Attention-Passing Models for Robust and Data-Efficient End-to-End Speech Translation

Matthias Sperber[1], **Graham Neubig**[2], **Jan Niehues**[1], **Alex Waibel**[1,2]

[1]Karlsruhe Institute of Technology, Germany
[2]Carnegie Mellon University, USA
{first}.{last}@kit.edu, gneubig@cs.cmu.edu

Abstract

Speech translation has traditionally been approached through cascaded models consisting of a speech recognizer trained on a corpus of transcribed speech, and a machine translation system trained on parallel texts. Several recent works have shown the feasibility of collapsing the cascade into a single, direct model that can be trained in an end-to-end fashion on a corpus of translated speech. However, experiments are inconclusive on whether the cascade or the direct model is stronger, and have only been conducted under the unrealistic assumption that both are trained on equal amounts of data, ignoring other available speech recognition and machine translation corpora.

In this paper, we demonstrate that direct speech translation models require more data to perform well than cascaded models, and although they allow including auxiliary data through multi-task training, they are poor at exploiting such data, putting them at a severe disadvantage. As a remedy, we propose the use of end-to-end trainable models with two attention mechanisms, the first establishing source speech to source text alignments, the second modeling source to target text alignment. We show that such models naturally decompose into multi-task–trainable recognition and translation tasks and propose an *attention-passing* technique that alleviates error propagation issues in a previous formulation of a model with two attention stages. Our proposed model outperforms all examined baselines and is able to exploit auxiliary training data much more effectively than direct attentional models.

1 Introduction

Speech translation takes audio signals of speech as input and produces text translations as output. Although traditionally realized by cascading an automatic speech recognition (ASR) and a machine translation (MT) component, recent work has shown that it is feasible to use a single sequence-to-sequence model instead (Duong et al., 2016; Weiss et al., 2017; Bérard et al., 2018; Anastasopoulos and Chiang, 2018). An appealing property of such direct models is that we no longer suffer from propagation of errors, where the speech recognizer passes an erroneous source text to the machine translation component, potentially leading to compounding follow-up errors. Another advantage is the ability to train all model parameters jointly.

Despite these obvious advantages, two problems persist: (1) Reports on whether direct models outperform cascaded models (Fig. 1a,d) are inconclusive, with some work in favor of direct models (Weiss et al., 2017), some work in favor of cascaded models (Kano et al., 2017; Bérard et al., 2018), and one work in favor of direct models for two out of the three examined language pairs (Anastasopoulos and Chiang, 2018). (2) Cascaded and direct models have been compared under identical data situations, but this is an unrealistic assumption: In practice, cascaded models can be trained on much more abundant independent ASR and MT corpora, whereas end-to-end models require hard-to-acquire end-to-end corpora of speech utterances paired with textual translations.

Our first contribution is a closer investigation of these two issues. Regarding the question of whether direct models or cascaded models are generally stronger, we hypothesize that direct models require more data to work well, due to the more complex mapping between inputs (source speech) and outputs (target text). This would imply that direct models outperform cascades when enough data are available, but underperform in low-data scenarios. We conduct experiments and present empirical evidence in favor of this

Transactions of the Association for Computational Linguistics, vol. 7, pp. 313–325, 2019. Action Editor: Adam Lopez.
Submission batch: 12/2018; Revision batch: 2/2019; Published 6/2019.

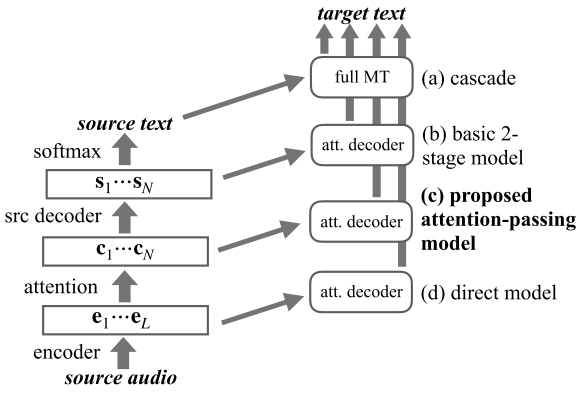

Figure 1: Conceptual diagrams for various speech translation approaches. *Cascade* (a) uses separate machine translation and speech recognition models. The *direct* model (d) is a standard attentional encoder-decoder model. The basic 2-stage model (b) uses two attention stages and passes source-text decoder states to the translation component. Our proposed *attention-passing* model (c) applies two attention stages, but passes context vectors to the translation component for improved robustness.

hypothesis. Next, for a more realistic comparison with regard to data conditions, we train a direct speech translation model using more auxiliary ASR and MT training data than end-to-end data. This can be implemented through multi-task training (Weiss et al., 2017; Bérard et al., 2018). Our results show that the auxiliary data are beneficial only to a limited extent, and that direct multi-task models are still heavily dependent on the end-to-end data.

As our second contribution, we apply a *two-stage* model (Tu et al., 2017; Kano et al., 2017) as an alternative solution to our problem, hoping that such models may overcome the data efficiency shortcoming of the direct model. Two-stage models consist of a first-stage attentional sequence-to-sequence model that performs speech recognition and then passes the decoder states as input to a second attentional model that performs translation (Fig. 1b). This architecture is closer to cascaded translation while maintaining end-to-end trainability. Introducing supervision from the source-side transcripts midway through the model creates inductive bias that guides the complex transformation between source speech and target text through a reasonable intermediate representation closely tied to the source text. The architecture has been proposed by Tu et al. (2017) to realize a reconstruction objective, and a similar model was also applied to speech translation (Kano et al., 2017) to ease trainability, although no experiments under varying data conditions have been conducted. We hypothesize that such a model may help to address the identified data efficiency issue: Unlike multi-task training for the direct model that trains auxiliary models on additional data but then discards many of the additionally learned parameters, the two-stage model uses all parameters of sub-models in the final end-to-end model. Empirical results confirm that the two-stage model is indeed successful at improving data efficiency, but suffers from some degradation in translation accuracy under high data conditions compared with the direct model. One reason for this degradation is that this model re-introduces the problem of error propagation, because the second stage of the model depends on the decoder states of the first model stage which often contain errors.

Our third contribution, therefore, is an *attention-passing* variant of the two-stage model that, rather than passing on possibly erroneous decoder states from the first to the second stage, passes on only the computed attention context vectors (Fig. 1c). We can view this approach as replacing the early decision on a source-side transcript by an early decision only on the *attention scores* needed to compute the same transcript, where the attention scores are expectedly more robust to errors in source text decoding. We explore several variants of this model and show that it outperforms both the direct model and the vanilla two-stage model, while maintaining the improved data efficiency of the latter. Through an analysis, we further observe a trade-off between sensitivity to error propagation and data efficiency.

2 Baseline Models

This section introduces two types of end-to-end trainable models for speech translation, along with a cascaded approach, which will serve as our baselines. All models are based on the attentional encoder-decoder architecture of Bahdanau et al. (2015) with character-level outputs, and use the architecture described in §2.1 as audio encoders. The end-to-end trainable models include a direct model and a two-stage model. Both are limited[1] by the fact that they can *only* be trained on end-to-end

[1]Prior work noted that in severe low-resource situations it may actually be easier to collect speech paired with translations than transcriptions (Duong et al., 2016). However, we focus on well-resourced languages for which ASR

data, which is much harder to obtain than ASR or MT data used to train traditional cascades.[2] §3 will introduce multi-task training as a way to overcome this limitation.

2.1 Audio Encoder

Sequence-to-sequence models can be adopted for audio inputs by directly feeding speech features (here, Mel filterbank features) instead of word embeddings as encoder inputs (Chorowski et al., 2015; Chan et al., 2016). Such an encoder transforms M feature vectors $\mathbf{x}_{1:M}$ into L encoded vectors $\mathbf{e}_{1:L}$, performing downsampling such that $L<M$. We use an encoder architecture that follows one of the variants described by Zhang et al. (2017): We stack two blocks, each consisting of a bidirectional long short-term memory (LSTM), a network-in-network (NiN) projection that downsamples by factor two, and batch normalization. After the second block, we add a final bidirectional LSTM layer. NiN denotes a simple linear projection applied at every time step, performing downsampling by concatenating pairs of adjacent projection inputs. Because of space constraints, we do not present detailed equations, but refer interested readers to Zhang et al. (2017) as well as to our provided code for details.

2.2 Direct Model

The sequence-to-sequence model with audio inputs outlined above can be trained as a direct speech translation model by using speech data as input and the corresponding translations as outputs. Such a model does not rely on intermediate ASR output and is therefore not subject to error propagation. However, the transformation from source speech inputs to target text outputs is much more complex than that of an ASR or MT system taken individually, which may cause the model to require more data to perform well.

To make matters precise, given L audio encoder states $\mathbf{e}_{1:L}$ computed by the audio encoder as described in §2.1, the direct model is computed as

$$\mathbf{s}_i = \text{LSTM}\left(\left[W_{\text{e}} y_{i-1}; \mathbf{c}_{i-1}\right], \mathbf{s}_{i-1}; \theta_{\text{lstm}}\right) \quad (1)$$

$$\mathbf{c}_i = \text{Attention}\left(\mathbf{s}_i, \mathbf{e}_{1:L}; \theta_{\text{att}}\right) \quad (2)$$

$$\tilde{\mathbf{s}}_i = \tanh\left(W_{\text{s}}\left[\mathbf{s}_i; \mathbf{c}_i\right] + \mathbf{b}_{\text{s}}\right) \quad (3)$$

$$p\left(y_i \mid y_{<i}, \mathbf{e}_{1:L}\right) = \text{SoftmaxOut}\left(\tilde{\mathbf{s}}_i; \theta_{\text{out}}\right). \quad (4)$$

Here, W_*, $\mathbf{b}_*$, and θ_* are the trainable parameters, y_i are output characters, and SoftmaxOut denotes an affine projection followed by a softmax operation. $\mathbf{s}_i$ are decoder states with $\mathbf{s}_0$ initialized to the last encoder state, and $\mathbf{c}_i$ are attentional context vectors with $\mathbf{c}_0{=}\mathbf{0}$. In Equation 2, we compute $\text{Attention}(\cdot){=}\sum_{j=1}^{L}\alpha_{ij}\mathbf{e}_j$ with weights α_{ij} conditioned on $\mathbf{e}_j$ and $\mathbf{s}_i$, parameterized by θ_{att}, and normalized via a softmax operation.

2.3 Two-Stage Model

As an alternative to the direct model, the two-stage model uses a cascaded information flow while maintaining end-to-end trainability. Our main motivation for using this model is the potentially improved data efficiency when adding auxiliary ASR and MT training data (§3). This model is similar to the architecture first described by Tu et al. (2017). It combines two encoder-decoder models in a cascade-like fashion, with the decoder of the first stage and the encoder of the second stage being shared (Fig. 2). In other words, while a cascade would use the source-text outputs of the first stage as inputs into the second stage, in this model the second stage directly computes attentional context vectors over the decoder states of the first stage. The inputs of the two-stage model are speech frames, the outputs of the first stage are transcribed characters in the source language, and the outputs of the second stage are translated characters in the target language.

Again assuming L audio encoder states $\mathbf{e}_{1:L}$, the first stage outputs of length N are computed identically to equations 1–4, except that input feeding (conditioning the decoding step on the previous context vector) is not used in the first

and MT corpora exist and for which it is more realistic to obtain good speech translation accuracy.

[2] As a case in point, the largest available speech translation corpora (Post et al., 2013; Kocabiyikoglu et al., 2018) are an order of magnitude smaller than the largest speech recognition corpora (Cieri et al., 2004; Panayotov et al., 2015) ($\sim$ 200 hours vs 2000 hours) and several orders of magnitude smaller than the largest machine translation corpora, e.g., those provided by the Conference on Machine Translation (WMT).

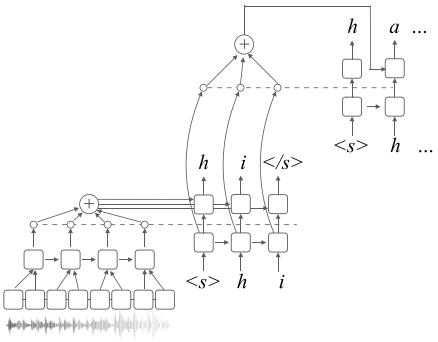

Figure 2: Basic two-stage model. Decoder states of the first stage double as encoder states for the second stage.

stage decoder to keep components compatible for multi-task training (§3.2):

$$\mathbf{s}_i^{\text{src}} = \text{LSTM}\left(W_{\text{e}}^{\text{src}} y_{i-1}^{\text{src}}, \mathbf{s}_{i-1}^{\text{src}}; \theta_{\text{lstm}}^{\text{src}}\right) \quad (5)$$

$$\mathbf{c}_i^{\text{src}} = \text{Attention}\left(\mathbf{s}_i^{\text{src}}, \mathbf{e}_{1:L}; \theta_{\text{att}}^{\text{src}}\right) \quad (6)$$

$$\tilde{\mathbf{s}}_i^{\text{src}} = \tanh\left(W_{\text{s}}^{\text{src}}\left[\mathbf{s}_i^{\text{src}}; \mathbf{c}_i^{\text{src}}\right] + \mathbf{b}_{\text{s}}^{\text{src}}\right) \quad (7)$$

$$p\left(y_i^{\text{src}} \mid y_{<i}, \mathbf{e}_{1:L}\right) = \text{SoftmaxOut}\left(\tilde{\mathbf{s}}_i^{\text{src}}; \theta_{\text{out}}^{\text{src}}\right) \quad (8)$$

Next, the second stage proceeds similarly but uses the stage 1 decoder states as input:

$$\mathbf{s}_j^{\text{trg}} = \text{LSTM}\left(\left[W_{\text{e}}^{\text{trg}} y_{i-1}^{\text{trg}}; \mathbf{c}_{j-1}^{\text{trg}}\right], \mathbf{s}_{j-1}^{\text{trg}}; \theta_{\text{lstm}}^{\text{trg}}\right) \quad (9)$$

$$\mathbf{c}_j^{\text{trg}} = \text{Attention}\left(\mathbf{s}_j^{\text{trg}}, \mathbf{s}_{1:N}^{\text{src}}; \theta_{\text{att}}^{\text{trg}}\right) \quad (10)$$

$$\tilde{\mathbf{s}}_j^{\text{trg}} = \tanh\left(W_{\text{s}}^{\text{trg}}\left[\mathbf{s}_j^{\text{trg}}; \mathbf{c}_j^{\text{trg}}\right] + \mathbf{b}_{\text{s}}^{\text{trg}}\right) \quad (11)$$

$$p\left(y_j^{\text{trg}} \mid y_{<j}, \mathbf{s}_{1:N}^{\text{src}}\right) = \text{SoftmaxOut}\left(\tilde{\mathbf{s}}_j^{\text{trg}}; \theta_{\text{out}}^{\text{trg}}\right) \quad (12)$$

2.4 Cascaded Model

We finally utilize a traditional cascaded model as a baseline, whose architecture is kept as similar to the above models as possible in order to facilitate meaningful comparisons. The cascade consists of an ASR component and an MT component, which are both attentional sequence-to-sequence models according to equations 1–4, trained on the appropriate data. The ASR component uses the acoustic encoder of §2.1, and the MT model uses a bidirectional LSTM with 2 layers as encoder.

3 Incorporating Auxiliary Data

The models described in §2.2 and §2.3 are trained only on speech utterances paired with translations (and transcripts in the case of §2.3), which is a severe limitation. To incorporate auxiliary ASR and MT data into the training, we make use of a multi-task training strategy. Such a strategy trains auxiliary ASR and MT models that share certain parameters with the main speech translation model. We implement multi-task training by drawing several minibatches, one minibatch for each task, and performing an update based on the accumulated gradients across tasks. Note that this results in a balanced contribution of each task.[3]

3.1 Multi-Task Training for the Direct Model

Multi-task training for direct speech translation models has previously been used by Weiss et al. (2017) and Bérard et al. (2018), although not for the purpose of adding additional training utterances that are not shown to the main speech translation task.[4] We distinguish five model components: a source speech encoder, a source text encoder (a two-layer bidirectional LSTM working on character level), a source text decoder, a target text decoder, and an attention mechanism that we opt to share across all tasks. There are four ways in which these components can be combined into a complete sequence-to-sequence model (see Figure 3), corresponding to the following four tasks:

ASR: Combines source speech encoder, general-purpose attention, source text decoder. This is similar to the auxiliary ASR task used by Weiss et al. (2017) and can be trained on common ASR data.

MT: Combines source text encoder, general-purpose-attention, target text decoder. The addition of an MT task has been mentioned by Bérard et al. (2018) and allows training on common MT data.

ST: Combines source speech encoder, general-purpose-attention, target text decoder. This is our main task and requires end-to-end data for training.

[3]We also experimented with a final fine-tuning phase on only the main task (Niehues and Cho, 2017), but discarded this strategy for lack of consistent gains.

[4]Note that Bansal et al. (2019) do experiment with additional speech recognition data, although, differently from our work, for purposes of cross-lingual transfer learning.

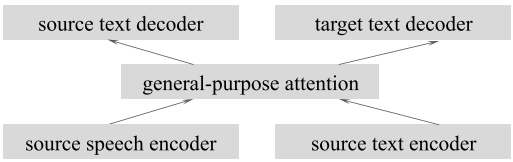

Figure 3: Direct multi-task model.

Auto-encoder (AE): Combines source text encoder, general-purpose attention, source text decoder. The AE task can be trained on monolingual corpora in the source language and may serve to tighten the coupling between components and potentially improves the parameters of the general-purpose attention model. We have observed slight improvements by adding the AE task in preliminary experiments and will therefore use it throughout this paper.

3.2 Multi-Task Training for the Two-Stage Model

Including an auxiliary ASR task is straightforward with the two-stage model by simply computing the cross-entropy loss with respect to the softmax output of the first stage, and dropping the second stage.

The auxiliary MT task computes only the second stage, replacing the inputs $\mathbf{s}^{\text{src}}_{1:N}$ by states $\mathbf{e}^{\text{asr}}_{1:N}$ computed as:

$$\mathbf{e}^{\text{asr}}_i = \text{LSTM}\left(W^{\text{src}}_{\text{e}} y^{\text{transcr}}_i, \mathbf{e}^{\text{src}}_{i-1}; \theta^{\text{src}}_{\text{lstm}}\right). \quad (13)$$

That is, instead of computing the second-stage inputs using the first stage, we compute these inputs through a conventional encoder that encodes the reference transcript $y^{\text{transcr}}_{1:N}$ and uses the same embeddings matrix and unidirectional LSTM as the first stage decoder. Note that there is no equivalent to the auxiliary auto-encoder task of the direct multi-task model here.

Why might this architecture help to make better use of auxiliary ASR and MT data? Note that in the direct model only roughly half of the model parameters are shared between the main task and the ASR task, and likewise for main and MT tasks (§3.1). Additional data would therefore only have a rather indirect impact on the main task. In contrast, in the two-stage model all parameters of the auxiliary tasks are shared with the main task and therefore have a more direct impact, potentially leading to better data efficiency.

Note that somewhat related to our multi-task strategy, Kano et al. (2017) have decomposed their two-stage model in a similar way to perform pretraining for the individual stages, although not with the goal of incorporating additional auxiliary data.

4 Attention-Passing Model

We have so far described a direct model that has the appealing property of avoiding error propagation in a principled way but that may not be particularly data-efficient, and have described a two-stage model that addresses the latter disadvantage. Unfortunately, the two-stage model re-introduces the error propagation problem into end-to-end modeling, because the second stage heavily depends on the potentially erroneous decoder states of the first stage. We therefore propose an improved *attention-passing* model in this section that is less impacted by error propagation issues.

4.1 Preventing Error Propagation

The main idea behind the attention-passing model is to not feed the erroneous first-stage decoder states to the second stage, but instead to pass on only the *context vectors* that summarize the relevant encoded audio at each decoding step. The first stage decoder is unfolded as usual by using discrete source-text representations, but the only information exposed to the translation stage are the per-timestep context vectors created as a by-product of the decoding. Figure 4 illustrates this idea. Intuitively, we expect this to help because we no longer make an early decision on the identity of the source-language text, but only on the corresponding attentions. This is motivated by our observation that speech recognition attentions are sufficiently robust against decoding errors (§5.7).

Formally, the first stage remains unchanged from equations 5–8. The context vectors $\mathbf{c}^{\text{src}}_i$ then form the input to the second stage:

$$\mathbf{x}^{\text{trg}}_i = \text{LSTM}\left(\mathbf{c}^{\text{src}}_i, \mathbf{x}^{\text{trg}}_{i-1}; \theta^{\text{src}}_{\text{lstm}}\right) \quad (14)$$

$$\mathbf{s}^{\text{trg}}_j = \text{LSTM}\left(\left[W^{\text{trg}}_{\text{e}} y^{\text{trg}}_{i-1}; \mathbf{c}^{\text{trg}}_{j-1}\right], \mathbf{s}^{\text{trg}}_{j-1}; \theta^{\text{trg}}_{\text{lstm}}\right) \quad (15)$$

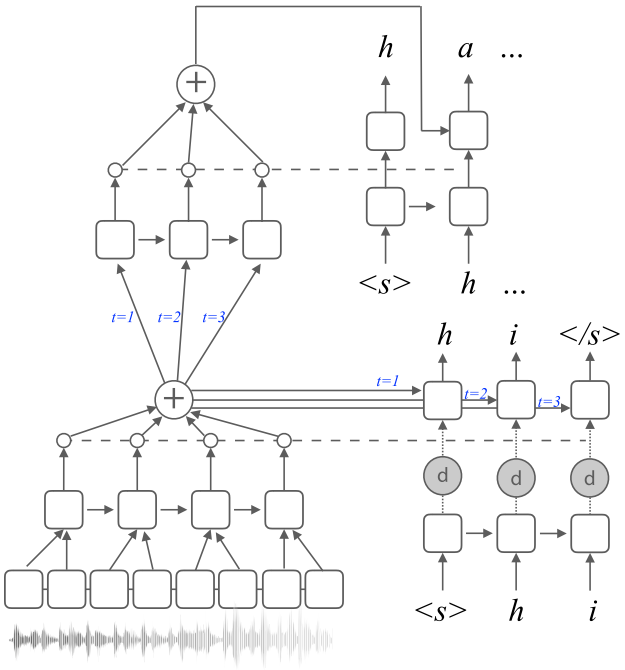

Figure 4: Attention-passing model.

$$\mathbf{c}_j^{\text{trg}} = \text{Attention}\left(\mathbf{s}_j^{\text{trg}}, \mathbf{x}_{1:N}^{\text{trg}}; \theta_{\text{att}}^{\text{trg}}\right) \quad (16)$$

$$\tilde{\mathbf{s}}_j^{\text{trg}} = \tanh\left(W_{\text{s}}^{\text{trg}}\left[\mathbf{s}_j^{\text{trg}}; \mathbf{c}_j^{\text{src}}\right] + \mathbf{b}_{\text{s}}^{\text{trg}}\right) \quad (17)$$

$$p\left(y_j^{\text{trg}} \mid y_{<j}, \mathbf{s}_{1:N}^{\text{src}}\right) = \text{SoftmaxOut}\left(\tilde{\mathbf{s}}_j^{\text{trg}}; \theta_{\text{out}}^{\text{trg}}\right) \quad (18)$$

4.2 Decoder State Drop-Out

Along with the modifications described in §4.1, we introduce an additional block drop-out operation (Ammar et al., 2016) on the decoder states, replacing equation 7 by

$$\tilde{\mathbf{s}}_i^{\text{src}} = \tanh\left(W_{\text{s}}^{\text{src}}\left[\text{BDrop}\left\{\mathbf{s}_i^{\text{src}}\right\}; \mathbf{c}_i^{\text{src}}\right] + \mathbf{b}_{\text{s}}^{\text{src}}\right).$$

The block drop-out operation, denoted as BDrop, replaces the whole vector by zero with a certain probability (here: 0.5). This results in the context vectors $\mathbf{c}_i^{\text{src}}$ becoming the only information available to the output layer whenever the decoder states are dropped out. The motivation for this is to force the model to maximize the informativeness of the context vectors, which are later relied upon as sole inputs to the second stage.

4.3 Multi-Task Training

Similar to the basic two-stage model, the attention-passing model as a whole is trained on speech-transcript-translation triplets, but can be decomposed into two sub-models that correspond to ASR and MT tasks. In fact, the ASR task is unchanged with the exception of the new block dropout operation. The MT task is obtained by replacing equation 14 by $\mathbf{x}_i^{\text{trg}} = \text{LSTM}\left(W_e \text{y}_i^{\text{src}}, \mathbf{x}_{i-1}^{\text{trg}}; \theta_{\text{lstm}}^{\text{src}}\right)$—that is, by using the transcript character embeddings as inputs instead of the context vectors used when training the main task. Note that the LSTMs in equations 5 and 14 are shared in order to have a match between stage 1 decoder and stage 2 encoder as with the basic model.

4.4 Cross Connections

As a further extension to the attention-passing model of §4.1, we can introduce cross connections that concatenate the dropped-out first stage hidden decoder states to the second-stage inputs encoder. This causes equation 14 to be replaced by

$$\mathbf{x}_i^{\text{trg}} = \quad (19)$$
$$\text{LSTM}\left(\text{Affine}\left[\mathbf{c}_i^{\text{src}}; \text{BDrop}\left\{\mathbf{s}_i^{\text{src}}\right\}\right], \mathbf{x}_{i-1}^{\text{trg}}; \theta_{\text{lstm}}^{\text{src}}\right)$$

This extension moves the model closer to the basic two-stage model, and the inclusion of the context vectors and the block drop-out operation on the hidden decoder states ensures that the second stage decoder does not rely too strongly on the first stage outputs.

4.5 Additional Loss

We further experiment with introducing an additional loss aimed at making the LSTM inputs between first stage decoder and second stage encoder RNN more similarly. Recall that in our attention-passing model, both RNNs share parameters (equations 5 and 14), so that similar inputs at both times is desirable. The loss is defined as follows:

$$\mathcal{L}_{\text{add}} = ||\mathbf{c}_i^{\text{src}} - W_e \text{y}_i^{\text{src}}||_2.$$

If combined with the cross connections (§4.4), the formula is adjusted to $\mathcal{L}_{\text{add}} = ||\text{Affine}\left[\mathbf{c}_i^{\text{src}}; \text{BDrop}\left\{\mathbf{s}_i^{\text{src}}\right\}\right] - W_e \text{y}_i^{\text{src}}||_2$. We did not find it beneficial to apply a scaling factor when adding this loss to the main cross-entropy loss in our experiments.

5 Experiments

We conduct experiments on the Fisher and Callhome Spanish–English Speech Translation

Corpus (Post et al., 2013), a corpus of Spanish telephone conversations that includes audio, transcriptions, and translations into English. We use the Fisher portion that consists of telephone conversations between strangers. The training data size is 138,819 sentences, corresponding to 162 hours of speech. ASR word error rates on this dataset are usually relatively high because of the spontaneous speaking style and challenging acoustics. From a translation viewpoint, the data can be considered as relatively easy with regard to both the topical domain and particular language pair.

Our implementation is based on the `xnmt` toolkit.[5] We use the speech recognition recipe as a starting point, which has previously been shown to achieve competitive ASR results (Neubig et al., 2018). [6]

The vocabulary consists of the common characters appearing in English and Spanish, apostrophe, whitespace, and special start-of-sequence and unknown-character tokens. The same vocabulary is used on both encoder (for the MT auxiliary task) and decoder sides. We set the batch size dynamically depending on the input sequence size such that the average batch size is 24 sentences. We use Adam (Kingma and Ba, 2014) with initial learning rate of 0.0005, decayed by 0.5 when the validation BLEU score did not improve over 10 check points initially and 5 check points after the first decay. We initialize attention-passing models using weights from a basic two-stage model trained on the same data.

Following Weiss et al. (2017), we lowercase texts and remove punctuation. As speech features, we use 40-dimensional Mel filter bank features with per-speaker mean and variance normalization. We exclude a small number of utterances longer than 1500 frames from training to avoid running out of memory. The encoder-decoder attention is MLP-based, and the decoder uses a single LSTM layer.[7] Source text encoders for the multi-task direct model and the cascaded models use two LSTM layers. The number of hidden units is 128 for the encoder-decoder attention MLP, 64 for target character embeddings, 256 for the encoder LSTMs in each direction, and 512 elsewhere. The model uses variational recurrent dropout with probability 0.3 and target character dropout with probability 0.1 (Gal and Ghahramani, 2016). We apply label smoothing (Szegedy et al., 2016) and fix the target embedding norm to 1 (Nguyen and Chiang, 2018). We use beam search with beam size 15 and polynomial length normalization with exponent 1.5.[8]

All BLEU scores are computed on Fisher/Test against 4 references.

Training sents.	Cascade	Direct model
139k	32.45	**35.30**
69k	**26.52**	24.68
35k	**16.84**	14.91
14k	**6.59**	6.08

Table 1: BLEU scores (4 references) on the Fisher/ Test for various amounts of training data. The direct (multi-task) model performs best in the full data condition, but the cascaded model is best in all reduced conditions.

5.1 Cascaded vs. Direct Models

We first wish to shed light on the question of whether cascaded or direct models can be expected to perform better. This question has been investigated previously (Weiss et al., 2017; Kano et al., 2017; Bérard et al., 2018; Anastasopoulos and Chiang, 2018), but with contradictory findings. We hypothesize that the increased complexity of the direct mapping from speech to translation increases the data requirements of such models. Table 1 compares the direct multi-task model (§3.1) against a cascaded model with identical architecture to the respective ASR and MT submodels of the multi-task model. The direct model is trained with multi-task training on the auxiliary ASR, MT, and AE tasks on the same data that outperformed single-task training considerably in preliminary experiments. As can be seen, the direct model outperforms the traditional cascaded setup only when both are trained on the full data, but not when using only parts of the training data. This

[5] `https://github.com/neulab/xnmt`.

[6] Code and configuration files can be found at `http://www.msperber.com/research/tacl-attention-passing/`.

[7] Weiss et al. (2017) report improvements from deeper decoders, but we encountered stability issues and therefore restricted the decoder to a single layer.

[8] For two-stage and attention-passing models, we apply beam search only for the second stage decoder. We do not use the two-phase beam search of Tu et al. (2017) because of its prohibitive memory requirements.

Model	BLEU
Cascade	32.45
Direct	35.30
Basic two-stage	34.36
APM	35.31
APM + cross connections	36.51
APM + cross conn. + additional loss	**36.70**
Best APM w/o block dropout	36.04

Table 2: Results for cascaded and multi-task models under full training data conditions.

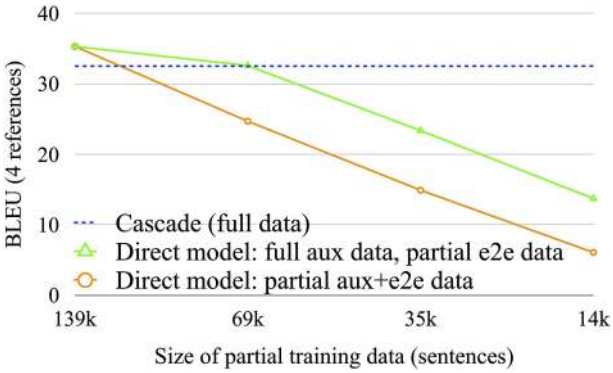

Figure 5: Data efficiency for the direct (multi-task) model, compared against cascade on full auxiliary data.

provides evidence in favor of our hypothesis and indicates that direct end-to-end models should be expected to perform strongly only in a case where enough training data is available.

5.2 Two-Stage Models

Next, we investigate the performance of the two-stage models, for both the basic variant (§3.2) and our proposed attention-passing model (§4). Again, all models are trained in a multi-task fashion by including auxiliary ASR and MT tasks based on the same data. Table 2 shows the results. The basic two-stage model performs in between the direct and cascaded models from §5.1. APM, the attention-passing model of §4.1 which is designed to circumvent the negative effects of error propagation, outperforms the basic variant and performs similarly to the direct model. The APM extensions (§4.4, §4.5) further improved the results, with the best model outperforming the direct model by 1.40 BLEU points and the basic two-stage model by 2.34 BLEU points absolute. The last row in the table confirms that the block dropout operation contributed to the gains: Removing it led to a drop by 0.66 BLEU points.

5.3 Data Efficiency: Direct Model

Having established results in favor of our proposed model on the full data, we now examine the data efficiency of the different models. Our experimental strategy is to compare model performance (1) when trained on the full data, (2) when trained on partial data, and (3) when trained on partial speech-to-translation data but full auxiliary (ASR+MT) data.[9]

Figure 5 shows the results, comparing the cascaded model against the direct model trained under conditions (1), (2), and (3).[10] Unsurprisingly, the performance of the direct model trained on partial data declines sharply as the amount of data is reduced. Adding auxiliary data through multi-task training improves performance in all cases. For instance, in the case of 69k speech-to-translation instances, adding the full auxiliary data helps to reach the accuracy of the cascaded model. However, this is already somewhat disappointing because the end-to-end data, which is not available to the cascaded model, no longer yields an advantage. Moreover, reducing the end-to-end data further reveals that multi-task training is not able to close the gap to the cascade. In the scenario with 35k end-to-end instances and full auxiliary data, the direct model underperforms the cascade by 9.14 BLEU points (32.50 vs. 23.36), despite being trained on *more* data. The unsatisfactory data efficiency in this controlled ablation study strongly indicates that the direct model will also fall behind a cascade that is trained on large amounts of external data. This claim is verified in §5.5.

5.4 Data Efficiency: Two-Stage Models

We showed that the direct model is poor at integrating auxiliary data and heavily depends on sufficient amounts of end-to-end training data.

[9] An alternative experimental strategy is to train on the full data and then add auxiliary data from other domains to the training. We pursue this strategy in §5.5 as a more realistic scenario, but point out several problems that lead us to not use this as our main approach: Adding external auxiliary data (1) leads to side-effects through domain mismatch and (2) severely limits the number of experiments that we can conduct because of the considerably increased training time.

[10] Note that the above hyper-parameters were selected for best full-data performance and are not re-tuned here.

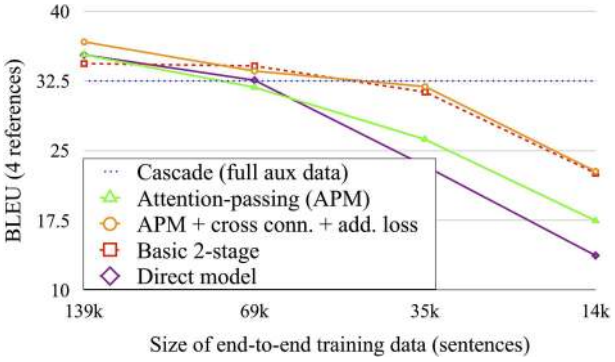

Figure 6: Data efficiency across model types. All models use full auxiliary data through multi-task training.

How do two-stage models behave with regard to this data efficiency issue? Figure 6 shows that both the basic two-stage model and the best APM perform reasonably well even when having seen much less end-to-end data. We can explain this by noticing that these models can be naturally decomposed into an ASR sub-model and an MT sub-model, while the direct model needs to add auxiliary sub-models to support multi-task training. Interestingly, the attention-passing model without cross-connections does better than the direct model with regard to data efficiency, but falls behind the basic and best proposed two-stage models. This indicates that access to ASR labels in some form contributes to favorable data efficiency of speech translation models.

5.5 Adding External Data

Our approach for evaluating data efficiency so far has been to assume that end-to-end data are available for only a subset of the available auxiliary data. In practice, we can often train ASR and MT tasks on abundant external data. We therefore run experiments in which we use the full Fisher training data for all tasks as before, and add OpenSubtitle[11] data for the auxiliary MT task. We clean and normalize the Spanish–English OpenSubtitle 2018 data (Lison and Tiedemann, 2016) to be consistent with the employed Fisher training data by lowercasing and removing punctuation. We apply a basic length filter and obtain 61 million sentences. During training, we include the same number of sentences from in-domain and out-of-domain MT tasks in each minibatch in order to prevent degradation due to domain mismatch.

[11] http://www.opensubtitles.org/.

Model	Fisher	Fisher+OpenSub
Cascade	32.45	34.58 (+6.2% rel.)
Direct model	35.30	36.45 (+3.2% rel.)
Basic two-stage	34.36	36.91 (+6.9% rel.)
Best APM	36.70	38.81 (+5.4% rel.)

Table 3: Adding auxiliary OpenSubtitles MT data to the training. The two-stage models benefit much more strongly than the direct model, with our proposed model yielding the strongest overall results.

Our models converged before a full pass over the OpenSubtitle data, but needed between two and three times more steps than the in-domain model to converge.

Table 3 shows that all models were able to benefit from the added data. However, when examining the relative gains we can see that both the cascaded model and the models with two attention stages benefitted about twice as much from the external data as the direct model. In fact, the basic two-stage model now slightly surpasses the direct model, and the best APM is ahead of the basic two-stage model by almost the same absolute difference as before (2.36 BLEU points). The superior relative gains show that our findings from §5.3 and §5.4, namely, that two-stage models are much more efficient at exploiting auxiliary training data, generalizes to the setting in which large amounts of out-of-domain data are added to the MT task. Out-of-domain data are often much easier to obtain, and we can therefore conclude that the proposed approach is preferable in many practically relevant situations. Because these experiments are very expensive to conduct, we leave experiments with external ASR data for future work.

5.6 Error Propagation

To better understand the impact of error propagation, we analyze how improved or degraded ASR labels impact the translation results. This experiment is applicable to APM, the two-stage model and the cascade, but not to the direct model which does not compute intermediate ASR outputs. We analyze three different settings: using the standard decoded ASR labels, replacing these labels with the gold labels, or artificially degrading the decoded labels by randomly introducing 10% of substitution, insertion, and deletion noise (Sperber

Labels	Gold	Decod.	Perturbed
Cascade	58.15 (+44%)	32.45	25.67 (-26%)
B2S	56.60 (+39%)	34.36	28.81 (-19%)
APM	40.70 (+13%)	35.31	31.96 (-10%)
+ cross	58.29 (+37%)	36.70	30.48 (-20%)

Table 4: Effect of altering the ASR labels for different models as a measure for robustness against error propagation. We compare results for the cascade, the basic two-stage model (B2S), and APM without and with cross connections. Percentages are relative to the results for unaltered (decoded) ASR labels.

et al., 2017). Intuitively, models that suffer from error propagation issues are expected to rely most heavily on these intermediate labels and would therefore be most impacted by both degraded and improved labels.

Table 4 shows the results. Unsurprisingly, the cascade responds most strongly to improved or degraded noise, confirming that it is severely impacted by error propagation. The APM, which does not directly expose the labels to the translation sub-model, is much less impacted. However, the impact is still more significant than perhaps expected, suggesting that improved attention models that are more robust to decoding errors (Chorowski et al., 2015; Tjandra et al., 2017) may serve to further improve our model in the future. Note that the APM benefits poorly from gold ASR labels, which is expected because gold labels only improve the ASR alignments and by extension the passed context vectors, but these are quite robust against decoding errors in the first place.

The basic two-stage model is impacted significantly, although less strongly than the cascade, in line with our claim that such models are subject to error propagation despite being end-to-end trainable. Note that it falls behind the cascade for gold labels, despite both models being seemingly identical under this condition. This can be explained by the cascaded model's use of beam search and greater number of encoder layers.

Somewhat contrary to our expectations, APM with cross connections appears equally subject to error propagation despite the block dropout on these connections, displaying the same accuracy gains across the three different settings. This suggests future explorations toward model variants with an even better trade-off between overall accuracy, data efficiency, and amount of degradation due to error propagation.

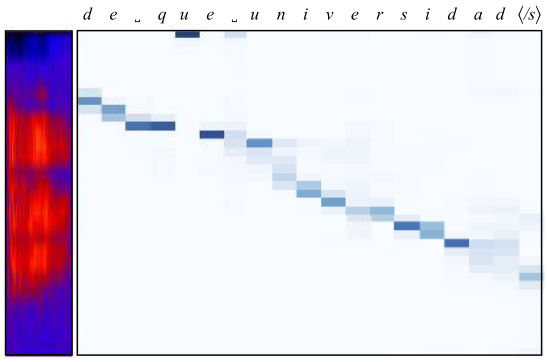

Figure 7: ASR attentions when force-decoding the oracle transcripts.

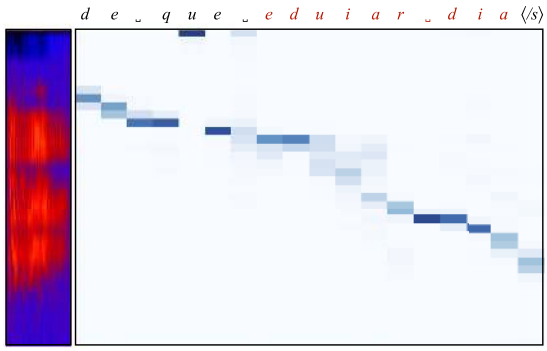

Figure 8: ASR attentions after regular decoding.

5.7 Robustness of ASR Attentions

The attention-passing model was motivated by the assumption that attention scores are relatively robust against recognition errors. We perform a qualitative analysis to validate this assumption. Figure 7 shows the first-stage attention matrix when force-decoding the reference transcript, and Figure 8 shows the same for regular decoding, which for this utterance produced significant errors. Despite the errors, we can see that the attention matrices are very similar. We manually inspected the first 100 test attention matrices and confirm that this behavior occurs very consistently. Further quantitative evidence is given in §5.6, which showed that the attention-passing model is more resistent to error propagation than the other models.

6 Prior Work

Model architectures similar to what we have referred to as the basic two-stage model have first been used by Tu et al. (2017) for a reconstruction

task, where the first stage performs translation and the second stage attempts to reconstruct the original inputs based on the outputs of the first stage. A second variant of a similar architecture are Xia et al. (2017)'s deliberation networks, where the second stage refines or polishes the outputs of the first stage. For our purposes, the first stage performs speech recognition, a natural intermediate representation for the speech translation task, corresponding to the second stage output. Toshniwal et al. (2017) explore a different way of lower-level supervision during training of an attentional speech recognizer by jointly training an auxiliary phoneme recognizer based on a lower layer in the acoustic encoder. Similarly to the discussed multi-task direct model, this approach discards many of the learned parameters when used on the main task and consequently may also suffer from data efficiency issues.

Direct end-to-end speech translation models were first used by Duong et al. (2016), although the authors did not actually evaluate translation performance. Weiss et al. (2017) extended this model into a multi-task model and report excellent translation results. Our baselines do not match their results, despite considerable efforts. We note that other research groups have encountered similar replicability issues (Bansal et al., 2018), explanations include the lack of a large GPU cluster to perform ASGD training, as well as to explore an ideal number of training schedules and other hyper-parameter settings. Bérard et al. (2018) explored the translation of audio books with direct models and report reasonable results, but do not outperform a cascaded baseline. Kano et al. (2017) have first used a basic two-stage model for speech translation. They use a pretraining strategy for the individual sub-models, related to our multi-task approach, but do not attempt to integrate auxiliary data. Moreover, the authors only evaluated the translation of synthesized speech, which greatly simplifies training and may not lead to generalizable conclusions, as indicated by the fact that they were actually able to outperform a translation model that used the gold transcripts as input. Anastasopoulos and Chiang (2018) conducted experiments on low-resource speech translation and used a triangle model that can be seen as a combination of a direct model and a two-stage model, but is not easily trainable in a multi-task fashion. It is therefore not a suitable choice for exploiting auxiliary data in order to compete with cascaded models under well-resourced data conditions. Finally, contemporaneous work explores transferring knowledge from high-resource to low-resource languages (Bansal et al., 2019).

7 Conclusion

This work explored *direct* and *two-stage* models for speech translation with the aim of obtaining models that are strong, not only in favorable data conditions, but are also efficient at exploiting auxiliary data. We started by demonstrating that direct models do outperform cascaded models, but only when enough data is available, shedding light on inconclusive results from prior work. We further showed that these models are poor at exploiting auxiliary data, making them a poor choice in realistic situations. We were motivated to use two-stage models by their ability to overcome this shortcoming of the direct models, and found that two-stage models are in fact more data-efficient, but suffer from error propagation issues. We addressed this by introducing a novel attention-passing model that alleviates error propagation issues, as well as several model variants. The best proposed model outperforms all other tested models and is much more data efficient than the direct model, allowing this model to compete with cascaded models even under realistic assumptions with auxiliary data available. Analysis showed that there seems to be a trade-off between data efficiency and error propagation. Avenues for future work include testing better ASR attention models; adding other types of external data such as ASR data, unlabeled speech, or monolingual texts; and exploring further model variants.

Acknowledgments

We thank Adam Lopez, Stefan Constantin, and the anonymous reviewers for their helpful comments. The work leading to these results has received funding from the European Union under grant agreement no. 825460.

References

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith.

2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics (TACL)*, 4:431–444.

Antonios Anastasopoulos and David Chiang. 2018. Tied multitask learning for neural speech translation. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, New Orleans, USA.

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Representation Learning (ICLR)*. San Diego, CA.

Sameer Bansal, Herman Kamper, Karen Livescu, Adam Lopez, and Sharon Goldwater. 2018. Low-resource speech-to-text translation. In *Annual Conference of the International Speech Communication Association (InterSpeech)*.

Sameer Bansal, Herman Kamper, Karen Livescu, Adam Lopez, and Sharon Goldwater. 2019. Pre-training on high-resource speech recognition improves low-resource speech-to-text translation. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Alexandre Bérard, Laurent Besacier, Ali Can Kocabiyikoglu, and Olivier Pietquin. 2018. End-to-end automatic speech translation of audiobooks. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary.

William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP)*.

Jan K. Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Advances in Neural Information Processing Systems (NIPS)*, pages 577–585.

Christopher Cieri, David Miller, and Kevin Walker. 2004. The Fisher corpus: A resource for the next generations of speech-to-text. In *Language Resources and Evaluation (LREC)*, pages 69–71.

Long Duong, Antonios Anastasopoulos, David Chiang, Steven Bird, and Trevor Cohn. 2016. An attentional model for speech translation without transcription. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 949–959, San Diego, CA.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Neural Information Processing Systems Conference (NIPS)*, pages 1019–1027, Barcelona.

Takatomo Kano, Sakriani Sakti, and Satoshi Nakamura. 2017. Structured-based curriculum Learning for end-to-end English-Japanese Speech translation. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, pages 2630–2634.

Diederik P. Kingma and Jimmy L. Ba. 2014. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, Banff.

Ali Can Kocabiyikoglu, Laurent Besacier, and Olivier Kraif. 2018. Augmenting Librispeech with French translations: A multimodal corpus for direct speech translation evaluation. In *Language Resources and Evaluation (LREC)*, Miyazaki, Japan.

Pierre Lison and Jörg Tiedemann. 2016. OpenSubtitles2016: Extracting large parallel corpora from movie and TV subtitles. In *Conference on Language Resources and Evaluation (LREC)*, pages 923–929. Portorož.

Graham Neubig, Matthias Sperber, Xinyi Wang, Matthieu Felix, Austin Matthews, Sarguna Padmanabhan, Ye Qi, Devendra Singh Sachan, Philip Arthur, Pierre Godard, John Hewitt, Rachid Riad, and Liming Wang. 2018. XNMT: The eXtensible Neural Machine Translation toolkit. In *Conference of the Association for Machine Translation in the Americas (AMTA) Open Source Software Showcase*, Boston, MA.

Toan Q. Nguyen and David Chiang. 2018. Improving lexical choice in neural machine translation. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, New Orleans, LA.

Jan Niehues and Eunah Cho. 2017. Exploiting linguistic resources for neural machine translation using multi-task learning. In *Conference on Machine Translation (WMT)*, pages 80–89. Copenhagen.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: An ASR corpus based on public domain audio books. In *Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. Brisbane.

Matt Post, Gaurav Kumar, Adam Lopez, Damianos Karakos, Chris Callison-Burch, and Sanjeev Khudanpur. 2013. Improved speech-to-text translation with the Fisher and Callhome Spanish–English speech translation corpus. In *International Workshop on Spoken Language Translation (IWSLT)*, Heidelberg.

Matthias Sperber, Jan Niehues, and Alex Waibel. 2017. Toward robust neural machine translation for noisy input sequences. In *International Workshop on Spoken Language Translation (IWSLT)*, Tokyo.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, Las Vegas, NV.

Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura. 2017. Local monotonic attention mechanism for end-to-end speech recognition. In *International Joint Conference on Natural Language Processing (IJCNLP)*, pages 431–440.

Shubham Toshniwal, Hao Tang, Liang Lu, and Karen Livescu. 2017. Multitask learning with low-level auxiliary tasks for encoder-decoder based speech recognition. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, Stockholm.

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2017. Neural machine translation with reconstruction. In *Conference on Artificial Intelligence (AAAI)*.

Ron J. Weiss, Jan Chorowski, Navdeep Jaitly, Yonghui Wu, and Zhifeng Chen. 2017. Sequence-to-sequence models can directly transcribe foreign speech. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, Stockholm.

Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2017. Deliberation networks: sequence generation. In *Neural Information Processing Systems Conference (NIPS)*, Long Beach, CA.

Yu Zhang, William Chan, and Navdeep Jaitly. 2017. Very deep convolutional Networks for end-to-end speech recognition. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.