STTATTS: Unified Speech-To-Text And Text-To-Speech Model

Hawau Olamide Toyin[1], **Hao Li**[1,2], **Hanan Aldarmaki**[1]
[1]Mohamed Bin Zayed University of Artificial Intelligence, UAE
[2]Pinscreen, USA
{hawau.toyin,hao.li,hanan.aldarmaki}@mbzuai.ac.ae

Abstract

Speech recognition and speech synthesis models are typically trained separately, each with its own set of learning objectives, training data, and model parameters, resulting in two distinct large networks. We propose a parameter-efficient approach to learning ASR and TTS jointly via a multi-task learning objective and shared parameters. Our evaluation demonstrates that the performance of our multi-task model is comparable to that of individually trained models while significantly saving computational and memory costs (∼50% reduction in the total number of parameters required for the two tasks combined). We experiment with English as a resource-rich language, and Arabic as a relatively low-resource language due to shortage of TTS data. Our models are trained with publicly available data, and both the training code and model checkpoints are openly available for further research.[1]

1 Introduction

Fundamentally, text and speech are different representations of similar information, with text being a far more condensed form of the linguistic content of speech. Conversion between text and speech modalities in the form Automatic Speech Recognition (ASR), text-to-speech synthesis (TTS), or Voice Conversion (VC), are traditionally achieved by training separate ASR, TTS, and VC models as the input and output modalities and training objectives differ significantly. However, considering the recent developments in self-supervised and multi-modal pre-training, a more integrated approach can now be promising. A unified model trained simultaneously for multiple speech/text to speech/text, improves on generalization to new data, cross-task knowledge transfer, simplifies maintenance, and also reduces the computational and memory requirements for training, storage, and inference. Recent studies seek to achieve a smooth fusion of text and audio by developing unified audio-text models capable of addressing diverse tasks both within and across these modalities. While these models are considered multi-modal if they can process different input modalities, for our purposes we group audio-text models into two categories based on their output modality: uni-modal and cross-modal. We describe uni-modal approaches as models capable of generating output in a single modality only, such as Whisper (Radford et al., 2022), and Google USM (Zhang et al., 2023b), which only generate text outputs. Cross-modal approaches, on the other hand, are capable of generating outputs in both speech and text modalities, such as Viola (Wang et al., 2023), SpeechT5 (Ao et al., 2022), and SpeechGPT (Zhang et al., 2023a). Some of these models (Zhang et al., 2023a; Maiti et al., 2024) use discrete representations for audio tasks, merging text and audio tokens in a shared vocabulary while jointly training multiple tasks; other models, such as SpeechT5 (Ao et al., 2022) use continuous representations for audio. While SpeechT5 is pre-trained with a cross-modal objective, handling both text and speech as input and output modalities, the model is fine-tuned individually for downstream tasks such as ASR and TTS.

This work builds on these developments, particularly SpeechT5 (Ao et al., 2022), by achieving a truly cross-modal speech and text conversion in a single architecture. Unlike previous work, our approach does not separate the speech/text-to-speech/text tasks based on input/output modalities; instead, we fine-tune these components concurrently using a unified model and loss function with the help of a simple MLP-based task fusion module. The resulting model is a single encoder-decoder that can handle both modalities at the input and output, depending on the desired task. We summarize our contributions below:

[1]https://github.com/mbzuai-nlp/sttatts

Findings of the Association for Computational Linguistics: EMNLP 2024, pages 6853–6863
November 12-16, 2024

1. We propose a novel parameter-efficient fine-tuning methodology for jointly learning multiple speech tasks: **ASR and Multi-Speaker TTS**. We demonstrate the **efficiency** of our approach in terms of computational requirements, training time, and its **scalability** to additional tasks, namely **Voice Conversion**.

2. We empirically demonstrate the effectiveness and efficiency of the proposed approach, resulting in **improved performance** compared to the only comparable open-source model at the time of writing (i.e. VoxLM) with a fraction (1/2) of the parameters.

3. We demonstrate the **robustness** and **performance** of our approach, showing its applicability on both high-resource and **low-resource** settings, as we present the **first multi-modal, multi-task model** for the **Arabic** language.

2 Related Work

2.1 Multi-Task Speech Models

Radford et al. introduced Whisper, an encoder-decoder model trained on vast amount of speech-text (680K hours) data. Whisper a multilingual model with multitasking capabilities. However, it only uses speech as input and can not generate speech output. In SLAM (Bapna et al., 2021), the authors unified speech and text pre-training within a single model using a single encoder with the combined BERT (Devlin et al., 2019) and W2V-BERT (Chung et al., 2021) objectives on unlabeled text and speech. To align their model's representations across modalities, they used Translation Language Modeling (TLM) and Speech Text Matching (STM) alignment losses that use supervised speech-text recognition data. They show that joint pre-training improves model performance on downstream speech translation and recognition tasks. However, these models are not jointly trained for multi-task purposes and require dedicated fine tuning for each task.

Unified Speech-Text Models

SpeechT5 (Ao et al., 2022) introduces a multi-modal encoder-decoder pre-training approach for spoken language processing. The authors attempted a joint pre-training approach of speech and text to improve the model's performance on downstream speech/text tasks like ASR, TTS, speaker identification, speech enhancement, and voice conversion. They built on the transformer architecture (Vaswani et al., 2017), adding modal-specific pre-nets and post-nets to handle latent feature extraction/conversion for different modalities. Although their model is pre-trained jointly with speech and text data in a self-supervised manner, the supervised downstream models (e.g. ASR, TTS) were *trained individually for each task*. The model was trained and evaluated on English only. A subsequent work followed the same architecture and training paradigm for building an Arabic version of the model, named ArTST (Toyin et al., 2023), which also requires task-specific training.

Some recent methods employ a decoder-only framework post-conversion of continuous audio into discrete tokens, subsequently combining text and audio tokens into a unified vocabulary (Maiti et al., 2024; Zhang et al., 2023a; Wang et al., 2023). These models can generate both text and speech output from speech/text input. Some models (Maiti et al., 2024) discretize speech using k-means on features extracted from speech-text pre-trained models like HuBert (Hsu et al., 2021). However, their method can suffer from information loss caused by quantizing speech signals into discrete tokens and its performance highly depends on the value of k used for feature extraction. Moreover, combining text and discrete speech tokens into a vocabulary can lead to a large vocabulary size for multilingual training.

3 Our Method

In this section, we describe a unified **S**peech-**T**o-**T**ext **A**nd **T**ext-**T**o-**S**peech model, **STTATTS**, our proposed architecture for jointly training ASR and TTS. After unified self-supervised training as described in Ao et al. (2022), we propose utilizing a multi-task loss objective to optimize our model parameters for multiple tasks, along with a task fusion module to handle the different tasks. Unlike the fine-tuning methodology followed in SpeechT5 (Ao et al., 2022; Toyin et al., 2023), which results in a completely disjoint copy of the model for each task (see Figure 2: *Right*), `STTATTS` utilizes a task fusion module with negligible number of additional parameters to handle multiple tasks using the same encoder-decoder backbone (See Figure 1). This results in a ∼50% reduction in the total number of parameters required for the two tasks combined.

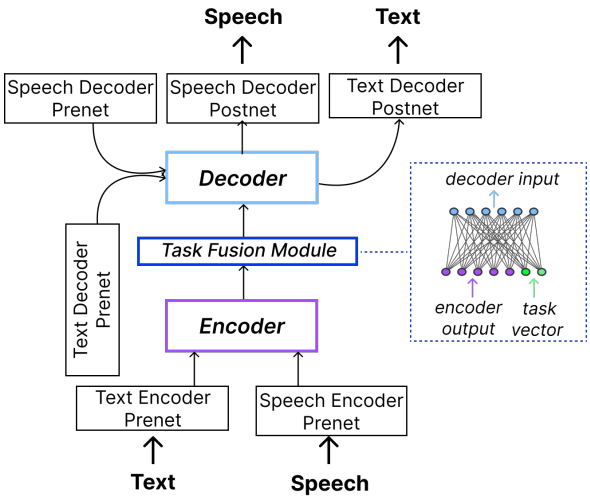

Figure 1: STTATTS architecture: the task fusion module is used to condition the encoder output to a specific task.

3.1 Underlying Architecture

The model is based on the SpeechT5 architecture, which comprises transformer encoder-decoder blocks and some auxiliary modules for modality-specific feature extraction and decoding. Specifically, the modal-specific encoder prenets, and modal-specific decoder pre/postnet modules, are used to handle the text and speech modalities at the input and output, while the main encoder/decoder network processes the unified representations.

Unified Encoder-Decoder. The unified encoder-decoder follows the transformer architecture (Vaswani et al., 2017). The transformer encoder in our base model has 12 blocks, with a model dimension of 768 and an inner feed-forward network dimension of 3072. The decoder comprises 6 transformer decoder blocks with a model dimension of 768 and an inner dimension of 3072. The unified encoder and decoder can process either latent representation of text or speech. We use the pre-trained encoder weights from SpeechT5 and ArTST for English and Arabic experiments, respectively.

Text Encoder and Decoder Pre/Post-Nets. The text encoder pre-net and text decoder pre/post-nets use shared embeddings. The pre-nets transform a token in the sequence to a 768 embedding vector. The text decoder post-net projects the decoder's hidden state into the probability distribution of tokens, normalized by the softmax function.

Speech Encoder and Decoder Pre/Post-Nets. The speech encoder pre-net is a convolutional feature extraction model with 6 1-dimensional convolutional layers, and GELU activation following wav2vec2.0 (Baevski et al., 2020). The output of the convolutional network is normalized before passing to a linear layer to up-sample from 512 to 768. The speech decoder pre-net upsamples the mel spectrogram to a 768-dimensional vector. It consists of two sequential layers of Linear transformations with ReLU activations, followed by an additional Linear layer that upsamples the output from the previous layers. To enable multi-speaker TTS, the speaker embedding vector of the target speaker extracted using x-vectors (Snyder et al., 2018) is concatenated to the output of the speech decoder pre-net before being down-sampled with a linear layer to the decoder hidden size followed by RELU activation.

The speech decoder post-net utilizes a linear layer to predict the log mel-filterbank from the decoder output and five 1-dimensional convolutional layers to generate a residual for enhancing the predicted mel (Ao et al., 2022). Another linear module is incorporated to transform the decoder output into a scalar to predict the stop token. Ao et al.'s (2022) HiFi-GAN vocoder is used to synthesize speech from the generated mel spectrogram.

3.2 Task Fusion Module

Each task is represented with a 128-dimensional vector, which is concatenated with the encoder's output, followed by a fully connected layer (See Figure 1) to project the learned representation back to the encoder embedding size. The output of this module is used as input to the decoder.

3.3 Multi-Task Loss Objective

For ASR, we use $\mathcal{L}_{ce}$: the cross-entropy loss of the decoder and $\mathcal{L}_{ctc}$: the standard CTC loss from ESPNet (Watanabe et al., 2018).

$$\mathcal{L}_{asr} = \mathcal{L}_{ce} + \mathcal{L}_{ctc} \quad (1)$$

For optimizing speech synthesis, we use the $\mathcal{L}_1$ loss to minimize the distance between the target and generated mel spectrograms; the binary cross entropy $\mathcal{L}^s_{bce}$ loss is used to predict the stop token for generation; and the guided attention loss is used to speed up training convergence for speech synthesis as described by Tachibana et al. (2018). The latter was added to speed up the training of TTS (Ao et al., 2022) which is typically a lot slower

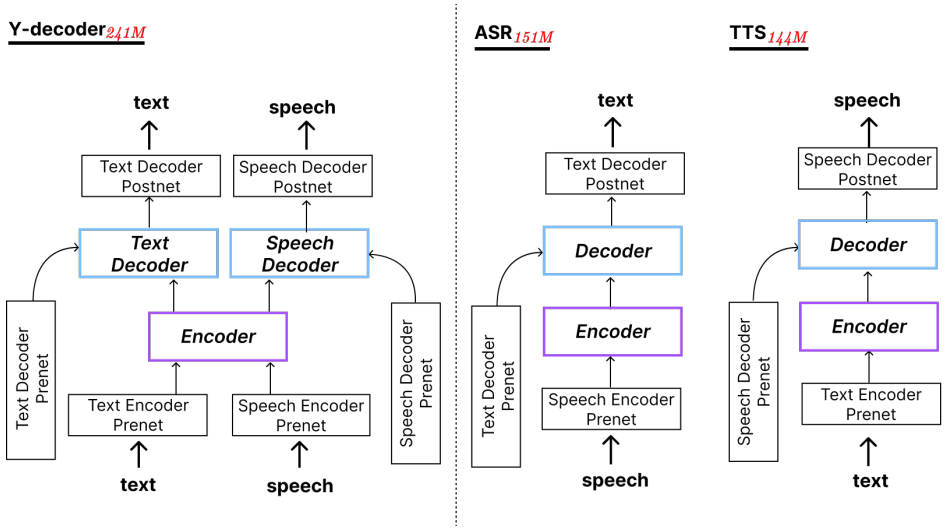

Figure 2: *Left:* Other modifications we experimented with. *Right:* Architecture for single task training using the SpeechT5/ArTST methodology.

compared to ASR.

$$\mathcal{L}_{tts} = \mathcal{L}_1 + \mathcal{L}_{bce} + \mathcal{L}_{attn} \quad (2)$$

For joint training, similar convergence rates enable better and consistent training. The ASR and TTS objectives are combined as $\mathcal{L} = \mathcal{L}_{asr} + \mathcal{L}_{tts}$. At each step, we calculate the loss for each task and normalize by the number of samples per task in that step. The model is updated after some k gradient accumulation steps.

Input/Output Representation. For speech, we use the raw waveform sampled at 16kHz as input and the 80-dimensional log mel-filterbank features as target output for all experiments. We trained with a maximum input token size of 480K, which corresponds to 30 seconds of speech. For text, character-level tokens served as input and output for all experiments. The maximum input token length is 600. Our vocabulary size is 98 for experiments using Arabic following ArTST (Toyin et al., 2023) and 84 for English following SpeechT5 (Ao et al., 2022).

4 Experimental Settings

We conducted experiments across two languages: English and Arabic using open-source datasets. English was used as a resource-rich language, and trained using the benchmark LibriSpeech dataset; this enables the utilization of pre-trained models from SpeechT5 (Ao et al., 2022), and a direct comparison with their downstream models. Arabic served as a relatively low-resource language, mainly because of the shortage of clean speech data that can be utilized for training high-quality TTS systems (Kulkarni et al., 2023). This also enables using the pre-trained checkpoints from ArTST (Toyin et al., 2023).

4.1 Datasets

English. We used LibriSpeech (Panayotov et al., 2015), referred to as `LS` for ASR. For TTS, we use LibriTTS (Zen et al., 2019), referred to as `Ltts`, along with LJSpeech (Ito and Johnson, 2017). We conducted experiments with varying training data sizes to evaluate the impact of data size on performance. See Table 1 for data combinations.

	LS			Ltts			LJSpeech
	100hr	360hr	500hr	100hr	360hr	500hr	-
en_s	✓	✗	✗	✓	✓	✗	✗
en_m	✓	✓	✗	✓	✓	✓	✗
en_l	✓	✓	✓	✓	✓	✓	✓

Table 1: Datasets for English experiments.

Arabic. We combined quality data for speech synthesis from two publicly available datasets: Arabic Speech Corpus (ASC) (Halabi, 2016), and Classical Arabic Text-to-Speech Corpus (ClArTTS) (Kulkarni et al., 2023) for both ASR and TTS. Since TTS quality is highly sensitive to the quality and consistency of the audios and annotations used for training, we did not utilize other large speech

data for joint training [2]. This also serves as a good setting for validating the approach on low-resource settings. We combined the train and test sets of both datasets. However, the original transcripts for the ASC corpus were modified to match the phones (Halabi, 2016), leading to the removal/addition of characters that are silenced/pronounced. These modifications negatively impact both ASR and TTS performance in our set-up as they are inconsistent with the transcripts used in ClArTTS, which follows standard Arabic spelling conventions. To rectify this issue, we restored the transcripts to standard Arabic spelling. We utilized ChatGPT 4.0 to restore the spelling, and manually inspected a subset of the resulting transcripts, which were deemed to be correct by a native Arabic evaluator. We used regex to remove punctuation marks, newline characters, and additional English text from the model's output. The prompt used was: *In the given arabic texts, fix the incorrect characters in the words, take each line as a sentence, and return the same number of lines as passed. Don't remove the diacritics {line separated transcriptions}* .

4.2 Data Preparation

All punctuation marks were removed for both English and Arabic texts and all English characters were converted to lowercase letters. The standardized sampling rate for speech data across all collected datasets was 16 kHz. For Arabic, we trained the model for TTS with and without diacritics.

4.3 Training Details

The pre-trained checkpoints from SpeechT5[3] and ArTST[4] were used as initialization weights for our experiments on English and Arabic, respectively. For reproducibility, we summarize training details per experiment in Table 2. All experiments were carried out on A100 GPUs with 80GB memory. Our training code and checkpoints are available. [5]

Warm Fine-tuning. Toyin et al. (2023) discuss that in the low-resource TTS setting for Arabic, fine-tuning with larger ASR datasets first, followed by continual fine-tuning with high-quality TTS data improved synthesized speech quality. The resulting model achieved higher intelligibility even without the use of diacritics. We incorporate this finding in our Arabic model by first fine-tuning the model for single-task TTS on the MGB2 (Ali et al., 2016) dataset, followed by continual fine-tuning for joint ASR and TTS using our smaller dataset, which improves TTS performance. For English, we utilize a similar approach for a different purpose: In en_l, the sizes of ASR and TTS training sets are imbalanced, which negatively impacts TTS performance. In the first 200k updates, we use the train-other-500 split from LS (Panayotov et al., 2015) for optimizing the ASR part of the joint objective. For the rest of the training duration, we use the rest of the data (the combined LS 100 and LS 360) to continue optimizing ASR in the joint objective. The training data for TTS (see Table 1) remain unchanged throughout training. This approach enables the utilization of the full ASR train set without negatively impacting TTS performance. Ablation results with and without this approach are discussed in Section 6.4.

	Arabic	English ($\mathcal{D}_s^{en}$)
Data	-	-
Total hours	32	291
- ASR	16	100
- TTS	16	191
Character vocabulary size	98	84
Model architecture	-	-
Parameters (M)	155	155
Training Configurations	-	-
Max. input tokens (M)	4.0	3.2
Total updates (K)	80	150
Update frequency (k)	6	8
Learning rate	1e-4	1e-4
LR scheduler	tri-stage	tri-stage
phase ratio	0.25,0.4,0.35	0.25,0.4,0.35
Optimizer	adam	adam
# GPU	3	3

Table 2: Experiment details. For en_m we use 250K training updates and 4 A100 GPU's. For en_l we use 450K training updates and 4 A100 GPU's.

4.4 Evaluation Metrics

ASR. We report the Word and Character Error rates (WER & CER) for evaluating our model's performance for ASR task. For Arabic, the texts are normalized before evaluation by removing diacritics. Diacritics are mostly useful as input for TTS models, but most ASR models are trained and evaluated without diacritics. We transcribe speech using CTC weight of 0.5.

[2] Our preliminary experiments showed that balancing training data for TTS and ASR are needed to achieve balanced outcomes on both tasks. As TTS data are limited for Arabic, we limited the data used for ASR training to achieve that balance.

[3] https://github.com/microsoft/SpeechT5/tree/main/SpeechT5

[4] https://github.com/mbzuai-nlp/ArTST

[5] https://github.com/mbzuai-nlp/sttatts

TTS. We employ objective metrics to evaluate our model's synthesized speech intelligibility. We use Whisper's (Radford et al., 2022) large model to transcribe synthesized speech and calculate CER of transcribed speech. Subjective Mean Opinion Score (MOS) was used to measure naturalness and intelligibility for the main results, in addition to predicted MOS values using Wav2Vec2.0 as described in Andreev et al. (2023). Native speakers of both languages rated 5 random samples of synthesized speech on a scale of 1 to 5, with a step of 0.5.

5 Results

In this section, we report our model's performance, comparison against SpeechT5 single-task models (see Figure 2) and comparison against a baseline joint-task model, VoxLM. Table 3 shows the results for Arabic and English with various scales of training data. The performance improves with additional training data, approaching state-of-the-art results on the LibriSpeech test set. On Arabic, the high WER is attributed to the low-resource setting as it was trained with only 16 hours for ASR. Examples of ASR predictions from the Arabic model are shown in Figure 3.

Data	ASR		TTS			
	WER ↓	CER ↓	CER ↓	Naturalness ↑	Intelligibility ↑	WV-MOS ↑
ar	10.22	2.63	6.22	3.28	2.78	3.69
en_s	4.84	1.63	3.18	3.36	4.00	**4.40**
en_m	3.47	1.07	2.74	3.20	4.04	4.24
en_l	**2.99**	**0.90**	**2.10**	3.00	**4.38**	4.26

Table 3: Evaluation of ASR performance using WER and CER on LibriSpeech test set, and TTS performance using CER, Naturalness MOS, Intelligibility MOS, and MOS scores predicted by Wav2Vec2.0 (WV-MOS).

5.1 Comparison with Single-Task Models

In Table 4, we compare our model with single task models from ArTST (Toyin et al., 2023) and SpeechT5 (Ao et al., 2022), which have the same pre-trained checkpoint and fine-tuning data, so they are directly comparable: we used the en_s described in Table 1 for English and ar. STTATTS achieves comparable performance to the single-task models, demonstrating the effectiveness of the proposed multi-task methodology. Compared to the large-scale multi-lingual model, Whisper, STTATTS performs better than Whisper$_{small}$, in spite of having smaller number of parameters and being trained on a much smaller data set. In English, Whisper$_{large}$ performs marginally better than STTATTS trained on en_s. For Arabic, STTATTS performs substantially better than both small and large variants.

Model	# params(M)	ASR	TTS
		WER ↓	CER ↓
Arabic			
*ArTST$_{ASR}$ (Toyin et al., 2023)	151	**7.59**	×
*ArTST$_{TTS}$	145	×	9.61
*Whisper$_{small}$ (Radford et al., 2022)	244	32.2	×
*Whisper$_{large}$	1550	23.4	×
STTATTS	155	10.22	**6.22**
English en_s			
†SpeechT5$_{ASR}$ (Ao et al., 2022)	151	4.4	×
††SpeechT5$_{TTS}$	145	×	6.3
†Whisper$_{small}$	244	3.4	×
†Whisper$_{large}$	1550	**3.0**	×
STTATTS	155	4.8	**3.2**

Table 4: Comparison of STTATTS with single task models on our test sets. Whisper models use 438K hours of English ASR data, and SpeechT5 uses 100 hours LibriSpeech for ASR and 460 hours LibriTTS for TTS. For ArTST results, we train the model with our combined dataset and report evaluation results on our test set. * indicates we trained the models and evaluated ourselves, † indicates results as reported in the corresponding paper, †† indicates the evaluation of model on randomly selected synthesized text as ours, × indicates the models cannot perform the corresponding task.

5.2 Comparison with Joint-Task Models

We compare our model's performance with reported joint models that perform both TTS and ASR in Table 5. In particular, we use VoxtLM (Maiti et al., 2024) as it is publicly available for direct comparison, while other models are not currently available to perform comparable evaluation. Using the same training data, STTATTS performs better in both tasks, even when compared with the large VoxtLM model of 1.3B parameters. Compared to the VoxtLM base, which has a comparable number of parameters, STTATTS performs on a par with a fraction of the training data and is far better when an almost equal amount of data is used.

6 Ablations & Analysis

6.1 Task Scaling

To evaluate whether the model can be scaled to perform additional speech/text to speech/text tasks, we experiment with adding Voice Conversion in the en_m setting. We use the CMU Arctic (Kominek and Black, 2004) dataset, and optimize voice con-

Model	# params	Train Data(hrs)	ASR	TTS
		ASR/TTS	WER↓	CER↓
English				
VoxtLM_base† 2024	350M	0.9K/0.5K	6.5	3.5
VoxtLM_large†	1.3B	0.9K/0.5K	4.6	3.9
VoxtLM_large†	1.3B	45K/0.6K	2.7	3.6
STTATTS(en_s)	154M	0.1K/0.2K	4.8	3.2
STTATTS(en_l)	154M	0.9K/0.5K	**3.0**	**2.1**

Table 5: Comparison with other joint-task models. † indicates results as reported in the reference paper. ASR results are on the Librispeech test-clean set for all models. TTS results are on 100 subset from the LibriTTS test set, we can't compare with the same samples since the samples used for testing in VoxtLM are not publicly available.

version using the same loss function used for TTS (see Equation 3.3). We evaluate VC performance using CER and Speaker Similarity (SS) using the cosine similarity function on speaker embedding (Snyder et al., 2018) extracted from the speech. The results are shown in Table 6. Although there's a slight reduction in WER for the ASR task, TTS MOS score has improved, possibly due to the shared output space and loss function. We achieve speaker similairy score of 0.99, and a low CER, demonstrating the high quality in the additional VC task without any additional parameters added to the model.

	ASR		TTS		VC	
Data	WER	CER	CER	WV-MOS	CER	SS
en_m	3.59	1.13	2.83	4.28	1.58	0.99

Table 6: Results for joint training of ASR,TTS and VC.

6.2 Architectural Variations

Y-Decoder. **Y** in **Y**-decoder stands for the desired output: either text or speech. Here, we use a similar approach to STTATTS but with modal-specific decoders, i.e., a separate text decoder and a speech decoder, while sharing the same encoder. In this model, no task fusion module is used since each task is parameterized by its own standalone decoder. This results in a larger model than STTATTS but still smaller than the disjoint ASR and TTS SpeechT5 models. See Figure 2 (left) for an illustration of Y-decoder architecture.

Multi-Stage Training. We also experimented with multi-stage training, where we alternate updating the weights for a specific task with some model components frozen/unfrozen. We performed two experiments using STTATTS, where we update the parameters of the ASR first, followed by TTS, or vice versa, with the respective single-task loss function in each stage. For example, if we follow an ASR-first schedule, we first update the auxiliary weights, the encoder, and the decoder for the ASR task, then in the second stage, we keep the encoder and decoder frozen and update the auxiliary weights for the TTS task.

Task-Specific Adapters. We experimented with a shared transformer encoder-decoder backbone using pre-trained weights with adapters (Houlsby et al., 2019) as an alternative architecture. We tried using the base model architecture offline and only updating the weights of an adapter for each task. We added a 64 inner dimension adapter to the decoder, but this approach didn't yield good results. For ASR, the WER was 150% and the TTS loss converged at a high value (≈ 0.8) relative to STTATTS (≈ 0.4).

Results. Evaluation results are shown in Table 7. While performance on ASR is comparable across models and slightly better with the Y-decoder architecture, STTATTS is the best overall in terms of balancing performance across multiple tasks while maintaining a lower number of parameters. With multi-stage training, we find that it works better when we update the encoder weights for the ASR task first. We observe TTS performance is good whether the transformer encoder is optimized for ASR or TTS first, but the performance for ASR significantly degrades with $\times 10$ difference between the WER for ASR-first and TTS-first approaches. This may be attributed to the dimensionality of the input features for each task, as ASR requires more computations to process the input in the encoder, while TTS works with discrete text input.

6.3 Effect of Data Imbalance

For en_s, we first fine-tuned with LS-100 and Ltts-100 (which contains ≈ 58 hours of speech). This data combination resulted in less intelligible and robotic synthesized speech (MOS of 1.5). Interestingly, the ASR performance is not affected when we have more TTS data, as shown for en_s in Table 3; on the contrary, the results improved from 5.61 to 4.84. As a result, and since TTS data are generally more scarce, our main experimental settings

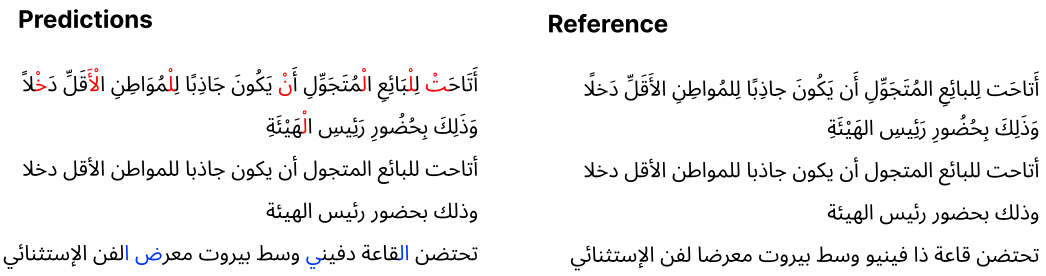

Figure 3: Sample ASR predictions from the Arabic STTATTS model. Diacritic Errors Character Errors

Model	# params(M)	ASR		TTS	
		WER↓	CER↓	CER↓	WV-MOS↑
Arabic					
Multistage					
TTS-first	155	109.94	78.47	9.61	3.70
ASR-first	155	11.60	7.80	67.87	2.79
Y-decoder (enc - 2× dec)	211	10.37	2.78	8.31	3.68
STTATTS	155	10.22	**2.63**	**6.22**	**3.69**
English en_s					
Y-decoder (enc - 2× dec)	211	5.67	1.91	4.36	4.45
STTATTS	155	**4.84**	**1.63**	**3.18**	**4.40**

Table 7: Results from other architectural variation experiments. For *Multistage*, we only show our preliminary results for Arabic since performance was sub-optimal.

are all conducted with ASR data downsampled to match the size of the TTS training set.

6.4 Effect of Warm Fine-Tuning

We compare STTATTS's performance with and without the warm fine-tuning approach introduced described in section 6.4. The results are shown in Table 8. Except for the Arabic ASR performance that is degraded by ≈ 2% absolute WER, we find that this approach results in improved performance.

Data	Warm ft	ASR		TTS	
		WER ↓	CER ↓	CER ↓	WV-MOS ↑
ar	✗	8.61	5.60	9.94	3.61
ar	✓	10.22	**2.63**	**6.22**	**3.69**
en_l	✗	3.08	0.96	3.28	4.24
en_l	✓	**2.99**	**0.90**	**2.10**	**4.26**

Table 8: Results of experimenting with warm fine-tuning (ft) and without.

6.5 Diacritization in Arabic

TTS models for Arabic are typically trained with full diacritics (Kulkarni et al., 2023). This is because diacritics contain essential information about most vowels, without which the text is highly ambiguous. However, Toyin et al. (2023) demonstrated good TTS performance without the inclusion of diacritics, which is mainly attributed to the warm fine-tuning they perform with ASR data. As the TTS data include diacritics, we performed experiments where we train models with and without diacritics. We evaluate ASR on normalized text where all diacritics are removed. The results are reported in Table 9. We notice that performance in TTS is in fact better without diacritics using this model. This surprising observation may be attributed to the fact that ArTST(Toyin et al., 2023) was pre-trained without diacritics, so adding diacritics in the fine-tuning stage with small data size may not be sufficient. However, as shown in the examples in Figure 3, we note that ASR transcription with diacritics (first line) is in fact correct, even though it does not match exactly the reference, which is mainly a result of the *sukoon* diacritic that is often omitted in the reference.

Data	Diacritics	ASR		TTS	
		WER ↓	CER ↓	CER ↓	WV-MOS ↑
ar	✓	10.10	2.75	7.40	3.68
ar	✗	10.22	2.63	6.22	3.69

Table 9: Results with and without diacritics.

6.6 Effect of Task-Fusion Module Position

We experimented with having the task fusion module before the encoder with the aim of guiding latent feature extraction based on the given output. This approach performs fairly well for ASR (CER 4%) but fails for TTS with CER of 80%.

6.7 Effect of Pre-Trained Weights

The results above are all reported with pre-trained weights from SpeechT5 and ArTST pre-trained checkpoints. We examined the effect on performance with and without starting with these pre-

trained weights. See Figure 4 for the learning curve in terms of loss reduction during training. We see that using pre-trained weights is beneficial for maximizing performance. Pre-training is particularly crucial for TTS tasks, as starting from scratch results in 10-fold degradation in CER. On the other hand, the performance for ASR is far less affected (only +1% increase in CER value for STTATTS, and a larger increase for Y-decoder). Overall, starting with pre-trained weights is crucial in downstream tasks for all model variations.

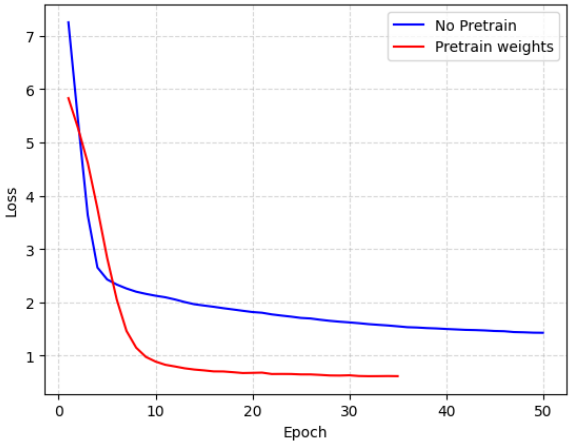

Figure 4: Comparison of training objective with and without using pre-trained weight.

7 Discussion

We described our experiments of jointly learning Speech-to-Text and Text-to-Speech models based on the SpeechT5 architecture, resulting in a truly multi-modal and functional model, that accepts both speech and text as input and output. We experimented with both Arabic and English languages, with Arabic being a relatively low-resource language due to limited amounts of data available for TTS training. Our results show that while it is possible to train models using our framework with less than 20 hours of speech in total, more data is always better for maximizing performance in both ASR and TTS tasks. Our results are the first to report multi-task ASR/TTS in the Arabic language, showing promising results in low-resource settings, and potential for improvement with additional data. For English, a few other models have been recently proposed; our comparative analysis with the only publicly available model of this variety, named VoxtLM (Maiti et al., 2024), favors STTATTS in both performance and parameter efficiency. We experimented with different parameter-efficient approaches to jointly learn ASR and TTS tasks and we find using the task-fusion module strikes a perfect balance in performance between both tasks with the least amount of parameters. It's also worth noting that the task-fusion module makes incorporating more tasks and output modalities feasible as the module aligns latent representation to match the desired output modality. Furthermore, the integrated multi-task approach, in addition to being more efficient in model size, is more efficient in training as it requires fewer updates in total. Future work will explore the possibility of integrating additional text output tasks within the framework and improving synthesized speech naturalness.

We believe that the proposed model, being trained on publicly available data with the code and checkpoints publicly available[6], can serve as a strong baseline for multi-task speech processing.

Limitations. Our experiments focus on three key aspects: language generalization, scalability (in terms of training data requirements and tasks), and parameter efficiency. Although we explored two languages separately, we did not experiment with a joint multilingual models. Additionally, we used VoxLM as our only baseline for multi-task models. While other models have recently been proposed, their code and models are not yet publicly available for comparison. We did not explore increasing model size, as it would require pre-training from scratch, which is computationally expensive. However, slightly larger models could potentially enhance performance in the multi-task setting by better embedding the diverse input and output modalities. Finally, we found that subjective MOS evaluation was rather difficult to conduct as most outputs were intelligible but somewhat noisy and unnatural. For Arabic, the small data size and lack of diacritics does result in degraded intelligibility due to mismatched pronunciation of short vowels. Therefore, the reported TTS results provide some signal of quality, but may not be informative of the actual quality of the synthesized speech.

Acknowledgements

This work was funded in part by a Google research award, awarded in November 2023.

[6] https://github.com/mbzuai-nlp/sttatts

References

Ahmed Ali, Peter Bell, James Glass, Yacine Messaoui, Hamdy Mubarak, Steve Renals, and Yifan Zhang. 2016. The mgb-2 challenge: Arabic multi-dialect broadcast media recognition. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 279–284. IEEE.

Pavel Andreev, Aibek Alanov, Oleg Ivanov, and Dmitry Vetrov. 2023. Hifi++: A unified framework for bandwidth extension and speech enhancement. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.

Junyi Ao, Rui Wang, Long Zhou, Chengyi Wang, Shuo Ren, Yu Wu, Shujie Liu, Tom Ko, Qing Li, Yu Zhang, Zhihua Wei, Yao Qian, Jinyu Li, and Furu Wei. 2022. Speecht5: Unified-modal encoder-decoder pre-training for spoken language processing. *Preprint*, arXiv:2110.07205.

Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *CoRR*, abs/2006.11477.

Ankur Bapna, Yu an Chung, Nan Wu, Anmol Gulati, Ye Jia, Jonathan H. Clark, Melvin Johnson, Jason Riesa, Alexis Conneau, and Yu Zhang. 2021. Slam: A unified encoder for speech and language modeling via speech-text joint pre-training. *Preprint*, arXiv:2110.10329.

Yu-An Chung, Yu Zhang, Wei Han, Chung-Cheng Chiu, James Qin, Ruoming Pang, and Yonghui Wu. 2021. W2v-bert: Combining contrastive learning and masked language modeling for self-supervised speech pre-training. *Preprint*, arXiv:2108.06209.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

Nawar Halabi. 2016. *Modern standard Arabic phonetics for speech synthesis*. Ph.D. thesis, University of Southampton.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. 2021. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3451–3460.

Keith Ito and Linda Johnson. 2017. The lj speech dataset. `https://keithito.com/LJ-Speech-Dataset/`.

John Kominek and Alan W. Black. 2004. The cmu arctic speech databases. In *5th ISCA Workshop on Speech Synthesis (SSW 5)*, pages 223–224.

Ajinkya Kulkarni, Atharva Kulkarni, Sara Abedalmonem Mohammad Shatnawi, and Hanan Aldarmaki. 2023. Clartts: An open-source classical arabic text-to-speech corpus. *Preprint*, arXiv:2303.00069.

Soumi Maiti, Yifan Peng, Shukjae Choi, Jee weon Jung, Xuankai Chang, and Shinji Watanabe. 2024. Voxtlm: unified decoder-only models for consolidating speech recognition/synthesis and speech/text continuation tasks. *Preprint*, arXiv:2309.07937.

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

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Robust speech recognition via large-scale weak supervision. *Preprint*, arXiv:2212.04356.

David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. 2018. X-vectors: Robust dnn embeddings for speaker recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5329–5333.

Hideyuki Tachibana, Katsuya Uenoyama, and Shunsuke Aihara. 2018. Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4784–4788.

Hawau Toyin, Amirbek Djanibekov, Ajinkya Kulkarni, and Hanan Aldarmaki. 2023. ArTST: Arabic text and speech transformer. In *Proceedings of ArabicNLP 2023*, pages 41–51, Singapore (Hybrid). Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Tianrui Wang, Long Zhou, Ziqiang Zhang, Yu Wu, Shujie Liu, Yashesh Gaur, Zhuo Chen, Jinyu Li, and Furu Wei. 2023. Viola: Unified codec language models for speech recognition, synthesis, and translation. *Preprint*, arXiv:2305.16107.

Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, Adithya Renduchintala, and Tsubasa

Ochiai. 2018. Espnet: End-to-end speech processing toolkit. *CoRR*, abs/1804.00015.

Heiga Zen, Viet Dang, Rob Clark, Yu Zhang, Ron J. Weiss, Ye Jia, Zhifeng Chen, and Yonghui Wu. 2019. Libritts: A corpus derived from librispeech for text-to-speech. *Preprint*, arXiv:1904.02882.

Dong Zhang, Shimin Li, Xin Zhang, Jun Zhan, Pengyu Wang, Yaqian Zhou, and Xipeng Qiu. 2023a. Speechgpt: Empowering large language models with intrinsic cross-modal conversational abilities. *Preprint*, arXiv:2305.11000.

Yu Zhang, Wei Han, James Qin, Yongqiang Wang, Ankur Bapna, Zhehuai Chen, Nanxin Chen, Bo Li, Vera Axelrod, Gary Wang, Zhong Meng, Ke Hu, Andrew Rosenberg, Rohit Prabhavalkar, Daniel S. Park, Parisa Haghani, Jason Riesa, Ginger Perng, Hagen Soltau, Trevor Strohman, Bhuvana Ramabhadran, Tara Sainath, Pedro Moreno, Chung-Cheng Chiu, Johan Schalkwyk, Françoise Beaufays, and Yonghui Wu. 2023b. Google usm: Scaling automatic speech recognition beyond 100 languages. *Preprint*, arXiv:2303.01037.