Dual-decoder Transformer for Joint Automatic Speech Recognition and Multilingual Speech Translation

Hang Le[1] Juan Pino[2] Changhan Wang[2]
Jiatao Gu[2] Didier Schwab[1] Laurent Besacier[1]
[1]Univ. Grenoble Alpes, CNRS, LIG [2]Facebook AI
{hang.le, didier.schwab, laurent.besacier}@univ-grenoble-alpes.fr
{juancarabina, changhan, jgu}@fb.com

Abstract

We introduce dual-decoder Transformer, a new model architecture that jointly performs automatic speech recognition (ASR) and multilingual speech translation (ST). Our models are based on the original Transformer architecture (Vaswani et al., 2017) but consist of two decoders, each responsible for one task (ASR or ST). Our major contribution lies in how these decoders interact with each other: one decoder can attend to different information sources from the other via a *dual-attention* mechanism. We propose two variants of these architectures corresponding to two different levels of dependencies between the decoders, called the *parallel* and *cross* dual-decoder Transformers, respectively. Extensive experiments on the MuST-C dataset show that our models outperform the previously-reported highest translation performance in the multilingual settings, and outperform as well bilingual one-to-one results. Furthermore, our parallel models demonstrate no trade-off between ASR and ST compared to the vanilla multi-task architecture. Our code and pre-trained models are available at `https://github.com/formiel/speech-translation`.

1 Introduction

While cascade speech-to-text translation (ST) systems operate in two steps: source language automatic speech recognition (ASR) and source-to-target text machine translation (MT), recent works have attempted to build end-to-end ST without using source language transcription during decoding (Bérard et al., 2016; Weiss et al., 2017; Bérard et al., 2018). After two years of extensions to these pioneering works, the last results of the IWSLT 2020 shared task on offline speech translation (Ansari et al., 2020) demonstrate that end-to-end models are now on par (if not better) than their cascade counterparts. Such a finding motivates even more strongly the works on multilingual (one-to-many, many-to-one, many-to-many) ST (Gangi et al., 2019; Inaguma et al., 2019; Wang et al., 2020a) for which end-to-end models are well adapted by design. Moreover, of these two approaches: *cascade* proposes a very loose integration of ASR and MT (even if lattices or word confusion networks were used between ASR and MT before end-to-end models appeared) while most *end-to-end* approaches simply ignore ASR subtask, trying to directly translate from source speech to target text. We believe that these are two edge design choices and that a tighter coupling of ASR and MT is desirable for future end-to-end ST applications, in which the display of transcripts alongside translations can be beneficial to the users (Sperber et al., 2020).

This paper addresses multilingual ST and investigates more closely the interactions between speech transcription (ASR) and speech translation (ST) in a multilingual end-to-end architecture based on Transformer. While those interactions were previously investigated as a simple multi-task framework for a bilingual case (Anastasopoulos and Chiang, 2018), we propose a dual-decoder with an ASR decoder tightly coupled with an ST decoder and evaluate its effectiveness on one-to-many ST. Our model is inspired by Liu et al. (2020), but the interaction between ASR and ST decoders is much tighter.[1] Finally, experiments show that our model outperforms theirs on the MuST-C benchmark (Di Gangi et al., 2019).

Our contributions are summarized as follows: (1) a new model architecture for joint ASR and multilingual ST; (2) an integrated beam search decoding strategy which jointly transcribes and translates,

[1] The model of Liu et al. (2020) does not have interaction between internal hidden states of the decoders (*c.f.* Section 3.4).

Proceedings of the 28th International Conference on Computational Linguistics, pages 3520–3533
Barcelona, Spain (Online), December 8-13, 2020

and that is extended to a wait-k strategy where the ASR hypothesis is ahead of the ST hypothesis by k tokens and vice-versa; and (3) competitive performance on the MuST-C dataset in both bilingual and multilingual settings and improvements on previous joint ASR/ST work.

2 Related Work

Multilingual ST Multilingual translation (Johnson et al., 2016) consists in translating between different language pairs with a single model, thereby improving maintainability and the quality of low resource languages. Gangi et al. (2019) adapt this method to one-to-many multilingual speech translation by adding a language embedding to each source feature vector. They also observe that using the source language (English) as one of the target languages improves performance. Inaguma et al. (2019) simplify the previous approach by prepending a target language token to the decoder and apply it to one-to-many and many-to-many speech translation. They do not investigate many-to-one due to the lack of a large corpus for this. To fill this void, Wang et al. (2020a) release the CoVoST dataset for ST from 11 languages into English and demonstrate the effectiveness of many-to-one ST.

Joint ASR and ST Joint ASR and ST decoding was first proposed by Anastasopoulos and Chiang (2018) through a multi-task learning framework. Chuang et al. (2020) improve multitask ST by using word embedding as an intermediate level instead of text. A two-stage model that performs first ASR and then passes the decoder states as input to a second ST model was also studied previously (Anastasopoulos and Chiang, 2018; Sperber et al., 2019). This architecture is closer to cascaded translation while maintaining end-to-end trainability. Sperber et al. (2020) introduce the notion of consistency between transcripts and translations and propose metrics to gauge it. They evaluate different model types for the joint ASR and ST task and conclude that end-to-end models with coupled inference procedure are able to achieve strong consistency. In addition to existing models having coupled architectures, they also investigate a model where the transcripts are concatenated to the translations, and the shared encoder-decoder network learns to predict this concatenated outputs. It should be noted that our models have lower latency compared to this approach since the concatenation of outputs makes the two tasks sequential in nature. Our work is closely related to that of Liu et al. (2020) who propose an *interactive attention* mechanism which enables ASR and ST to be performed synchronously. Both ASR and ST decoders do not only rely on their previous outputs but also on the outputs predicted in the other task. We highlight three differences between their work and ours: (a) we propose a more general framework in which (Liu et al., 2020) is a special case; (b) tighter integration of ASR and ST is proposed in our work; and (c) we experiment in a multilingual ST setting while previous works on joint ASR and ST only investigated bilingual ST.

3 Dual-decoder Transformer for Joint ASR and Multilingual ST

We now present the proposed dual-decoder Transformer for jointly performing ASR and multilingual ST. Our models are based on the Transformer architecture (Vaswani et al., 2017) but consist of two decoders. Each decoder is responsible for one task (ASR or ST). The intuition is that the problem at hand consists in solving two different tasks with different characteristics and different levels of difficulty (multilingual ST is considered more difficult than ASR). Having different decoders specialized in different tasks may thus produce better results. In addition, since these two tasks can be complementary, it is natural to allow the decoders to help each other. Therefore, in our models, we introduce a *dual-attention* mechanism: in addition to attending to the encoder, the decoders also attend to each other.

3.1 Model overview

The model takes as input a sequence of speech features $\mathbf{x} = (x_1, x_2, \dots, x_{T_x})$ in a specific *source* language (*e.g.* English) and outputs a transcription $\mathbf{y} = (y_0, y_1, \dots, y_{T_y})$ in the same language as well as translations $\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_M$ in M different *target* languages (*e.g.* French, Spanish, *etc.*). When $M = 1$, this corresponds to joint ASR and bilingual ST (Liu et al., 2020). For simplicity, our presentation considers only a single target language with output $\mathbf{z} = (z_0, z_1, \dots, z_{T_z})$. All results, however, apply to the general multilingual case. In the sequel, denote $\mathbf{y}_{<t} \triangleq (y_0, y_1, \dots, y_{t-1})$ and $\mathbf{y}_{>t} \triangleq (y_{t+1}, y_{t+2}, \dots, y_{T_y})$

(y_t is included if "$<$" and "$>$" are replaced by "$\leq$" and "$\geq$" respectively). In addition, assume that y_t is ignored if t is outside of the interval $[0, T_y]$. Notations apply to $\mathbf{z}$ as well.

The dual-decoder model jointly predicts the transcript and translation in an autoregressive fashion:

$$p(\mathbf{y}, \mathbf{z} \mid \mathbf{x}) = \prod_{t=0}^{\max(T_y, T_z)} p(y_t, z_t \mid \mathbf{y}_{<t}, \mathbf{z}_{<t}, \mathbf{x}). \tag{1}$$

A natural model would consist of a single decoder followed by a softmax layer. However, even if the capacity of the decoder were large enough for handling both ASR and ST generation, a single softmax would require a very large joint vocabulary (with size $V_y V_z$ where V_y and V_z are respectively the vocabulary sizes for $\mathbf{y}$ and $\mathbf{z}$). Instead, our dual-decoder consists of two sub-decoders that are specialized in producing outputs tailored to the ASR and ST tasks separately. Formally, our model predicts the next output tokens $(\hat{y}_s, \hat{z}_t)$ (where $1 \leq s \leq T_y, 1 \leq t \leq T_z$) given a pair of previous outputs $(\mathbf{y}_{<s}, \mathbf{z}_{<t})$ as:

$$\mathbf{h}_s^y, \mathbf{h}_t^z = \text{DECODER}_{\text{dual}}(\mathbf{y}_{<s}, \mathbf{z}_{<t}, \text{ENCODER}(\mathbf{x})) \in \mathbb{R}^{d_y} \times \mathbb{R}^{d_z}, \tag{2}$$

$$p(y_s \mid \mathbf{y}_{<s}, \mathbf{z}_{<t}, \mathbf{x}) = [\text{softmax}(\mathbf{W}^y \mathbf{h}_s^y + \mathbf{b}^y)]_{y_s}, \qquad \hat{y}_s = \text{argmax}_{y_s}\, p(y_s \mid \mathbf{y}_{<s}, \mathbf{z}_{<t}, \mathbf{x}), \tag{3}$$

$$p(z_t \mid \mathbf{y}_{<s}, \mathbf{z}_{<t}, \mathbf{x}) = [\text{softmax}(\mathbf{W}^z \mathbf{h}_t^z + \mathbf{b}^z)]_{z_t}, \qquad \hat{z}_t = \text{argmax}_{z_t}\, p(z_t \mid \mathbf{y}_{<s}, \mathbf{z}_{<t}, \mathbf{x}), \tag{4}$$

where $[\mathbf{v}]_i$ denotes the i^{th} element of the vector $\mathbf{v}$. Note that y_s and z_t are token indices ($1 \leq y_s \leq V_y, 1 \leq z_t \leq V_z$). In (3) and (4), we detail the intermediate quantities $p(y_s \mid \cdot)$ and $p(z_t \mid \cdot)$ as obtained from the probability distributions over the output vocabulary. In the above, we have made an important assumption about the joint probability $p(y_s, z_t \mid \cdot)$ that it can be factorized into $p(y_s \mid \cdot)p(z_t \mid \cdot)$. Therefore, the joint distribution (1) encoded by the dual-decoder Transformer can be rewritten as

$$p(\mathbf{y}, \mathbf{z} \mid \mathbf{x}) = \prod_{t=0}^{\max(T_y, T_z)} p(y_t \mid \mathbf{y}_{<t}, \mathbf{z}_{<t}, \mathbf{x}) p(z_t \mid \mathbf{y}_{<t}, \mathbf{z}_{<t}, \mathbf{x}). \tag{5}$$

We also assumed so far that the sub-decoders start at the same time, which is the most basic configuration. In practice, however, one may allow one sequence to advance k steps compared to the other, known as the *wait-k* policy (Ma et al., 2019). For example, if ST waits for ASR to produce its first k tokens, then the joint distribution becomes

$$p(\mathbf{y}, \mathbf{z} \mid \mathbf{x}) = p(\mathbf{y}_{<k} \mid \mathbf{x}) p(\mathbf{y}_{\geq k}, \mathbf{z} \mid \mathbf{y}_{<k}, \mathbf{x}) \tag{6}$$

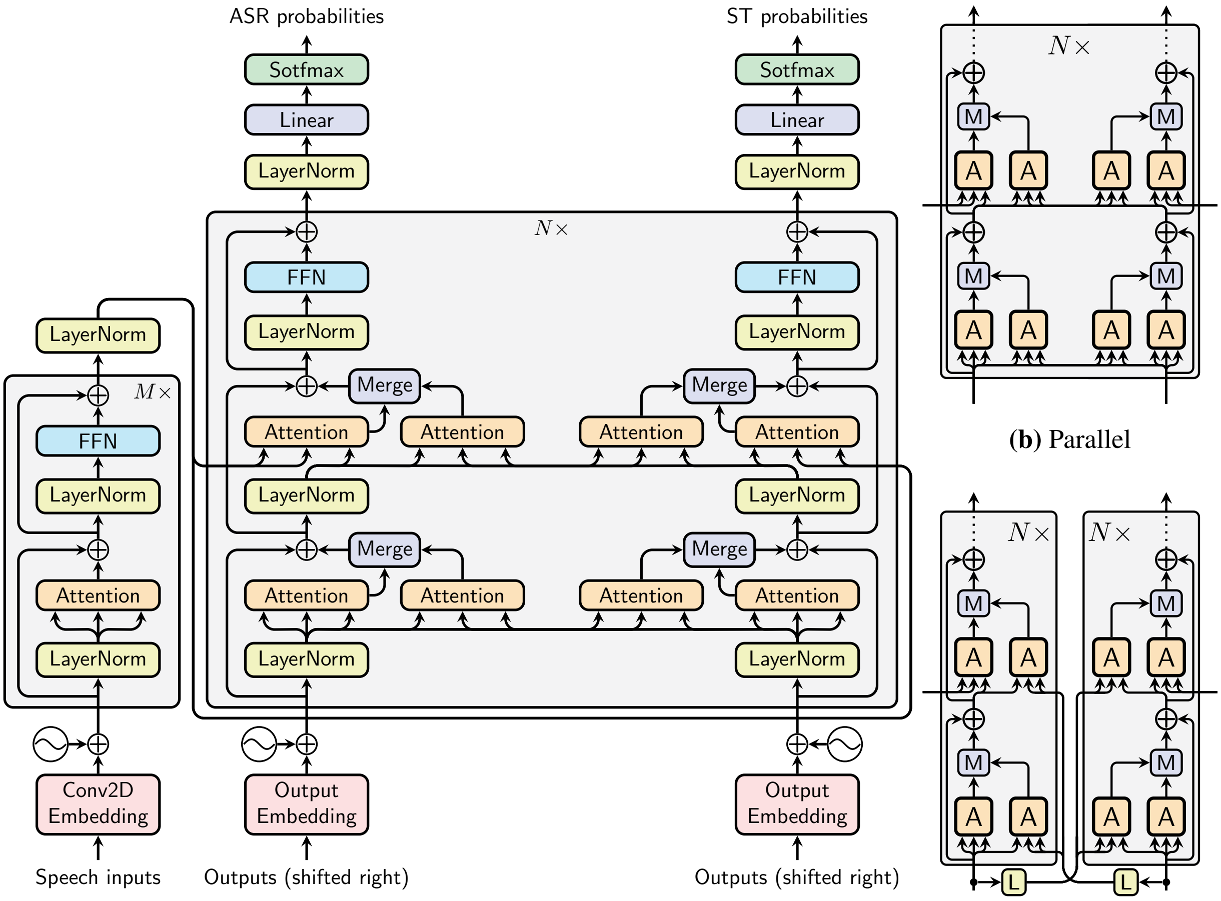
$$= \prod_{t=0}^{k-1} p(y_t \mid y_{<t}, \mathbf{x}) \prod_{t=0}^{\max(T_y - k, T_z)} p(y_{t+k} \mid \mathbf{y}_{<t+k}, \mathbf{z}_{<t}, \mathbf{x}) p(z_t \mid \mathbf{y}_{<t+k}, \mathbf{z}_{<t}, \mathbf{x}). \tag{7}$$

In the next section, we propose two concrete architectures for the dual-decoder, corresponding to different levels of dependencies between the two sub-decoders (ASR and ST). Then, we show that several known models in the literature are special cases of these architectures.

3.2 Parallel and cross dual-decoder Transformers

The first architecture is called *parallel dual-decoder Transformer*, which has the highest level of dependencies: one decoder uses the hidden states of the other to compute its outputs, as illustrated in Figure 1a. The encoder consists of an input embedding layer followed by a positional embedding and a number of *self-attention* and feed-forward network (FFN) layers whose inputs are normalized (Ba et al., 2016).[2] This is almost the same as the encoder of the original Transformer (Vaswani et al., 2017) (we refer to the corresponding paper for further details), except that the embedding layer in our encoder is a small convolutional neural network (CNN) (Fukushima and Miyake, 1982; LeCun et al., 1989) of two layers with ReLU activations and a stride of 2, therefore reducing the input length by 4.

[2] All the illustrations in this paper are for the so-called *pre-LayerNorm* configuration, in which the *input* of the layer is normalized. Likewise, if the *output* is normalized instead, the configuration is called *post-LayerNorm*. Since pre-LayerNorm is known to perform better than post-LayerNorm (Wang et al., 2019; Nguyen and Salazar, 2019), we only conducted experiments for the former, although our implementation supports both.



(a) The parallel dual-decoder Transformer

(b) Parallel

(c) Cross

Figure 1: The dual-decoder Transformers. Figure (a) shows the detailed architecture of the *parallel* dual-decoder Transformer, and Figure (b) shows its simplified view. The *cross* dual-decoder Transformer is very similar to the parallel one, except that the keys and values fed to the dual-attention layers come from the previous output, which is illustrated by Figure (c). From the above figures, one can easily infer the detailed architecture of the cross Transformer, which can be found in the Appendix. Abbreviations: **A** (Attention), **M** (Merge), **L** (LayerNorm).

The parallel dual-decoder consists of: (a) two decoders that follow closely the common Transformer decoder structure, and (b) four additional multi-head attention layers (called *dual-attention* layers). Each dual-attention layer is complementary to a corresponding main attention layer. We recall that an attention layer receives as inputs a query $\mathbf{Q} \in \mathbb{R}^{d_k}$, a key $\mathbf{K} \in \mathbb{R}^{d_k}$, a value $\mathbf{V} \in \mathbb{R}^{d_v}$ and outputs $\mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$.[3] A dual-attention layer receives $\mathbf{Q}$ from the main branch and $\mathbf{K}, \mathbf{V}$ from the other decoder at the same level (*i.e.* at the same depth in Transformer architecture) to compute hidden representations that will be merged back into the main branch (*i.e.* one decoder attends to the other in parallel). We present in more detail this merging operation in Section 3.4.

Our second proposed architecture is called *cross dual-decoder Transformer*, which is similar to the previous one, except that now the dual-attention layers receive $\mathbf{K}, \mathbf{V}$ from the previous decoding step outputs of the other decoder, as illustrated in Figure 1c. Thanks to this design, each prediction step can be performed separately on the two decoders. The hidden representations $\mathbf{h}_s^y, \mathbf{h}_t^z$ in (2) produced by the decoders can thus be decomposed into:[4]

$$\mathbf{h}_s^y = \textsc{Decoder}_{\text{asr}}(\mathbf{y}_{<s}, \mathbf{z}_{<t}, \textsc{Encoder}(\mathbf{x})) \in \mathbb{R}^{d_y}, \tag{8}$$

$$\mathbf{h}_t^z = \textsc{Decoder}_{\text{st}}(\mathbf{z}_{<t}, \mathbf{y}_{<s}, \textsc{Encoder}(\mathbf{x})) \in \mathbb{R}^{d_z}. \tag{9}$$

[3] Following Vaswani et al. (2017), we use *multi-head attention*, in which the inputs are linearly projected multiple times before feeding to the function, then the outputs are concatenated and projected back to the original dimension.

[4] This decomposition is clearly not possible for the parallel dual-decoder Transformer.

3.3 Special cases

In this section, we present some special cases of our dual-decoder architecture and discuss their links to existing models in the literature.

Independent decoders When there is no dual-attention, the two decoders become *independent*. In this case, the prediction joint probability can be factorized simply as $p(y_s, z_t \mid \mathbf{y}_{<s}, \mathbf{z}_{<t}, \mathbf{x}) = p(y_s \mid \mathbf{y}_{<s}, \mathbf{x})p(z_t \mid \mathbf{z}_{<t}, \mathbf{x})$. Therefore, *all* prediction steps are separable and thus this model is the most computationally efficient. In the literature, this model is often referred to as *multi-task* (Anastasopoulos and Chiang, 2018; Sperber et al., 2020).

Chained decoders Another special case corresponds to the extreme wait-k policy, in which one decoder waits for the other to completely finish before starting its own decoding. For example, if ST waits for ASR, then the prediction joint probability reads $p(y_s, z_t \mid \mathbf{y}_{<s}, \mathbf{z}_{<t}, \mathbf{x}) = p(y_s \mid \mathbf{y}_{<s}, \mathbf{x})p(z_t \mid \mathbf{z}_{<t}, \mathbf{y}, \mathbf{x})$. This model is called *triangle* in previous work (Anastasopoulos and Chiang, 2018; Sperber et al., 2020). A special case of this model is when the second decoder in the chain is not directly connected to the encoder, also referred to as *two-stage*[5] (Sperber et al., 2019; Sperber et al., 2020).

To summarize the different cases, we show below the joint probability distributions encoded by the presented models, in decreasing level of dependencies:

$$\text{(single output)} \qquad p(\mathbf{y}, \mathbf{z} \mid \mathbf{x}) = \prod_{t=0}^{T} p(y_t, z_t \mid \mathbf{y}_{<t}, \mathbf{z}_{<t}, \mathbf{x}), \tag{10}$$

$$\text{(dual-decoder)} \qquad p(\mathbf{y}, \mathbf{z} \mid \mathbf{x}) = \prod_{t=0}^{T} p(y_t \mid \mathbf{y}_{<t}, \mathbf{z}_{<t}, \mathbf{x})p(z_t \mid \mathbf{y}_{<t}, \mathbf{z}_{<t}, \mathbf{x}), \tag{11}$$

$$\text{(chained)} \qquad p(\mathbf{y}, \mathbf{z} \mid \mathbf{x}) = \prod_{t=0}^{T} p(y_t \mid \mathbf{y}_{<t}, \mathbf{x})p(z_t \mid \mathbf{z}_{<t}, \mathbf{y}, \mathbf{x}), \tag{12}$$

$$\text{(independent)} \qquad p(\mathbf{y}, \mathbf{z} \mid \mathbf{x}) = \prod_{t=0}^{T} p(y_t \mid \mathbf{y}_{<t}, \mathbf{x})p(z_t \mid \mathbf{z}_{<t}, \mathbf{x}), \tag{13}$$

where $T = \max(T_y, T_z)$. Similar formalization for the wait-k policy (7) can be obtained in a straightforward manner. Note that for independent decoders, the distribution is the same as in non-wait-k.

3.4 Variants

The previous section presents special cases of our formulation at a high level. In this section, we introduce different fine-grained variants of the dual-decoder Transformers used in the experiments (Section 5).

Asymmetric dual-decoder Instead of using all the dual-attention layers, one may want to allow a one-way attention: either ASR attends ST or the inverse, but not both.

At-self or at-source dual-attention In each decoder block, there are two different attention layers, which we respectively call *self-attention* (bottom) and *source-attention*[6] (top). For each, there is an associated dual-attention, named respectively *dual-attention at self* and *dual-attention at source*. In the experiments, we study the case where either only the at-self or at-source attention layers are retained.

Merging operators The Merge layers shown in Figure 1 combine the outputs of the main attention $\mathbf{H}_{\text{main}}$ and the dual-attention $\mathbf{H}_{\text{dual}}$. We experimented dual-attention with two different merging operators: weighted sum or concatenation. We can formally define the merging operators as

$$\mathbf{H}_{\text{out}} = \text{Merge}(\mathbf{H}_{\text{main}}, \mathbf{H}_{\text{dual}}) \triangleq \begin{cases} \mathbf{H}_{\text{main}} & \text{if no dual-attention,} \\ \mathbf{H}_{\text{main}} + \lambda \mathbf{H}_{\text{dual}}, & \text{if } \texttt{sum} \text{ operator,} \\ \text{linear}\left([\mathbf{H}_{\text{main}}; \mathbf{H}_{\text{dual}}]\right) & \text{if } \texttt{concat} \text{ operator.} \end{cases} \tag{14}$$

[5] Also called *cascade* by Anastasopoulos and Chiang (2018). We omit this term to avoid confusion with the common cascade models that are typically not trained end-to-end. Note that our chained-decoder (both *triangle* and *two-stage*) are end-to-end.

[6] Also referred to as *cross-attention* in the literature. We use a different name to avoid confusion with the *cross* dual-decoder.

For the sum operator, in particular, we perform experiments for *learnable* or *fixed* λ.

Remark. The model proposed by Liu et al. (2020) is a special case of our *cross* dual-decoder Transformer with no dual-attention at source, no layer normalization for the input embeddings (Figure 1c), and sum merging with fixed λ.

4 Training and Decoding

4.1 Training

The objective, $L(\hat{\mathbf{y}}, \hat{\mathbf{z}}, \mathbf{y}, \mathbf{z}) = \alpha L_{\mathrm{asr}}(\hat{\mathbf{y}}, \mathbf{y}) + (1-\alpha) L_{\mathrm{st}}(\hat{\mathbf{z}}, \mathbf{z})$, is a weighted sum of the cross-entropy ASR and ST losses, where $(\hat{\mathbf{y}}, \hat{\mathbf{z}})$ and $(\mathbf{y}, \mathbf{z})$ denote the predictions and the ground-truths for (ASR, ST), respectively. The weight α is set to 0.3 in all experiments. Here we favor the ST task based on the intuition that it is more difficult to train than the ASR one, simply because of the multilinguality. A hyperparameter search may further improve the results. We also employ label smoothing (Szegedy et al., 2016) with $\epsilon = 0.1$. For each language pair, training data is sorted by the number of frames. Each mini-batch contains all languages such that their numbers of frames are roughly the same. We follow Inaguma et al. (2019) and prepend a language-specific token to the target sentence. Preliminary experiments showed that this approach was more effective than adding a target language embedding along the temporal dimension to the speech feature inputs (Di Gangi et al., 2019).

4.2 Decoding

We present the beam search strategy used by our model. Since there are two different outputs (ASR and ST), one may naturally think about two different beams (with possibly some interactions). However, we found that a *single joint beam* works best for our model. In this beam search strategy, each hypothesis includes a tuple of ASR and ST sub-hypotheses. The two sub-hypotheses are expanded together and the score is computed based on the sum of log probabilities of the output token pairs. For a beam size B, the B best hypotheses are retained based on this score. In this setup, both sub-hypotheses evolve jointly, which resembles the training process more than in the case of two different beams. A limitation of this joint-beam strategy is that, in extreme cases, one of the task (ASR or ST) may only have a single hypothesis. Indeed, at a decoding step $t+1$, we take the best B predictions $(\hat{y}_t, \hat{z}_t)$ in terms of their sum of scores $s(y_t, z_t) \triangleq \log p(y_t \mid \mathbf{y}_{<t}, \mathbf{z}_{<t}) + \log p(z_t \mid \mathbf{y}_{<t}, \mathbf{z}_{<t})$; it can happen that, *e.g.*, some $\hat{y}_t$ has a so dominant score that it is selected for all the hypotheses, *i.e.* the B (different) hypotheses have a single $\hat{y}_t$ and B different $\hat{z}_t$. We leave the design of a joint-beam strategy with enforced diversity to future work. Finally, to produce translations for multiple target languages in our system, it suffices to feed different language-specific tokens to the dual-decoder at decoding time.

5 Experiments

5.1 Dataset

To build a one-to-many model that can jointly transcribe and translate, we use MuST-C (Di Gangi et al., 2019), which is currently the largest publicly available one-to-many speech translation dataset.[7] MuST-C covers language pairs from English to eight different target languages including Dutch, French, German, Italian, Portuguese, Romanian, Russian, and Spanish. Each language direction includes a triplet of source input speech, source transcription, and target translation, with size ranging from 385 hours (Portuguese) to 504 hours (Spanish). We refer to the original paper for more details.

5.2 Implementation details

Our implementation is based on the ESPnet-ST toolkit (Inaguma et al., 2020).[8] In the following, we provide details for reproducing the results. The pipeline is identical for all experiments.

[7] Smaller datasets include Europarl-ST (Iranzo-Sánchez et al., 2020) and MaSS (Boito et al., 2020). Recently, a very large many-to-many dataset called CoVoST-2 (Wang et al., 2020b) has been released, while its predecessor CoVoST (Wang et al., 2020a) only covers the many-to-one scenario.

[8] https://github.com/espnet/espnet

Models All experiments use the same encoder architecture with 12 layers. The decoder has 6 layers, except for the independent-decoder model where we also include a 8-layer version (`independent++`) to compare the effects of dual-attention against simply increasing the number of model parameters.

Text pre-processing Transcriptions and translations were normalized and tokenized using the Moses tokenizer (Koehn et al., 2007). The transcription was lower-cased and the punctuation was stripped. A joint BPE (Sennrich et al., 2016) with 8000 merge operations was learned on the concatenation of the English transcription and all target languages. We also experimented with two separate dictionaries (one for English and another for all target languages), but found that the results are worse.

Speech features We used Kaldi (Povey et al., 2011) to extract 83-dimensional features (80-channel log Mel filter-bank coefficients and 3-dimensional pitch features) that were normalized by the mean and standard deviation computed on the training set. Following common practice (Inaguma et al., 2019; Wang et al., 2020c), utterances having more than 3000 frames or more than 400 characters were removed. For data augmentation, we used *speed pertubation* (Ko et al., 2015) with three factors of 0.9, 1.0, and 1.1 and *SpecAugment* (Park et al., 2019) with three types of deterioration including time warping (W), time masking (T) and frequency masking (F), where $W = 5, T = 40$, and $F = 30$.

Optimization Following standard practice for training Transformer, we used the Adam optimizer (Kingma and Ba, 2015) with Noam learning rate schedule (Vaswani et al., 2017), in which the learning rate is linearly increased for the first 25K warm-up steps then decreased proportionally to the inverse square root of the step counter. We set the initial learning rate to $1\text{e}{-3}$ and the Adam parameters to $\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 1\text{e}{-9}$. We used a batch size of 32 sentences per GPU, with gradient accumulation of 2 training steps. All models were trained on a single-machine with 8 32GB GPUs for 250K steps unless otherwise specified. As for model initialization, we trained an independent-decoder model with the two decoders having *shared* weights for 150K steps and used its weights to initialize the other models. This resulted in much faster convergence for all models. We also included this shared model in the experiments, and for a fair comparison, we trained it for additional 250K steps. Finally, for decoding, we used a beam size of 10 with length penalty of 0.5.[9]

5.3 Results and analysis

In this section, we report detokenized case-sensitive BLEU[10] (Papineni et al., 2002) on the MuST-C `dev` sets (Table 1). Results on the `test` sets are discussed in Section 5.4. Following previous work (Inaguma et al., 2020), we remove non-verbal tokens in evaluation.[11] In Table 1, there are 3 main groups of models, corresponding to independent-decoder, cross dual-decoder (`crx`), and parallel dual-decoder (`par`), respectively. In particular, `independent++` corresponds to a 8-decoder-layer model and will serve as our strongest baseline for comparison. Figure 2 shows the relative performance of some representative models with respect to this baseline, together with their validation accuracies. In the following, when comparing models, we implicitly mean "on average" (over the 8 languages), except otherwise specified.

Parallel *vs.* cross Under the same configurations, parallel models outperform their cross counterparts in terms of translation (line 5 *vs.* line 13, line 6 *vs.* line 14, and line 7 *vs.* line 16), showing an improvement of 0.7 BLEU on average. In terms of recognition, however, the parallel architecture has on average a 0.33% higher (worse) WER compared to the cross models. On the other hand, parallel dual-decoders perform better than independent decoders in both translation and recognition tasks, except for the asymmetric case (line 12), the at-self and at-source with `sum` merging configuration (line 17), and the wait-k model where ST is ahead of ASR (line 19). This shows that both the translation and recognition tasks can benefit from the tight interaction between the two decoders, *i.e.* it is possible to achieve no trade-off between BLEUs and WERs for the parallel models compared to the independent architecture. This is not the case, however, for the cross dual-decoders that feature weaker interaction than the parallel ones.

[9] For a hypothesis of length L, a length penalty p means a score of pL will be added to the (log probability) score of that hypothesis (Section 4.2). Therefore, longer hypotheses are favored, or equivalently, shorter hypotheses are "penalized".

[10] We also tried sacreBLEU (Post, 2018) and found that the results are identical.

[11] This is for a fair comparison with the results of Inaguma et al. (2020), presented in Section 5.4.

No	type	side	self	src	merge	params	de	es	fr	it	nl	pt	ro	ru	avg	WER
1	independent (shared)					31.3M	19.40	27.77	24.65	19.93	21.53	24.24	18.19	10.99	20.84	14.2
2	independent					44.8M	20.11	28.18	25.61	20.76	21.83	25.45	18.45	11.31	21.46	12.6
3	independent++					51.2M	20.25	29.48	26.10	21.05	22.34	**26.71**	19.67	12.10	22.21	12.9
4	crx	st	-	✓	sum	46.4M	20.01	28.57	25.86	20.66	22.26	25.36	19.06	12.00	21.72	12.7
5	crx	both	-	✓	concat	51.2M	20.36	28.51	25.80	21.18	22.10	25.24	19.55	11.89	21.83	12.3
6	crx	both	-	✓	sum	48.0M	19.99	28.87	26.09	20.94	21.67	25.42	18.85	11.83	21.71	12.2
7	crx	both	✓	✓	concat	54.3M	20.07	28.73	26.01	20.93	22.59	25.60	19.08	12.46	21.93	12.4
8	crx	both	✓	✓	sum	51.2M	20.38	28.90	26.64	21.07	22.61	26.23	19.44	12.12	22.17	**12.1**
9	crx*	both	✓	-	sum	48.0M	19.72	27.96	25.49	20.52	21.56	25.01	18.53	11.33	21.26	12.8
10	crx*	both	✓	-	sum†	48.0M	18.62	27.11	24.41	19.73	20.47	24.49	17.23	11.09	20.39	12.8
11	crx*	both	✓	✓	sum	51.2M	19.54	28.17	25.68	20.95	21.55	24.77	18.76	11.28	21.34	12.3
12	par	st	✓	✓	concat	49.6M	20.57	28.84	26.08	20.85	22.11	25.70	19.36	11.90	21.93	13.0
13	par	both	-	✓	concat	51.2M	20.84	29.51	26.44	21.53	22.68	25.94	19.04	12.60	22.32	12.5
14	par	both	-	✓	sum	48.0M	20.85	29.18	26.38	**22.14**	22.87	26.49	19.70	12.74	22.54	12.7
15	par	both	✓	-	sum	48.0M	20.56	29.21	26.54	21.07	22.51	25.75	19.64	**12.80**	22.26	12.8
16	par	both	✓	✓	concat	54.3M	**21.22**	29.50	**26.66**	21.74	22.76	26.66	**20.25**	12.79	22.70	12.7
17	par	both	✓	✓	sum	51.2M	20.95	28.67	26.45	21.31	22.29	25.87	19.53	12.24	22.16	12.8
18	par	both^{R3}	-	✓	sum	48.0M	**21.22**	**30.12**	26.53	22.06	**23.37**	26.59	19.82	12.54	**22.78**	12.6
19	par	both^{T3}	-	✓	sum	48.0M	20.35	28.61	25.94	21.22	22.12	25.19	19.36	11.99	21.85	13.6

*no normalization for dual-attention input, †sum merging has $\lambda = 0.3$ fixed,
^{R3}ASR is 3 steps ahead of ST, ^{T3}ST is 3 steps ahead of ASR.

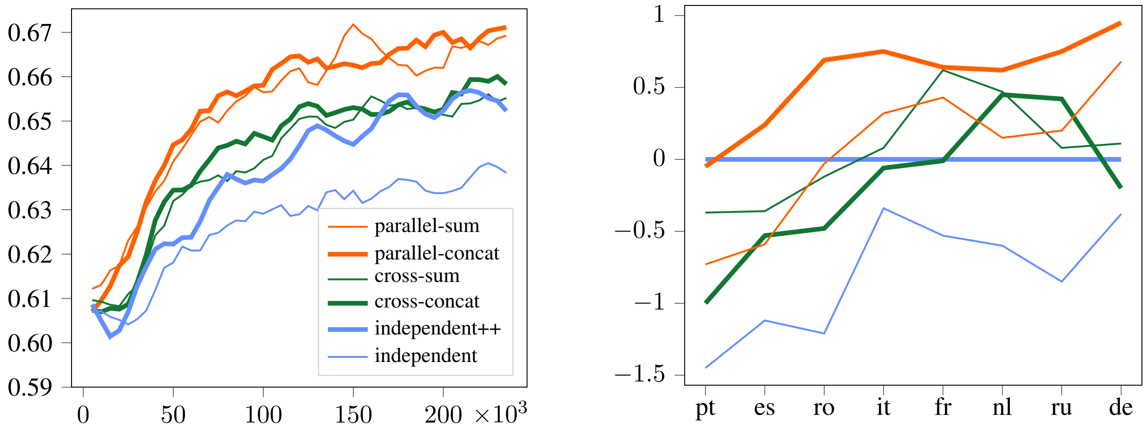
Table 1: BLEU and (average) WER on MuST-C dev set. In the second column (type), crx and parallel denote the cross and parallel dual-decoder, respectively. In the third column (side), st means only ST attends to ASR. Line 1 corresponds to the independent-decoder model where the weights of the two decoders are shared, and independent++ corresponds to the model with 8 decoder layers (instead of 6). It should be noted that line 10 corresponds to the model proposed by Liu et al. (2020). The values that are better than the baseline (independent++) are underlined and colored in blue, while the best values are highlighted in bold.

Interestingly, there is a slight trade-off between the parallel and cross designs: the parallel models are better in terms of BLEUs but worse in terms of WERs. This is to some extent similar to previous work where models having different types of trade-offs between BLEUs and WERs (He et al., 2011; Sperber et al., 2020; Chuang et al., 2020). It should be emphasized that most of the dual-decoder models have fewer or same numbers of parameters compared to independent++. This confirms our intuition that the tight connection between the two decoders in the parallel architecture improves performance. The cross dual-decoders perform relatively well compared to the baseline of two independent decoders with the same number of layers (6), but not so well compared to the stronger baseline with 8 layers.

Symmetric *vs*. asymmetric In some experiments, we only allow the ST decoder to attend to the ASR decoder. For the cross dual-decoder, this did not yield noticeable improvements in terms of BLEU (21.72 at line 4 *vs*. 21.71 at line 6), while for the parallel architecture, the results are worse (21.93 at line 12 *vs*. 22.70 at line 16). The symmetric models also outperform the asymmetric counterparts in terms of WER (12.7 at line 4 *vs*. 12.2 at line 6, 13.0 at line 12 *vs*. 12.7 at line 16). It is confirmed again that the two tasks are complementary and can help each other: removing the ASR-to-ST attention hurts performance. In fact, examining the learnable λ in the sum merging operator shows that the decoders learn to attend to each other, though at different rates. We observed that for the same layer depth, the ST decoder always attends more to the ASR one, and for both of them λ increases with the depth of the layer.

At-self dual-attention *vs*. at-source dual-attention For the parallel dual-decoder, the at-source dual-attention produces better results than the at-self counterpart (BLEU: 22.54 *vs*. 22.26, WER: 12.7 *vs*. 12.8 at line 14 *vs*. line 15), while the combination of both does not improve the results (BLEU 22.16, WER 12.8 at line 17). For the concat merging, using both yields better results in terms of translation but slightly hurts the recognition task (BLEU: 22.70 *vs*. 22.32, WER: 12.7 *vs*. 12.5 at line 16 *vs*. line 13).

Sum *vs*. concat merging The impact of merging operators is not consistent across different models. If we focus on the parallel dual-decoder, sum is better for models with only at-source attention (line 13 *vs*. line 14) and concat is better for models using both at-self and at-source attention (line 16 *vs*. line 17).



(a) ST validation accuracy per training step.

(b) Relative BLEU on MuST-C `dev` set.

Figure 2: Results on the MuST-C `dev` set. The models listed in the legends correspond respectively to lines 17, 16, 8, 7, 3, 2 in Table 1. The baseline used for relative BLEU is `independent++`. One can observe that the parallel models consistently outperform the others in terms of validation accuracy. Best viewed in color.

Input normalization and learnable sum Some experiments confirm the importance of normalizing the input fed to the dual-attention layers (*i.e.* the LayerNorm layers shown in Figure 1c). The results show that normalization substantially improves the performance (BLEU: 22.17 *vs.* 21.34, WER: 12.1 *vs.* 12.3 at line 8 *vs.* at line 11). It is also beneficial to use learnable weights compared to a fixed value for the `sum` merging operator (Equation (14)) (BLEU: 21.26 *vs.* 20.39 at line 9 *vs.* line 10). Note that the fixed weight and non-normalization configuration corresponds to the model of Liu et al. (2020) (line 10).

Wait-k policy We compare a non-wait-k parallel dual-decoder (line 14) with its wait-k ($k = 3$) counterparts. From the results, one can observe that letting ASR be ahead of ST (line 18) improves the performance (BLEU: 22.78 *vs.* 22.54, WER: 12.6 *vs.* 12.7), while letting ST be ahead of ASR (line 19) considerably worsen the results (BLEU: 21.85, WER: 13.6). This confirms our intuition that the ST task is more difficult and should not take the lead in the dual-decoder models.

ASR results From the results (last column of Table 1), one can observe that the dual-decoder models outperform the baseline `indepedent++`, except for the asymmetric case and the wait-k model where ST is 3 steps ahead of ASR. While using a single decoder leads to an average of 14.2% WER, all other symmetric architectures with two decoders (except the ASR-waits-for-ST) have better and rather stable WERs (from 12.1% to 13.0%). Detailed results for each data subset are provided in the Appendix.

5.4 Comparison to state-of-the-art

To avoid a hyper-parameter search over the test set, we only select three of our best models together with the baseline `independent++` for evaluation. All of the three models are symmetric parallel dual-decoders, the first one has at-source dual-attention with `sum` merging, the second one has both at-self and at-source dual-attentions with `concat` merging, and the last one is a wait-k model in which ASR is 3 steps ahead of ST. These models correspond to lines 5, 6, and 7 of Table 2, and will be referred to respectively as `par++`, `par`, and `par`$^{\mathrm{R3}}$ in the sequel. For `par++` we increase the number of decoder layers from 6 to 8, thus increasing the number of parameters from 48M to 51.2M, matching that of the baseline. We do not do this for `par`$^{\mathrm{R3}}$ (48M) as this model already has a higher latency due to the wait-k. All models are trained for 550K steps, corresponding to 25 epochs. Following Inaguma et al. (2020), we use the average of five checkpoints with the best validation accuracies on the dev sets for evaluation.

We compare the results with the previous work (Gangi et al., 2019) in the multilingual setting. In addition, to demonstrate the competitive performance of our models, we also include the best existing translation performance on MuST-C (Inaguma et al., 2020), although these results were obtained with bilingual systems and from a sophisticated training recipe. Indeed, to obtain the results for each language pair (*e.g.* `en-de`), Inaguma et al. (2020) pre-trained an ASR model and an MT model to initialize the

No	type	side	self	src	merge	epochs	de	es	fr	it	nl	pt	ro	ru	avg	WER
1	Bilingual one-to-one (Inaguma et al., 2020)					50	22.91	27.96	32.69	23.75	27.43	28.01	21.90	**15.75**	25.05	12.0
2	Multilingual one-to-many (Gangi et al., 2019)						17.70	20.90	26.50	18.00	20.00	22.60	-	-	-	-
3	Multilingual one-to-many (Gangi et al., 2019)						16.50	18.90	24.50	16.20	17.80	20.80	15.90	9.80	17.55	-
4	independent++					25	22.82	27.20	32.11	23.34	26.67	28.98	21.37	14.34	24.60	11.6
5	par++	both	-	✓	sum	25	**23.63**	**28.12**	**33.45**	**24.18**	**27.55**	**29.95**	**22.87**	15.21	**25.62**	**11.4**
6	par	both	✓	✓	concat	25	22.74	27.59	32.86	23.50	26.97	29.51	21.94	14.88	25.00	11.6
7	par^{R3}	both	-	✓	sum	25	22.84	27.92	32.12	23.61	27.29	29.48	21.16	14.50	24.87	11.6

Table 2: BLEU on MuST-C tst-COMMON test set. Line 2 corresponds to the best multilingual models of Gangi et al. (2019) trained separately for {de,nl} and {es,fr,it,pt}, while line 3 is a single model trained on 8 languages. The WER in line 1 corresponds to the best result reported by Inaguma et al. (2020).[12]

weights of (respectively) the encoder and decoder for ST training. This means that to obtain the results for the 8 language pairs, 24 independent trainings had to be performed in total (3 for each language pair).

The results in Table 2 show that our models achieved very competitive performance compared to bilingual one-to-one models (Inaguma et al., 2020), despite being trained for only half the number of epochs. In particular, the par++ model achieved the best results, consistently surpassing the others on all languages (except on Russian where it is outperformed by the bilingual model). Our results also surpassed those of Gangi et al. (2019) by a large margin. We observe the largest improvements on Portuguese ($+1.94$ at line 5, $+1.50$ at line 6, and $+1.47$ at line 7, compared to the bilingual result at line 1), which has the least data among the 8 language pairs in MuST-C. This phenomenon is also common in multilingual neural machine translation where multilingual joint training has been shown to improve performance on low-resource languages (Johnson et al., 2017).

6 Conclusion

We introduced a novel dual-decoder Transformer architecture for synchronous speech recognition and multilingual speech translation. Through a dual-attention mechanism, the decoders in this model are at the same time able to specialize in their tasks while being helpful to each other. The proposed model also generalizes previously proposed approaches using two independent (or weakly tied) decoders or chaining ASR and ST. It is also flexible enough to experiment with settings where ASR is ahead of ST which makes it promising for (one-to-many) simultaneous speech translation. Experiments on the MuST-C dataset showed that our model achieved very competitive performance compared to state-of-the-art.

Acknowledgements

This work was supported by a Facebook AI SRA grant, and was granted access to the HPC resources of IDRIS under the allocations 2020-AD011011695 and 2020-AP011011765 made by GENCI. It was also done as part of the Multidisciplinary Institute in Artificial Intelligence MIAI@Grenoble-Alpes (ANR-19-P3IA-0003). We thank the anonymous reviewers for their insightful feedback.

References

Antonios Anastasopoulos and David Chiang. 2018. Tied multitask learning for neural speech translation. In Marilyn A. Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 82–91. Association for Computational Linguistics. 1, 2, 5

Ebrahim Ansari, amittai axelrod, Nguyen Bach, Ondřej Bojar, Roldano Cattoni, Fahim Dalvi, Nadir Durrani, Marcello Federico, Christian Federmann, Jiatao Gu, Fei Huang, Kevin Knight, Xutai Ma, Ajay Nagesh, Matteo Negri, Jan Niehues, Juan Pino, Elizabeth Salesky, Xing Shi, Sebastian Stüker, Marco Turchi, Alexander Waibel, and Changhan Wang. 2020. FINDINGS OF THE IWSLT 2020 EVALUATION CAMPAIGN. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 1–34, Online, July. Association for Computational Linguistics. 1

[12] https://github.com/espnet/espnet/blob/master/egs/must_c/asr1/RESULTS.md

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*. 3

Alexandre Bérard, Olivier Pietquin, Christophe Servan, and Laurent Besacier. 2016. Listen and translate: A proof of concept for end-to-end speech-to-text translation. In *NIPS Workshop on End-to-end Learning for Speech and Audio Processing*. 1

Alexandre Bérard, Laurent Besacier, Ali Can Kocabiyikoglu, and Olivier Pietquin. 2018. End-to-end automatic speech translation of audiobooks. *CoRR*, abs/1802.04200. 1

Marcely Zanon Boito, William Havard, Mahault Garnerin, Éric Le Ferrand, and Laurent Besacier. 2020. Mass: A large and clean multilingual corpus of sentence-aligned spoken utterances extracted from the bible. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 6486–6493. 6

Shun-Po Chuang, Tzu-Wei Sung, Alexander H. Liu, and Hung-yi Lee. 2020. Worse wer, but better bleu? leveraging word embedding as intermediate in multitask end-to-end speech translation. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5998–6003. Association for Computational Linguistics. 8

Mattia A Di Gangi, Roldano Cattoni, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2019. Must-c: a multilingual speech translation corpus. In *2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2012–2017. Association for Computational Linguistics. 1, 6

Kunihiko Fukushima and Sei Miyake. 1982. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer. 3

Mattia Antonino Di Gangi, Matteo Negri, and Marco Turchi. 2019. One-to-many multilingual end-to-end speech translation. In *IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2019, Singapore, December 14-18, 2019*, pages 585–592. IEEE. 1, 9, 10

Xiaodong He, Li Deng, and Alex Acero. 2011. Why word error rate is not a good metric for speech recognizer training for the speech translation task? In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5632–5635. IEEE. 8

Hirofumi Inaguma, Kevin Duh, Tatsuya Kawahara, and Shinji Watanabe. 2019. Multilingual end-to-end speech translation. In *IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2019, Singapore, December 14-18, 2019*, pages 570–577. IEEE. 1, 7

Hirofumi Inaguma, Shun Kiyono, Kevin Duh, Shigeki Karita, Nelson Enrique Yalta Soplin, Tomoki Hayashi, and Shinji Watanabe. 2020. Espnet-st: All-in-one speech translation toolkit. *arXiv preprint arXiv:2004.10234*. 6, 7, 9, 10

Javier Iranzo-Sánchez, Joan Albert Silvestre-Cerdà, Javier Jorge, Nahuel Roselló, Adrià Giménez, Albert Sanchis, Jorge Civera, and Alfons Juan. 2020. Europarl-st: A multilingual corpus for speech translation of parliamentary debates. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8229–8233. IEEE. 6

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. *CoRR*. 2

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351. 10

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 7

Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. 2015. Audio augmentation for speech recognition. In *Sixteenth Annual Conference of the International Speech Communication Association*. 7

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics. 7

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551. 3

Yuchen Liu, Jiajun Zhang, Hao Xiong, Long Zhou, Zhongjun He, Hua Wu, Haifeng Wang, and Chengqing Zong. 2020. Synchronous speech recognition and speech-to-text translation with interactive decoding. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8417–8424. AAAI Press. 2

Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, Hua Wu, and Haifeng Wang. 2019. STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. In *Proc. of ACL*. 3

Toan Q Nguyen and Julian Salazar. 2019. Transformers without tears: Improving the normalization of self-attention. *arXiv preprint arXiv:1910.05895*. 3

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318. 7

Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. 2019. Specaugment: A simple data augmentation method for automatic speech recognition. In Gernot Kubin and Zdravko Kacic, editors, *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, pages 2613–2617. ISCA. 7

Matt Post. 2018. A call for clarity in reporting bleu scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191. 7

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. 2011. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society. 7

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics. 7

Matthias Sperber, Graham Neubig, Jan Niehues, and Alex Waibel. 2019. Attention-passing models for robust and data-efficient end-to-end speech translation. *Transactions of the Association for Computational Linguistics*, 7:313–325. 2, 5

Matthias Sperber, Hendra Setiawan, Christian Gollan, Udhyakumar Nallasamy, and Matthias Paulik. 2020. Consistent transcription and translation of speech. *CoRR*, abs/2007.12741. 1, 5, 8

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826. 6

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008. 1, 2, 3, 7

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. 2019. Learning deep transformer models for machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1810–1822. 3

Changhan Wang, Juan Pino, Anne Wu, and Jiatao Gu. 2020a. Covost: A diverse multilingual speech-to-text translation corpus. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 4197–4203, Marseille, France, May. European Language Resources Association. 1, 6

Changhan Wang, Anne Wu, and Juan Pino. 2020b. Covost 2: A massively multilingual speech-to-text translation corpus. *arXiv preprint arXiv:2007.10310*. 6

Chengyi Wang, Yu Wu, Shujie Liu, Ming Zhou, and Zhenglu Yang. 2020c. Curriculum pre-training for end-to-end speech translation. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 3728–3738. Association for Computational Linguistics. 7

Ron J. Weiss, Jan Chorowski, Navdeep Jaitly, Yonghui Wu, and Zhifeng Chen. 2017. Sequence-to-sequence models can directly translate foreign speech. In Francisco Lacerda, editor, *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20-24, 2017*, pages 2625–2629. ISCA. 1

Appendix

We present the detailed model architecture for the *cross dual-decoder Transformer* in Figure 3. Detailed WER results on MuST-C dev set are presented in Table 3.

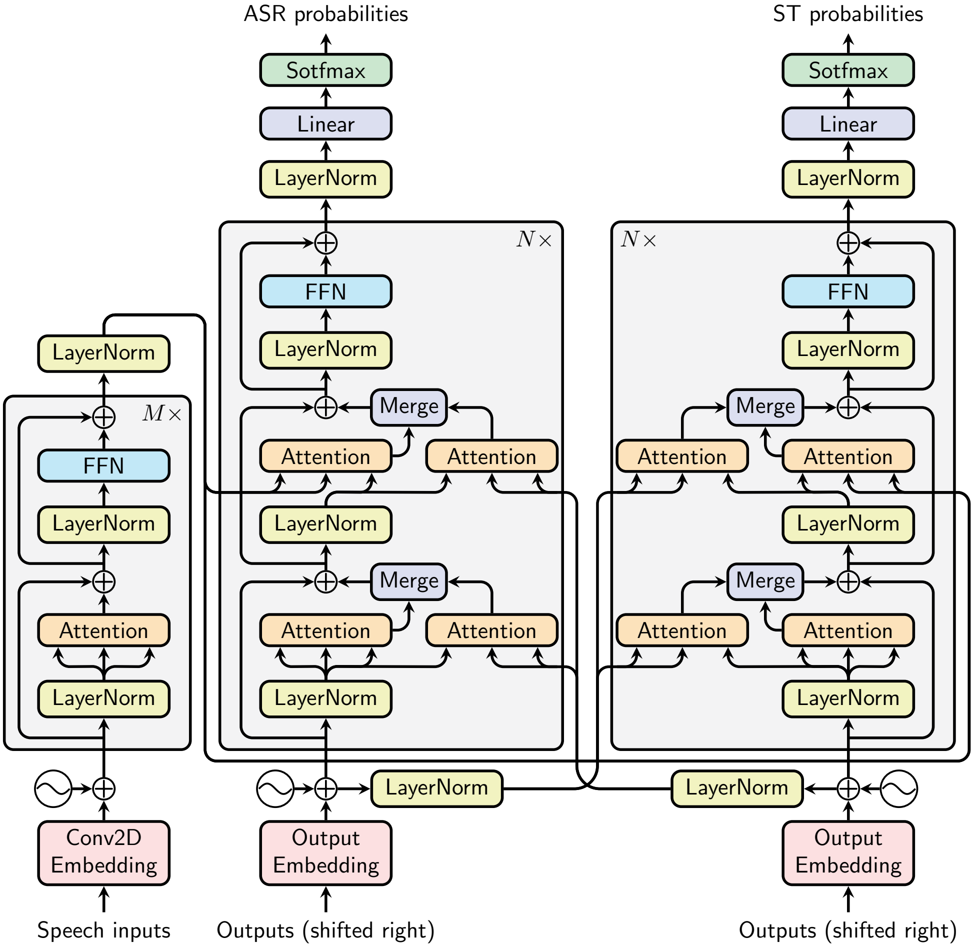


Figure 3: The *cross* dual-decoder Transformer. Unlike the *parallel* dual-decoder Transformer, here one decoder attends to the previous decoding-step outputs of the other and there is no interaction between their hidden states.

No	type	side	self	src	merge	params	de	es	fr	it	nl	pt	ro	ru	avg
1	independent (shared)					31.3M	14.9	13.8	14.5	13.6	14.0	13.6	14.8	14.0	14.2
2	independent					44.8M	12.6	12.6	13.1	11.9	12.4	12.3	13.1	12.4	12.6
3	independent++					51.2M	13.4	12.5	14.0	12.2	12.7	12.8	13.3	12.5	12.9
4	crx	st	-	✓	sum	46.4M	12.8	13.0	13.2	12.8	12.1	12.4	12.2	12.8	12.7
5	crx	both	-	✓	concat	51.2M	12.3	12.9	12.4	11.9	12.3	12.3	12.2	12.4	12.3
6	crx	both	-	✓	sum	48.0M	12.2	12.6	12.4	12.0	12.3	12.0	12.0	12.2	12.2
7	crx	both	✓	✓	concat	54.3M	12.5	12.9	12.6	12.0	12.2	12.3	12.4	12.2	12.4
8	crx	both	✓	✓	sum	51.2M	12.1	12.7	12.1	11.7	12.0	12.0	12.0	12.1	12.1
9	crx*	both	✓	-	sum	48.0M	12.7	12.9	13.3	12.5	12.6	12.5	12.6	12.9	12.8
10	crx*	both	✓	-	sum†	48.0M	12.7	12.8	13.2	12.3	11.5	13.1	12.7	12.9	12.8
11	crx*	both	✓	✓	sum	51.2M	12.2	12.6	12.4	11.8	12.1	12.2	12.4	12.4	12.3
12	par	st	✓	✓	concat	49.6M	13.0	13.7	13.1	12.5	13.1	12.7	13.3	12.9	13.0
13	par	both	-	✓	concat	51.2M	12.6	13.0	12.6	12.1	12.4	12.3	12.5	12.5	12.5
14	par	both	-	✓	sum	48.0M	12.8	13.0	13.2	12.2	12.8	12.4	12.6	12.6	12.7
15	par	both^{R3}	-	✓	sum	48.0M	13.1	13.4	12.8	12.3	12.9	12.5	12.7	12.9	12.8
16	par	both^{T3}	-	✓	sum	48.0M	14.0	14.3	13.9	13.4	13.9	13.6	13.8	13.5	13.8
17	par	both	✓	-	sum	48.0M	12.6	13.1	13.3	12.4	12.6	12.6	12.6	12.8	12.8
18	par	both	✓	✓	concat	54.3M	12.5	13.3	12.9	12.3	12.7	12.6	12.6	12.5	12.7
19	par	both	✓	✓	sum	51.2M	12.7	13.4	12.8	12.5	12.8	12.6	12.7	12.7	12.8

Table 3: Word error rate on MuST-C `dev` set. The values that are better than the baseline (`independent++`) are underlined and colored in blue.