + p_{nb}(s,t)$.

Parameters CTC probabilities $p_{1:T}^{ctc}$, word dictionary, beam width W, hyperparameters α, β
initialize $\mathbf{B_t} \leftarrow \{\varnothing\}$; $\mathbf{p_b}(\varnothing, \mathbf{0}) \leftarrow 1$; $\mathbf{p_{nb}}(\varnothing, \mathbf{0}) \leftarrow 0$
for $t = 1$ **to** T **do**
  $\mathbf{B_{t-1}} \leftarrow W$ prefixes with highest $\frac{log\ p(s,t)}{|s|^{\beta}}$ in $\mathbf{B}_t$
  $\mathbf{B_t} \leftarrow \{\}$
  for prefix s **in** $\mathbf{B_{t-1}}$ **do**
    $c^- \leftarrow$ last character of s
    $p_b(s,t) \leftarrow p_t^{ctc}(-,t)p(s,t-1)$ ▷ adding a blank
    $p_{nb}(s,t) \leftarrow p_t^{ctc}(c^-,t)p_{nb}(s,t-1)$ ▷ repeated
    add s to $\mathbf{B}$
    for character c **in** A **do**
      $s^+ \leftarrow s + c$
      if s ends in c **then**
        $p_c \leftarrow p_t^{ctc}(c,t)p(c,t-1)p_{LM}(c|s)^{\alpha}$
      else
        ▷ repeated chars must have blanks in between
        $p_c \leftarrow p_t^{ctc}(c,t)p_b(c,t-1)p_{LM}(c|s)^{\alpha}$
      if s^+ is already in $\mathbf{B_t}$ **then**
        $p_{nb}(s^+,t) \leftarrow p_{nb}(s^+,t) + p_c$
      else
        add s^+ to $\mathbf{B_t}$
        $p_{nb}(s,t) \leftarrow 0$
        $p_{nb}(s^+,t) \leftarrow p_c$
return $max_{s \in B_t} \frac{log\ p(s,T)}{|s|^{\beta}}$ in $\mathbf{B}_T$

A.5. Online CTC decoding algorithm

Algorithm 2 describes the online CTC decoding procedure introduced in Section 4.

Algorithm 2 Online CTC decoding with fully convolutional model. The algorithm runs in $O(TrW|A|)$ time, where T denotes the input sequence length, r is half the length of the network's total receptive field, W the beam width and $|A|$ the number of characters in the alphabet. The BeamStep routine performs one step of the CTC Beam Search decoding outer loop shown in Algorithm 1

.

   Parameters Input video frames $x_{1:T}$, FC network f_θ
   initialize $\mathbf{B_0} \leftarrow \{\varnothing\}$; $\mathbf{L_0} \leftarrow \{\varnothing\}$;   ▷ Beam & LM states
   for $t = 1$ **to** T **do** ▷ decoding steps lag by r behind real time
      $p_{t:t+r}^{ctc} \leftarrow f_\theta(x_{t-r:t})$ ▷ Slide network right by one step
      $\mathbf{B_t}, \mathbf{L_t} \leftarrow$ BEAMSTEP($p_t^{ctc}, \mathbf{B_t}, \mathbf{L_t}$)   ▷ $O(W \cdot |A|)$
      $\hat{\mathbf{B}}_t, \hat{\mathbf{L}}_t \leftarrow$ copy $\mathbf{B_t}, \mathbf{L_t}$
      for $\tau = t + 1$ **to** $t + r$ **do**
         $\hat{\mathbf{B}}_\tau, \hat{\mathbf{L}}_t \leftarrow$ BEAMSTEP($p_\tau^{ctc}, \hat{\mathbf{B}}_\tau, \hat{\mathbf{L}}_\tau$) ▷ $O(W \cdot |A|)$
      $D_t \leftarrow$ highest scoring sentence $S \in \hat{B}_{t+r}$
   return D_T

A.6. Confusion Matrix

Figure 2 shows the confusion between the predictions of the FC-15 model obtained with greedy decoding of the CTC posteriors.
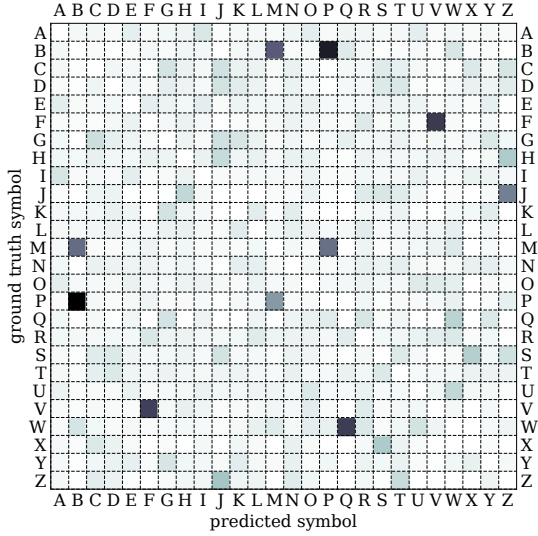
Figure 2: *Confusion matrix of CTC predictions. The entries of the table are the normalized substitution counts obtained from the minimum edit distance calculation between ground truth and sentences predicted with greedy CTC decoding, averaged over the whole dataset. We observe that the network confuses characters that are difficult to discriminate between using the visual information alone. For example B is frequently confused with M and P, while V is confused with F and vice versa. It is interesting to note that Q is often emitted instead of W. We hypothesize that this happens because these characters appear very similar visually in words like 'week / quick', 'woe / quote'.*

A.7. Decoding examples

Table 5 shows further examples of online decoding outputs.

frame #	Decoded string	Decoded string starting from the middle (frame #55)
002	one	
007	to	
010	it in	
011	on it	
012	to on	
013	to how	
014	at home	
026	at home	
027	at home and	
028	home	
029	home to	
032	home to	
033	home to your	
038	home you	
040	home you are	
041	home you and	
045	home to you and	
046	home to you and had	
047	home you and had	
048	home you and adam	
051	home you and animals	
054	home to an animal	
056	home you and animals	i
058		in
059		it in
060	home to an animal	i and
061	home to an animal and	in
062		in the
063	home to an animal that	then the
064		that is
066	home to an animal that	that here
067	home to an animal that is	
068		that is
070		that it's
070		that is
074	home to an animal that is	that it's
075	home to an animal that is right	that it's right
076	home to an animal that it's right	
078	home to an animal that is right	
081	home to an animal that is right in	that is right in
082		that it's right in
083	home to an animal that is right in the	that it's right in the
087	home to an animal that is right in the training	that it's right in the town
089	home to an animal that is right in the town	
090	home to an animal that it's right in the top	that it's right in the top
091	home to an animal that it's right in the top	that it's right in the top
092	home to an animal that it's right in the top of	that it's right in the top of
094	home to an animal that it's right in the top of	that it's right in the top of
097	home to an animal that it's right in the top of the	that it's right in the top of the
098	home to an animal that it's right in the top of the room	that it's right in the top of the front
099		that it's right in the top of the room
100	home to an animal that it's right in the top of the food	that it's right in the top of the foot
101		that it's right in the top of the food
102	home to an animal that it's right in the top of the foot	that it's right in the top of the foot
103	home to an animal that it's right in the top of the future	that it's right in the top of the future
# changes/frame	0.4	0.5

Table 5: *Example of sequential online decoding starting the beginning (**left**) and from the middle of the utterance (**right**). The ground truth transcription is "home to an animal that is right at the top of the food chain". Red color denotes the completions of words by the language model. It can be seen that after some initial frames where the model does not have enough context to make a confident prediction, it starts predicting correctly.*