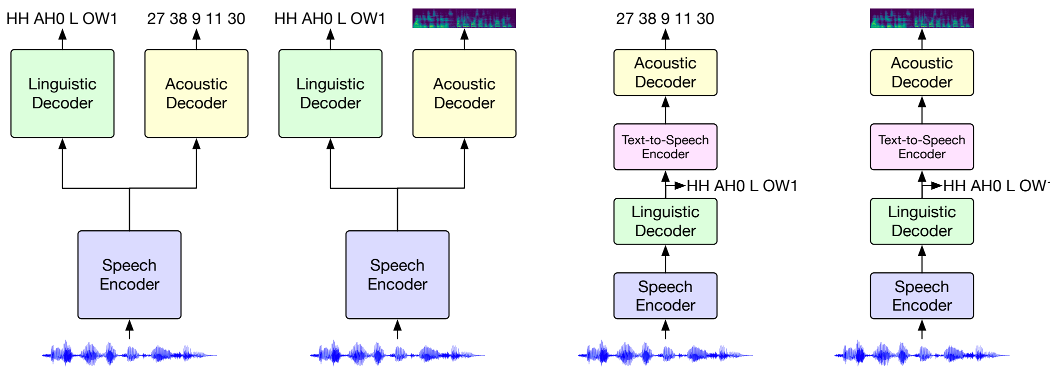
A Details of Baseline Models

In our experiments, we implement five baseline systems using *fairseq*: S2UT, Translatotron, UnitY, Translatotron 2, and TranSpeech. We reproduce TranSpeech with their open-source implementations[9]. In this section, we mainly introduce the configurations of the other four baseline systems.

Figure 2 shows the model architectures of these models. In terms of model architecture, S2UT and Translatotron are single-pass S2ST models while UnitY and Translatotron 2 are two-pass S2ST models. In terms of predicted targets, S2UT and UnitY predict discrete units while Translatotron and Translatotron 2 predict mel-spectrograms. Below we describe the details of each model. The detailed hyperparameters can be found in Table 5.



(a) S2UT (b) Translatotron (c) UnitY (d) Translatotron 2

Figure 2: Overview of baseline models.

S2UT Our implemented S2UT model includes three parts: a speech encoder, a linguistic decoder, and an acoustic decoder. The speech encoder is the same as DASpeech. The linguistic decoder is appended to the top layer of the speech encoder for multi-task learning, which predicts the target phonemes during training. The acoustic decoder generates the reduced discrete units derived from the 11-th layer of the pretrained mHuBERT model[10]. We do not include other auxiliary tasks and remove CTC decoding in Lee et al. [5] for simplification. The model is trained from scratch for 100k steps. We use beam search with a beam size of 10.

Translatotron The speech encoder and linguistic decoder of Translatotron are the same as S2UT. The acoustic decoder generates mel-spectrograms autoregressively. The pre-net dimension is 32 and the reduction factor of the acoustic decoder is 5. The model is trained from scratch for 100k steps.

UnitY UnitY is a two-pass model that includes four parts: a speech encoder, a linguistic decoder, a text-to-speech encoder, and an acoustic decoder. The architecture of the speech encoder, linguistic decoder, and acoustic decoder are the same as S2UT. The additional text-to-speech encoder is used to bridge the gap in representations between two decoders. We remove R-Drop training for simplification. We first conduct S2TT pretraining and finetune the model for 50k steps. We set the beam size of the first-pass and second-pass decoder to 10 and 1, respectively.

Translatotron 2 The model architecture of Translatotron 2 is similar to UnitY except that the second decoder generates mel-spectrograms rather than discrete units. The reduction factor of the acoustic decoder is set to 5. We first conduct S2TT pretraining and finetune the model for 50k steps. The beam size is set to 10 for the first-pass decoder.

For all the above models, we save checkpoints every 2000 steps and average the last 5 checkpoints for evaluation, which is the same as DASpeech. For S2UT and UnitY, we use the pretrained unit-based HiFi-GAN[11] vocoder to synthesize waveform. For Translatotron and Translatotron 2, we use the same pretrained HiFi-GAN vocoder as DASpeech.

[9] https://github.com/Rongjiehuang/TranSpeech

[10] https://dl.fbaipublicfiles.com/hubert/mhubert_base_vp_en_es_fr_it3_L11_km1000.bin

[11] https://dl.fbaipublicfiles.com/fairseq/speech_to_speech/vocoder/code_hifigan/mhubert_vp_en_es_fr_it3_400k_layer11_km1000_lj/g_00500000

Table 5: Hyperparameters of DASpeech and baseline models.

Hyperparameters		S2UT	Translatotron	UnitY	Translatotron 2	DASpeech
Speech Encoder	conv_kernel_sizes	(5, 5)	(5, 5)	(5, 5)	(5, 5)	(5, 5)
	encoder_type	conformer	conformer	conformer	conformer	conformer
	encoder_layers	12	12	12	12	12
	encoder_embed_dim	256	256	256	256	256
	encoder_ffn_embed_dim	2048	2048	2048	2048	2048
	encoder_attention_heads	4	4	4	4	4
	encoder_pos_enc_type	relative	relative	relative	relative	relative
	depthwise_conv_kernel_size	31	31	31	31	31
Linguistic Decoder	decoder_layers	4	4	4	4	4
	decoder_embed_dim	512	512	512	512	512
	decoder_ffn_embed_dim	2048	2048	2048	2048	2048
	decoder_attention_heads	8	8	8	8	8
	label_smoothing	0.1	0.1	0.1	0.1	0.0
	s2t_loss_weight	8.0	0.1	8.0	0.1	1.0
Text-to-Speech Encoder	encoder_layers	-	-	2	2	-
	encoder_embed_dim	-	-	512	512	-
	encoder_ffn_embed_dim	-	-	2048	2048	-
	encoder_attention_heads	-	-	8	8	-
Acoustic Decoder	decoder_layers	6	6	2	6	8
	decoder_embed_dim	512	512	512	512	256
	decoder_ffn_embed_dim	2048	2048	2048	2048	1024
	decoder_attention_heads	8	8	8	8	4
	label_smoothing	0.1	-	0.1	-	-
	n_frames_per_step	1	5	1	5	1
	unit_dictionary_size	1000	-	1000	-	-
	var_pred_hidden_dim	-	-	-	-	256
	var_pred_kernel_size	-	-	-	-	3
	var_pred_dropout	-	-	-	-	0.5
	s2s_loss_weight	1.0	1.0	1.0	1.0	5.0
Training	lr	1e-3	1e-3	1e-3	1e-3	1e-3
	lr_scheduler	inverse_sqrt	inverse_sqrt	inverse_sqrt	inverse_sqrt	inverse_sqrt
	warmup_updates	4000	4000	4000	4000	4000
	warmup_init_lr	1e-7	1e-7	1e-7	1e-7	1e-7
	optimizer	Adam	Adam	Adam	Adam	Adam
	dropout	0.1	0.1	0.1	0.1	0.1
	max_tokens	40k×4	40k×4	40k×4	40k×4	40k×8
	weight_decay	0.0	0.0	0.0	0.0	0.01
	clip_norm	1.0	1.0	1.0	1.0	1.0
	max_update	100k	100k	50k	50k	50k

B Detailed Results on CVSS-C X→En Datasets

Table 6 summarizes the detailed results of each language pair on CVSS-C `test` sets of the multilingual X→En S2ST models.

Table 6: Results on CVSS-C `test` sets of the multilingual X→En S2ST models.

Models		Avg.	High				Mid				
			Fr	De	Ca	Es	Fa	It	Ru	Zh	Pt
S2UT [5]		5.15	19.65	13.35	15.37	18.58	1.43	14.47	7.94	0.93	6.42
UnitY [7]		8.15	27.27	20.81	24.22	27.58	3.63	21.68	10.86	4.16	8.56
Translatotron 2 [6]		8.74	28.04	21.54	25.34	28.77	4.23	23.66	13.41	4.49	9.54
DASpeech	+ Lookahead	7.42	25.43	17.87	22.58	25.49	3.01	20.80	12.96	2.86	7.90
($\lambda = 0.5$)	+ Joint-Viterbi	7.43	25.39	18.36	22.33	25.10	2.81	20.76	12.94	3.05	7.89

Models		Low											
		Nl	Tr	Et	Mn	Ar	Lv	Sl	Sv	Cy	Ta	Ja	Id
S2UT [5]		4.67	0.52	0.36	0.14	0.56	0.39	0.73	1.28	0.66	0.17	0.20	0.38
UnitY [7]		10.60	3.79	1.07	0.12	0.78	1.50	0.81	1.38	1.74	0.10	0.15	0.27
Translatotron 2 [6]		11.17	4.58	1.12	0.32	1.35	1.37	0.93	1.49	1.50	0.10	0.22	0.33
DASpeech	+ Lookahead	9.04	1.75	0.04	0.08	0.64	1.43	1.20	1.33	0.70	0.09	0.29	0.29
($\lambda = 0.5$)	+ Joint-Viterbi	9.43	1.66	0.07	0.08	0.48	1.48	1.30	1.30	0.85	0.09	0.31	0.32

C Effects of the Graph Size

In this section, we investigate how the graph size affects the performance. We vary the size factor λ from 0.25 to 1.5, and measure the translation quality of both the S2TT DA-Transformer model and DASpeech on the CVSS-C Fr→En `test` set. As shown in Figures 3 and 4, we observe that the performance of S2TT DA-Transformer keeps increasing as the graph size gets larger, which is consistent with the observations in machine translation [11, 53]. However, DASpeech performs best at $\lambda = 0.5$ and shows a performance drop at larger λ. We speculate that this is because larger graph size makes end-to-end training more challenging. We will investigate this issue in the future.

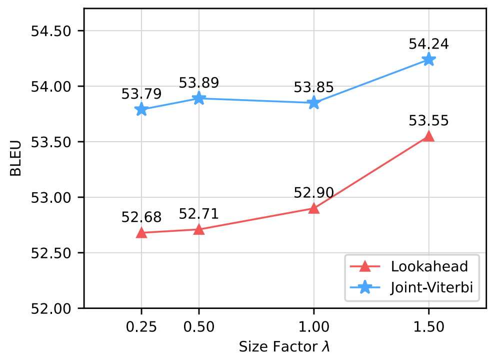


Figure 3: Phoneme-level BLEU scores of the S2TT DA-Transformer under different size factor λ.

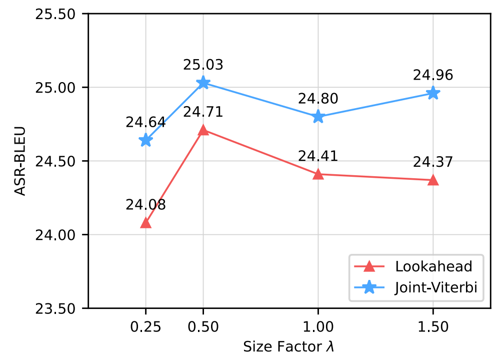


Figure 4: ASR-BLEU scores of DASpeech under different size factor λ.

D Speedup Under Batch Decoding

As Gu and Kong [41] pointed out, the speed benefits of non-autoregressive models may degrade during batch decoding. To better understand this problem, we evaluate the speedup ratio under different decoding batch sizes. As shown in Figure 5, the speedup ratio keeps dropping as the decoding batch size increases. Nevertheless, DASpeech ($\lambda = 0.5$ with Joint-Viterbi decoding) still achieves more than $6\times$ speedup with a decoding batch size of 64 and maintains comparable performance with Translatotron 2.

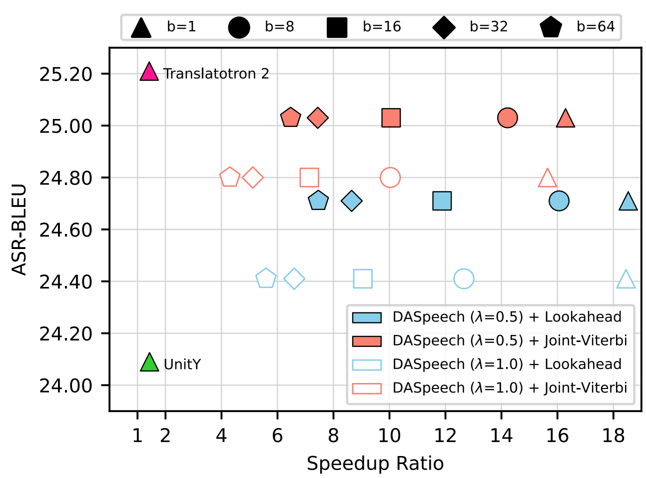


Figure 5: Speedup ratio compared to S2UT baseline (not shown in the figure) and ASR-BLEU score on the CVSS-C Fr→En `test` set with different batch decoding sizes ($b \in \{1, 8, 16, 32, 64\}$).

E Best-Path Training

Best-path-training selects the most probable path $\hat{A} = (\hat{a}_1, ..., \hat{a}_M)$ and takes the hidden states on path $\hat{A}$ as input to the acoustic decoder. Formally, given the target phoneme sequence Y, we can find the most probable path $\hat{A} = \arg\max_{A \in \Gamma} P_\theta(Y, A|X)$ via Viterbi algorithm [20]. Specifically, we use $\delta_i(j)$ to denote the probability of the most probable path so far $(\hat{a}_1, ..., \hat{a}_i)$ with $\hat{a}_i = j$ that generates $(y_1, ..., y_i)$. Considering the definition of $a_1 = 1$, we have $\delta_1(1) = \mathbf{P}_{1,y_1}$ and $\delta_1(1 < j \leq L) = 0$. For $i > 1$, we can sequentially calculate $\delta_i(\cdot)$ from its previous step $\delta_{i-1}(\cdot)$ due to the Markov property:

$$\delta_i(j) = \max_{k<j}(\delta_{i-1}(k) \cdot \mathbf{E}_{k,j} \cdot \mathbf{P}_{j,y_i}), \tag{16}$$

$$\phi_i(j) = \arg\max_{k<j}(\delta_{i-1}(k) \cdot \mathbf{E}_{k,j} \cdot \mathbf{P}_{j,y_i}), \tag{17}$$

where $\phi_i(j)$ stores $\hat{a}_{i-1}$ of the most probable path so far $(\hat{a}_1, ..., \hat{a}_{i-1}, \hat{a}_i = j)$. After M iterations, we can obtain the most probable path by backtracking from $\hat{a}_M = L$:

$$\hat{a}_i = \phi_{i+1}(\hat{a}_{i+1}). \tag{18}$$

Finally, we select the hidden states on the most probable path, i.e., $\mathbf{z}_i = \mathbf{v}_{\hat{a}_i}$, as the input sequence of the acoustic decoder.