tinyDigiClones: A Multi-Modal LLM-Based Framework for Edge-optimized Personalized Avatars

Abdul Basit, Muhammad Shafique
eBrain Lab, Division of Engineering, New York University Abu Dhabi (NYUAD), Abu Dhabi, United Arab Emirates
abdul.basit@nyu.edu, muhammad.shafique@nyu.edu

2024 International Joint Conference on Neural Networks (IJCNN) | 979-8-3503-5931-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/IJCNN60899.2024.10649909

***Abstract*—Conversational AI has made significant strides, however the integration of multi-modal interactions, particularly on edge devices, presents a novel frontier to be explored primarily due to the computational constraints. This paper proposes the tinyDigiClones framework, which enables communication with a personalized AI assistant leveraging optimized large language models (LLMs) for natural language processing (NLP), and deep-learning models for automatic speech recognition (ASR) and realistic voice synthesis. This paper explores various options for different AI models employed in our framework, with a primary focus on ensuring efficient deployment on edge devices while maintaining high accuracy. To replicate users' voice fonts and learn the unique vocal characteristics, the Text-to-Speech (TTS) models are trained using a custom dataset of audio-text pairs. It is generated automatically by the ASR module which segments extended sentences into shorter, transcript-matched audio files. Moreover, deploying state-of-the-art LLMs on resource-constrained devices presents a significant challenge, particularly in maintaining minimal latency, given their extensive parameter counts. Towards this, we explore several lightweight LLMs and employ optimization techniques aimed at reducing computational costs. The integration of these models is personified through a digital avatar mirroring the user's facial and voice likeness, offering an immersive experience. Deployment on the edge alleviates the server latency and enhances privacy enabling real-time interaction capabilities of AI chatbots, ideal for interactive digital avatars.**

***Index Terms*—Personalized Digital Avatars, Large Language Models, Automatic Speech Recognition, Text-to-speech Synthesis**

I. Introduction

Conversational AI has significantly impacted various sectors, from customer service to personal assistants. The key research problem addressed in this paper is the transition from text-based to multi-modal conversational interfaces. The evolving market trends and user behavior studies highlight a growing preference for voice based mediums [1], [2], valued for their convenience and intuitive interaction. This shift underscores the need for an integrated system that melds text-based chatbot functionality representation with voice and facial interaction. According to this survey [3] current technologies fall short in offering such immersive communication experiences with chatbots. Our framework focuses on this need, particularly optimizing for edge deployment to seamlessly integrate these user-preferred modalities, addressing the gap and advancing towards more realistic user interactions. Recognizing the pioneering efforts of Nvidia's Omniverse [4] and Facebook's Metaverse [5] in the domain of digital innovation serves as a significant motivation for our research. Their active pursuit in crafting immersive digital identities with a strong emphasis on realistic visual fidelity highlights the vast potential in this field.

The domain of Large Language Models (LLMs) has undergone remarkable growth and progress, reflecting advancements in natural language processing. Notably, models such as by GPT-4 [6], PaLM [7], and Falcon [8] etc. have evolved to encompass billions to trillions of parameters, achieving unprecedented depth in language comprehension and generation. This evolution is highlighted by some prominent LLMs based on Open-LLM leaderboard [9], illustrated in Figure 1.

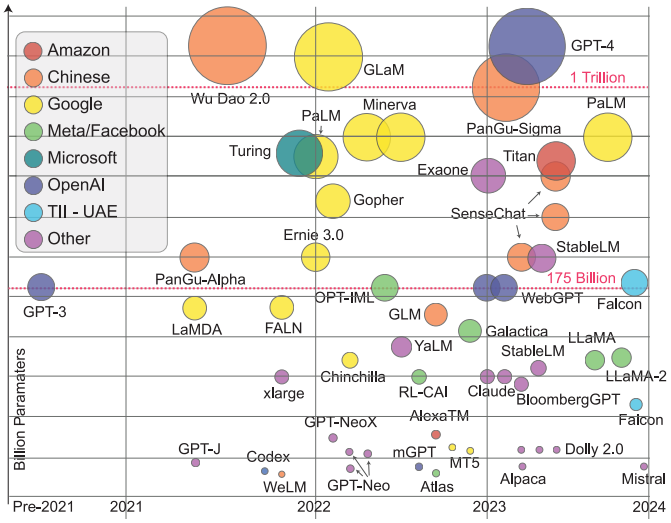

Fig. 1: Developments in LLMs and their parameter count based on OpenLLM leader-board reflecting quantitative increase in model sizes and a qualitative leap in their abilities. This may not be a comprehensive list amassing all models, but it highlights the ground-breaking developments in LLMs.

Notable works have demonstrated the effectiveness of LLMs like OpenAI's GPT-4 for text-based interactions [6], while systems like Google's Duplex [10] show promise in voice-based tasks. However, these solutions often operate in silos, leading to a disjointed user experience. Their major pros include advanced natural language understanding and generation, but they fall short in creating a cohesive multi-modal digital identity. Moreover, a significant drawback of contemporary state-of-the-art models lies in their server-based operational framework. This server dependency raises substantial concerns regarding data privacy [11], as sensitive

information is often transmitted and processed remotely, potentially exposing it to security vulnerabilities [12], [13], thereby requiring edge-based deployment to alleviate these concerns. However, adapting these complex models for edge device deployment presents a considerable challenge due to their constrained computational resources, making it difficult to replicate the same level of performance and efficiency as experienced in server-based environments [14]. This gap necessitates significant optimization to ensure that advanced AI capabilities are effectively accessible on edge platforms. To address these research challenges, we present *tinyDigiClones* with the following novel contributions (overview in Figure 2).

- The multi-modal framework proposes an end-to-end pipeline which combines voice and facial interaction through the integration of Automatic Speech Recognition (ASR) Models, Large Language Models (LLMs), Speech-to-text (STT) Models, and facial animation model offering a user-friendly digital communication experience by encompassing both auditory and visual aspects of human interaction. [Section III].
- Our Speech Synthesis system leverages TTS model to produce high-fidelity audio output, trained to replicate the user's voice likeness and inflection patterns, providing a highly personalized vocal experience [Section III-C].
- The framework optimizes deep-learning models to align with the user requirements and memory constraints of the target device. It employs a discrete set of models, each with varying complexity and resource requirements. Through Minimum Remaining Values (MRV) and Least Constraining Value (LCV) heuristics, the framework effectively identifies the best configuration, delivering efficient performance for a specific device [Section III-D].

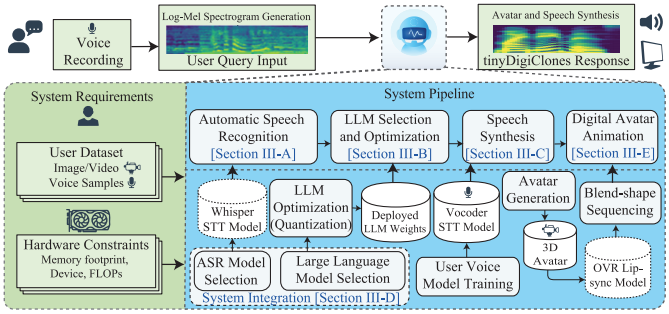

Fig. 2: The tinyDigiClones overview highlighting the system requirements and pipeline to generate the output response in the form of a digital avatar.

Paper Organization: Section II discusses state-of-the-art. Section III presents the tinyDigiClones system in further detail. The experimental setup and results are presented in Section IV.

II. Background and Related Work

A. Automatic Speech Recognition Models

The advancements in automatic speech recognition (ASR) systems have been significant, marked by remarkable improvements in transcription and translation accuracy, largely driven by deep-learning algorithms. Google has contributed to this progress with its BERT-based model [15] and it has shown promising results on the AISHELL dataset [16], especially when paired with a simple acoustic model. Moreover, OpenAI has introduced Whisper [17], an ASR model known for its robustness and versatility across various languages and accents. Whisper represents a significant leap forward in the field, demonstrating the capability to handle complex speech recognition tasks with high accuracy. Conformer-1, by AssemblyAI [18], stands out for its robust performance on real-world data, including noisy environments, demonstrating fewer errors compared to other popular models. Additionally, the w2v-BERT framework [19] innovates in self-supervised speech representation learning by combining contrastive learning with Masked Language Modelling (MLM) for end-to-end optimization, achieving significant Word Error Rate (WER) reductions and outperforming existing models like conformer-based wav2vec 2.0 and HuBERT [20] in key benchmarks. These developments collectively represent significant strides in ASR, showcasing versatility, accuracy, and adaptability to diverse linguistic contexts.

B. Large Language Models

Conversational bots and virtual agents proficient in NLP owe a significant part of their evolution to the advancements in LLMs like GPT [6], LLaMa-1&2 [21], [22], BLOOM, [23], Falcon [24] and many such emerging LLMs in the domain. These LLMs are the driving force behind the sophisticated capabilities of chatbots, offering unparalleled opportunities for interaction [25]–[27]. Meta AI's Mistral [28] represents a significant shift towards efficiency in language models. While its parameter count is not as expansive as GPT-4, Mistral focuses on reducing the environmental impact of training and operating LLMs, making it a noteworthy mention for applications where efficiency is paramount and outperforms other LLMs of similar scale.

The development of these LLMs, achieved through extensive pre-training on self-supervised data and fine-tuning via techniques such as Reinforcement Learning with Human Feedback (RLHF) [29], has been instrumental in enhancing the linguistic understanding and responsiveness of chatbots. The integration of these LLMs into digital avatars and edge devices presents an exciting avenue for research and development. The computational demands of LLMs necessitate innovations in model compression and optimization techniques [30] to facilitate their deployment on edge devices. Techniques such as model pruning, quantization, and knowledge distillation are crucial for enabling these powerful models to operate in resource-constrained environments.

C. Speech Synthesis Models

Recent innovations in voice synthesis technologies have introduced models such as Microsoft's VALL-E X [31] which excels in multilingual text-to-speech synthesis and zero-shot voice cloning with emotion-infused speech. Meta's Voicebox [32], a versatile generative model, supports speech synthesis in six languages and can perform editing, noise removal, and

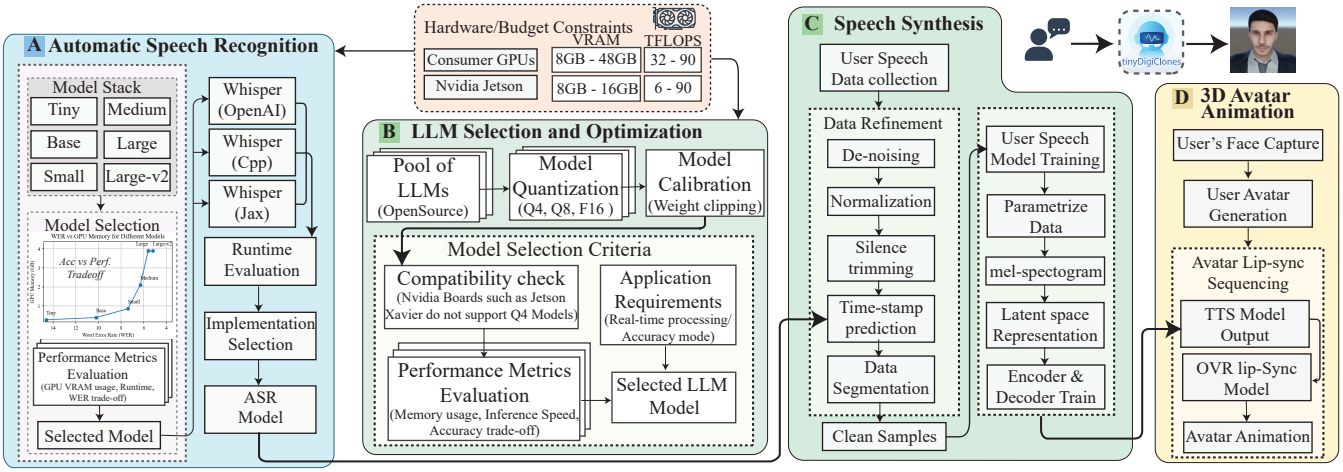

Fig. 3: An Overview of tinyDigiClones Framework: This diagram illustrates the integration of ASR, LLM, and Speech Synthesis with the digital avatar. (a) The choice of ASR model is dependent on the hardware's capabilities, ensuring optimal performance. (b) The LLMs are selected based on the device and application requirements, and optimized model is deployed. (c) The speech synthesis allows for user's voice to be cloned and is trained on user's speech dataset to learn unique vocal characteristics. (d) The systems is complemented by 3D avatar animation for a realistic virtual experience.

style transfer across languages. Collabora's WhisperSpeech [33], an open-source model, focuses on natural-sounding, multilingual speech suitable for various applications including announcements and interactive voice response systems. Glow-TTS [34] is a flow-based generative TTS model, enabling robust, fast, and diverse speech synthesis without needing an external text to voice aligner. The NaturalSpeech [35] system uses a variational autoencoder (VAE) for end-to-end text-to-waveform generation, incorporating several key modules like phoneme pre-training and bidirectional prior/posterior modeling. Notably, on the LJSpeech dataset [36], NaturalSpeech attains a top Mean Opinion Score (MOS) of 4.56. These models collectively push the boundaries of voice synthesis, offering advanced features like voice cloning, style transfer, and efficient generation capabilities.

III. TINY-DIGI-CLONES FRAMEWORK

The deployment of tinyDigiClones framework on edge devices is motivated by the critical need for real-time processing, privacy considerations, and the desire for uninterrupted service irrespective of network connectivity. As technological capabilities evolve, it is anticipated that such systems will become more advanced and deliver a user experience that is both seamless and highly responsive. Figure 3 presents a detailed depiction of the framework's methodology, illustrating the key functionalities that empower the system.

A. Automatic Speech Recognition Integration

Speech-to-text (STT) technology is a crucial component of modern conversational AI. Table 1, showcasing results on the VCC2018 dataset [37], highlights the enhanced capabilities of the Whisper model, as evidenced by its improved performance across several key metrics. Emphasis is also placed on resource management, targeting minimal memory consumption and run-time. Considering these metrics, the Whisper model is particularly well-suited for integration into the tinyDigiClones framework. It meets the stringent demands for real-time processing and showcases high accuracy. The Whisper model inference architecture is showcased in Figure 4.

TABLE I
ASR MODEL EVALUATION ON VCC2018 DATASET [1]

Model	Version	PCC↑	SRCC↑	KRC↑	MSE↓
Whisper	Tiny	0.7072	0.6881	0.5187	0.0281
	Base	0.7178	0.6951	0.5249	0.0225
	Small	0.7136	0.6906	0.5218	0.0212
	Medium	0.7205	0.6957	0.5267	0.0195
	Large	**0.7274**	**0.7061**	**0.5365**	0.0194
MOSNet	Base	0.5588	0.5159	0.3765	0.5166
TitaNet	Large	0.6933	0.6667	0.5005	0.0160
SpeakerNet	Medium	0.6428	0,6210	0.4598	0.0202
GE2E	Base	0.6118	0.5846	0,4306	0.0193
CLOVA	H/ASP	0.6903	0.6623	0,4966	0.0162
Wav2Vec 2.0	xls-r-300m	0.7090	0.6866	0.5190	0.0153
	x1s-r-1b	0.7140	0.6893	0.5210	0.0268
	x1s-r-2b	0.7014	0.6757	0.5096	0.0159
WavLM	Base-Plus	0.6917	0.6816	0.5122	0.0163
	Large	0.7120	0.7036	0.5316	**0.0151**
HuBERT	Large	0.6692	0.6441	0.4800	0.0170
	xLarge	0.6871	0.6684	0.5012	0.0170

This paper explores various implementations of OpenAI's Whisper [17], implemented in Pytorch (OpenAI) [38], JAX [39] and C++ [40] versions, catering to different operational needs. For our specific requirements, we selected the Whisper C++ implementation, primarily due to its optimization for speed and efficiency. This choice aligns with our objective to achieve high-performance ASR while maintaining resource efficiency on edge devices. In ASR systems, Word Error Rate (WER) is a key evaluation metric. It is calculated by aligning the ASR output with a human transcript and counting deletions, substitutions, and insertions. The WER formula is given by WER $= \frac{S+D+I}{N_{\text{ref}}}$ where S, D, I are the counts of

[1]Pearson correlation coefficient (PCC), Spearman-rank correlation coefficient (SRCC), Kendall rank correlation (KRC), Mean square error (MSE).

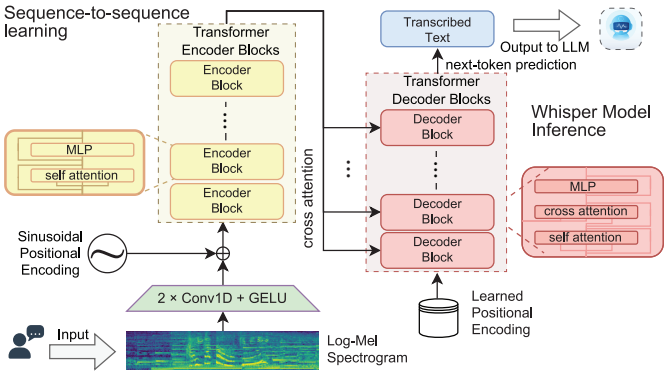

Fig. 4: Whisper, a sequence-to-sequence transformer model trained on several speech processing tasks, including multilingual speech recognition, are jointly represented as a sequence of tokens to be predicted by the decoder.

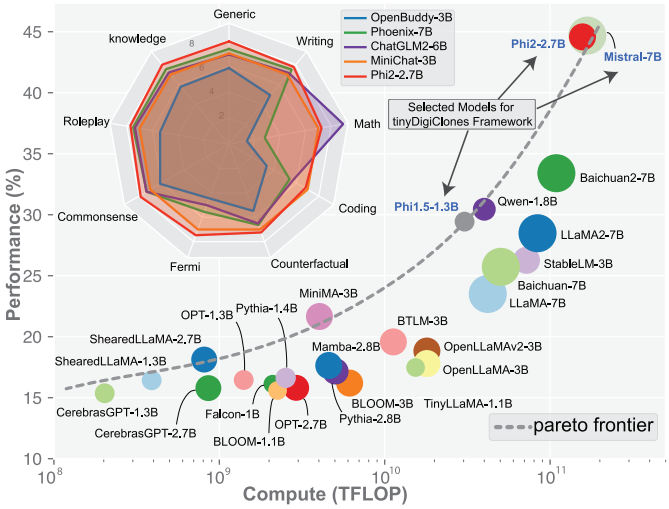

Fig. 5: Highlighting the compute-performance pareto frontier, the scale of each model is represented by the radius of circles. Model performance is quantified across various tasks and performance computed from estimated training resources in TFLOPs. The spider plot compares different baselines on Vicuna-Bench, and Phi2.0 outperforms in several domains.

substitutions, deletions, and insertions, respectively, and N_{ref} is the number of words in the reference transcript.

Table 2 provides a comparison of different Whisper model variants, detailing their parameters, WER on the LibriSpeech [41] test dataset, inference times for 10-second 16-bit audio samples based on brief dialogue scenarios for various implementations, and GPU memory usage reported on Nvidia Jetson Orin Nano. The trade-offs between accuracy (WER), inference speed, and resource requirements is evaluated to design an efficient solution for a particular hardware.

TABLE II
PERFORMANCE METRICS OF WHISPER MODELS

Model	Params	WER	Runtime (OpenAI)	Runtime (cpp)	Runtime (jax)	GPU Memory
Tiny	39 M	14.6	0.236	0.216 s	0.288 s	270 M
Base	74 M	10.2	0.274	0.255 s	0.345 s	388 M
Small	224 M	7.4	0.455	0.425 s	0.492 s	852 M
Medium	769 M	6.3	1.021	0.881 s	1.131 s	2.1 G
Large	1550 M	5.6	1.861	1.462 s	1.885 s	3.9 G
Large-v2	1550 M	5.2	1.788	1.455 s	1.882 s	3.9 G

B. LLM Selection and Optimization

In our exploration of Large Language Models (LLMs) and their optimization, we closely examined the new compute-performance Pareto frontier from the published Open-LLM leader-board [9]. MiniMA-3B [42], an adapted version of LLaMA2-7B [22], exhibits notable performance in the benchmark after being fine-tuned into the instruction-following model MiniChat-3B [42]. However, Phi2.0-2.7B takes the lead due to its superior performance across various benchmarks and outperforming larger models due to the model scaling and training data curation. It efficiently balances the computational demands with performance efficiency ideal for the tinyDigiClones framework. We extended our analysis to light-weight models with around 1B-1.5B parameters from leader-boards, focusing on those best suited for edge deployment. Our evaluation included several key benchmarks:

- **MMLU Accuracy** [43]: Measures knowledge in various subjects in zero-shot and few-shot settings.
- **DROP Score** [44]: Assesses discrete reasoning over paragraphs for deep content understanding.
- **CEval Accuracy** [45]: A Chinese evaluation suite covering diverse disciplines and difficulty levels.
- **GSM8K** [46]: A dataset of grade school math problems emphasizing multi-step mathematical reasoning.
- **BBH Score** [47]: Predicts LLM performance across tasks and models, based on BIG-bench records.
- **HumanEval** [48]: Comprises programming challenges to evaluate complex problem-solving skills.

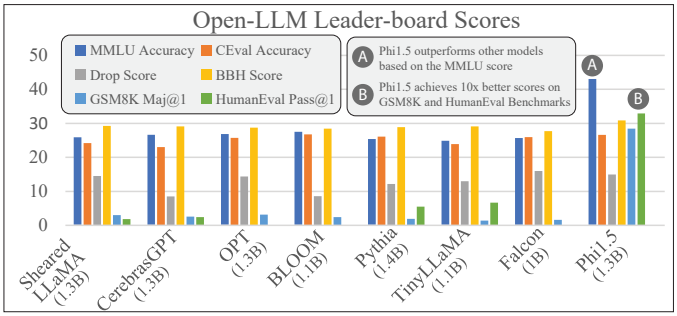

Fig. 6: LLMs with approximately 1B parameters evaluated across diverse benchmarks, with the Phi1.5-1.3B model distinguished by its significantly higher benchmark scores.

Building on the analysis presented in Figure 6, we have identified Phi1.5-1.3B [49] as a particularly promising model for our design scheme. Similarly, inferring from Figure 5, the Mistral-7B [28] model outperforms several other LLMs in performance and is a suitable choice where the accuracy of responses is paramount and the hardware resources allow for its deployment. For the implementation in the tinyDigiClones framework, we have selected Phi1.5 & 2.0, which stand out due to their smaller scale, making them well-suited for deployment on edge devices. Moreover,

Mistral-7B-instructv0.2 is selected based on its efficient trade-off between accuracy and performance.

These LLMs are optimized to achieve high efficiency in resource-constrained environments. Our optimization explores quantization in Q4, Q8, and FP16 formats for efficient deployment. Q4 offers reduced precision and size, Q8 balances performance with accuracy, and FP16 caters to complex tasks with higher computational demands. Post-quantization, models undergo weight clipping for precision retention and performance maintenance. Based on our analysis the model inference time per token and memory consumption for different quantization schemes are highlighted in Figure 7.

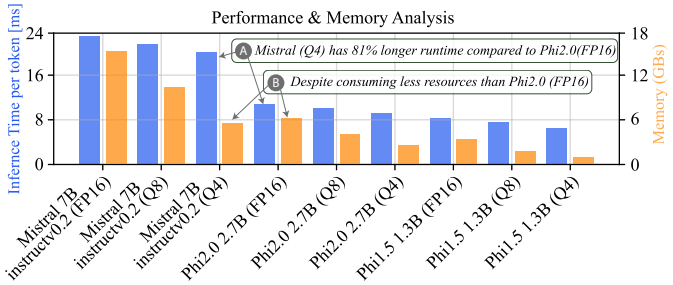

Fig. 7: Model Inference Metrics for quantized (FP16, Q8, Q4) models. These metrics are obtained on RTX 6000 Ada GPU.

C. Speech Synthesis

This paper conducts a comparative analysis of various Text-to-Speech (TTS) models, evaluating their speech synthesis capabilities with a focus on performance, resource consumption, and runtime efficiency. Our assessment utilizes the Mean Opinion Score (MOS) to gauge audio quality. As detailed in Table 3, while the NaturalSpeech model demonstrates a notably higher MOS, indicating superior speech quality, it also requires substantially more resources and incurs a longer runtime than its counterparts. Glow-TTS [34], despite not achieving the highest MOS score, stands out for its excellent runtime and memory efficiency. This balance makes it particularly suitable for deployment on resource-constrained platforms. Consequently, we have chosen Glow-TTS as our preferred model, coupled with the HiFi-GAN [50] vocoder for speech synthesis. A vocoder is a component that converts the raw, synthesized mel-spectrogram output of a TTS model like Glow-TTS into audible waveforms. Glow-TTS employs flow-based generative methods and dynamic programming to autonomously align text and speech. This results in robust performance for extended utterances and significantly faster synthesis. Glow-TTS innovates in training and inference by modeling the conditional distribution of mel-spectrograms, $P_X(x|c)$, using a flow-based decoder, f_{dec}. This method circumvents the need for external aligners, as expressed in the equation for the exact log-likelihood of the data.

$$\log P_X(x|c) = \log P_Z(z|c) + \log \left| \det \left(\frac{\partial f_{\text{dec}}^{-1}(z)}{\partial z} \right) \right| \quad (1)$$

Here, $P_Z(z)$ is an isotropic multivariate Gaussian distribution, parameterized with network parameters θ and an alignment function A. The statistics of the prior distribution are linked to the output from the text encoder, adapting to the text's length and content, enabling the model to handle variable text lengths and complexities effectively. The model architecture is illustrated in Figure 8.

TABLE III
PERFORMANCE METRICS OF TTS MODELS. RUNTIME AND MEMORY CONSUMPTION IS EVALUATED ON RTX 6000 ADA GPU

TTS Model	MOS	Memory	Runtime (s)
Glow-TTS + HiFi-GAN	4.34 ± 0.13	**1.9 GB**	**0.024**
FastSpeech 2 + HiFi-GAN	4.32 ± 0.15	2.0 GB	0.031
Grad-TTS + HiFi-GAN	4.37 ± 0.13	2.5 GB	0.076
VITS	4.43 ± 0.13	3.4 GB	0.25
NaturalSpeech	**4.56 ± 0.13**	4.1 GB	0.29

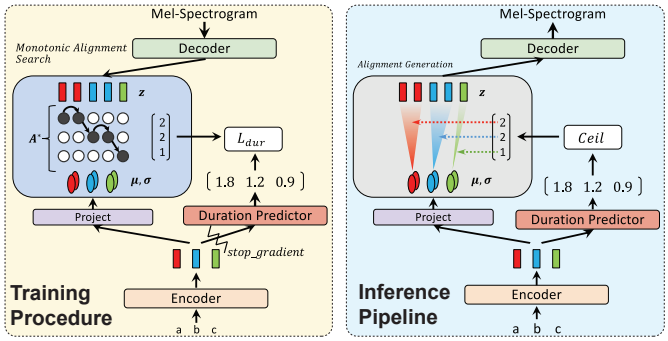

Fig. 8: An overview of the Glow-TTS training and inference pipeline: The training procedure utilizes a Monotonic Alignment Search to feed the Duration Predictor, optimizing the match between text input and Mel-Spectrogram output through the decoder. The inference pipeline highlights the role of the Duration Predictor in alignment generation, where it determines the exact duration of each phoneme for precise speech synthesis, ultimately resulting in the generation of a Mel-Spectrogram through the decoder.

The dataset generation process is automated using the Whisper model by transcript generation. The ASR models like Whisper are trained on short audio segments up to 30 seconds, constrained by transformer architectures that limit transcription of longer inputs due to memory constraints. To effectively automate the process, the speech is first segmented into shorter 16 bit mono wav files, which are then input to the ASR module for transcript and timestamp prediction. This process is illustrated in Figure 9 and the algorithm details the process of segmenting the audio files into 5-10 second clips at the completion of sentences or natural speech pauses. This automation streamlines the creation of a robust dataset.

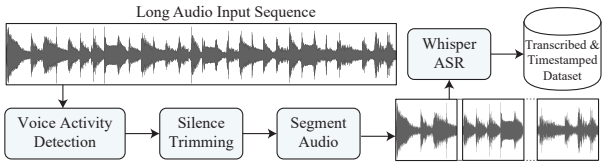

Fig. 9: Automated dataset generation for speech synthesis model using ASR.

The data refinement process encompasses several key steps:

- **De-noising:** Removing background noise to improve data clarity using filtering algorithms.

Algorithm 1 Segment Speech Algorithm

Require: vadScores (Voice Activity Detection Score), maxDuration, onsetTh, offsetTh, timeStep

$speechSeg \leftarrow []$, $segSt \leftarrow 0$
$isActiveSeg \leftarrow$ vadScores[0] $> offsetTh$
$maxLength \leftarrow$ int$(maxDuration \times timeStep)$
for $Idx = 1$ **to** length(vadScores) $- 1$ **do**
  $currentScore \leftarrow$ vadScores$[Idx]$
  if $isActiveSeg$ **then**
    if $Idx - segSt \geq maxLength$ **then**
      $StIdx \leftarrow Idx + maxLength/2$
      $EndIdx \leftarrow Idx + maxLength$
      $minIdx \leftarrow$ argmin(vadScores$[StIdx : EndIdx]$)
      speechSeg.append($(segSt, StIdx + minIdx)$)
      $segSt \leftarrow StIdx + minIdx$
    else if $currentScore < offsetTh$ **then**
      speechSeg.append($(segSt, Idx)$)
      $isActiveSeg \leftarrow$ False
    end if
  else
    if $currentScore > onsetTh$ **then**
      $segSt \leftarrow Idx$
      $isActiveSeg \leftarrow$ True
    end if
  end if
end for
return $speechSeg$

- **Normalization:** Adjusting data to a standard scale for consistency.
- **Time-stamp Prediction:** Utilizing Whisper for accurate time-stamping within the audio stream.
- **Silence Trimming:** Eliminating non-informative silent segments using voice activity detection (VAD) algorithm.
- **Speech Segmentation:** Breaking down data into smaller, manageable segments using Algorithm 1.

The model is trained using 15 min of audio samples segmented into 110 wav files each within 8-10 sec length. The user has the option to train the speech model on their device using an automated routine, provided the device meets the minimum requirements for training. Moreover, if the user lacks the necessary hardware to train the TTS model, it can be trained on a cloud service, which requires an internet connection for the initial setup. The training performance is shown in Figure 10:

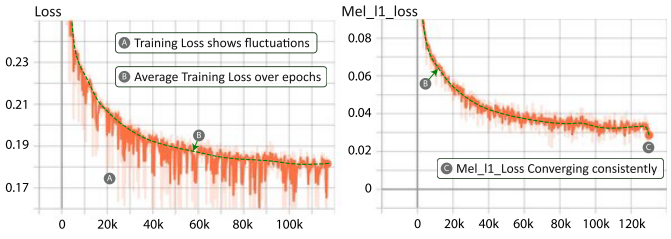

Fig. 10: Training loss of the Glow-TTS model across epochs; although the loss fluctuates, it eventually converges to an acceptable state.

D. System Integration and Model Selection

In the development of our integrated tinyDigiClones system, one of the critical tasks is the selection of models for ASR and LLM models that align with the specific hardware capabilities and user requirements. To address this, we have devised an algorithmic approach that systematically evaluates and selects the most suitable models. The algorithm, presented in this section, utilizes a combination of the Minimum Remaining Values (MRV) and Least Constraining Value (LCV) heuristics.

This algorithm is effective in our use-case since it considers the multifaceted nature of our problem, where each model must be evaluated against a range of criteria, including performance efficiency, resource consumption, and compatibility with hardware constraints. The MRV heuristic helps in quickly narrowing down the pool of potential models by prioritizing those with the fewest viable configurations. Simultaneously, the LCV heuristic aids in maintaining system flexibility, ensuring that the selected model imposes the least restrictions on subsequent choices. The model's compatibility is determined based on the hardware architecture; for example, older ARM-based development boards do not support 4-bit quantization. The flowchart outlining these steps is detailed in Figure 11. The algorithm 2 details the MRV heuristic based model selection and algorithm 3 explores the LCV function.

Algorithm 2 Select Model with Fewest Choices (MRV)

Require: Models, HardwareConstraints, UserRequirements
Ensure: SelectedModel

$MinChoices \leftarrow \infty$
$SelectedModel \leftarrow$ None
for $model$ in $Models$ **do**
  $ValidChoices \leftarrow 0$
  for $config$ in $model$ **do**
    if ConfigIsValid($config$, $HardwareConstraints$, $UserRequirements$) **then**
      $ValidChoices \leftarrow ValidChoices + 1$
    end if
  end for
  if $ValidChoices < MinChoices$ **then**
    $MinChoices \leftarrow ValidChoices$
    $SelectedModel \leftarrow model$
  end if
end for
return $SelectedModel$

Algorithm 3 Select Least Constraining Value (LCV)

Require: $Model$, $ConfigOptions$, $HwResources$, $UserRequirements$
Ensure: LCV

$LCV \leftarrow$ None
$MinResourceImpact \leftarrow \infty$
$MinExecutionImpact \leftarrow \infty$
for $config$ in $ConfigOptions$ **do**
  $ResourceImpact \leftarrow$ CalRI$(config, HwResources)$
  $ExecutionImpact \leftarrow$ CalEI$(config, HwResources)$
  if $UserRequirements$ is $RealTimeMode$ **then**
    if $ExecutionImpact < MinExecutionImpact$ **then**
      $MinExecutionImpact \leftarrow ExecutionImpact$
      $LCV \leftarrow config$
    end if
  else if $UserRequirements$ is $AccuracyMode$ **then**
    if $ResourceImpact < MinResourceImpact$ **then**
      $MinResourceImpact \leftarrow ResourceImpact$
      $LCV \leftarrow config$
    end if
  end if
end for
return LCV

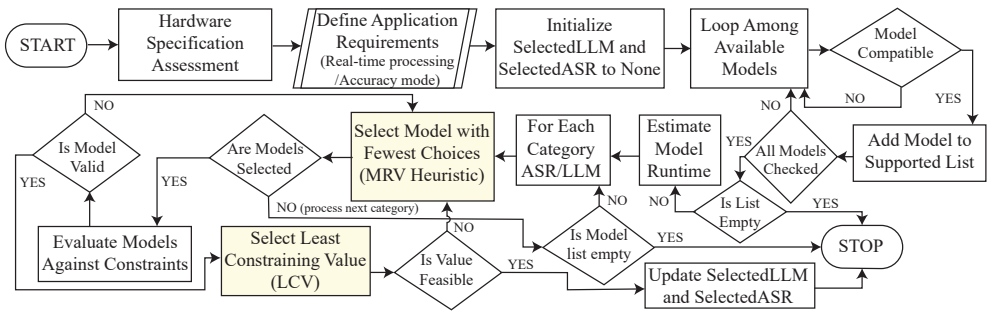

Fig. 11: Model Selection Flowchart detailing the process of ASR and LLM model selection based on user requirements and hardware constraints. The structured and heuristic-based nature of the flowchart ensures that our model selection process is both thorough and efficient, leading to an optimized configuration that is well-suited for the demands of the systems deployed in edge computing environments.

The runtime approximation for model inference on a given GPU is based on the computational characteristics of the hardware and the size of the model. This process considers two primary factors: the memory bandwidth of the GPU, which affects the time taken to move model weights during inference, and the computational throughput, measured in floating-point operations per second (TFLOPs), which dictates the time required for calculation-intensive operations.

Given a model with a number of parameters P, the time to generate a single token can be estimated using the available memory bandwidth BW (in bytes per second) of the GPU and BP (bytes per parameters): $t_{\text{token}} = \frac{BP \times P}{BW}$. Note that in case of FP16 precision, each parameter requires 2 bytes per parameter for loading the model weights during inference. For the prefill stage, where all prompt tokens are processed in a batch assuming the operation is compute-bound, the time can be estimated using the computational power of the GPU in TFLOPs (TF): $t_{\text{prefill}} = \frac{N_{\text{prompt}} \times FP \times P}{TF}$, where N_{prompt} represents the number of prompt tokens and FP represents FLOPs per parameters which is 2 in the case of FP16 precision. The total generation time T_{total} for generating the complete sequence, which includes the prefill time and the time to generate each subsequent token, is given by: $T_{\text{total}} = t_{\text{prefill}} + N_{\text{completion}} \times t_{\text{token}}$, where $N_{\text{completion}}$ is the number of tokens to be generated after the prefill stage.

This approximation assumes ideal conditions of communication overhead, negligible latency for each forward pass, and perfect parallelization. In practice, actual runtime may vary significantly, since it depends on many different factors, but given our use-case it would be an acceptable estimate, considering relative comparison on varying devices.

E. Digital Avatar Animation

The personalized 3D avatars creation process is automated using Union Avatars, a service that transforms user-provided images into customizable 3D models in FBX format. This process requires internet access only for the initial setup, where the user's image is uploaded and subsequently converted into a digital avatar. Post-creation, users have the flexibility to customize their avatar's appearance, including apparel, style, and body. Upon finalizing the avatar customization, it is integrated into our framework, and animation is incorporated.

We employ the Oculus Lipsync (OVR) technology for animating the avatar's lip movements, synchronizing them with speech output. Oculus Lipsync, a Unity plugin compiled for linux, offers sophisticated real-time lip-sync capabilities. It operates by analyzing audio streams generated by the speech synthesis model and predicts a set of visemes. These visemes represent facial expressions or gestures corresponding to specific speech sounds for achieving realistic lip movements. It is a lightweight model that runs in real-time on most low-end devices and occupies 50MB of GPU resources. Our framework integrates the OVR system by mapping these visemes to the avatar's blendshapes. Figure 12 shows this mapping enabling the avatar's lips to move realistically in sync with the generated speech from TTS model.

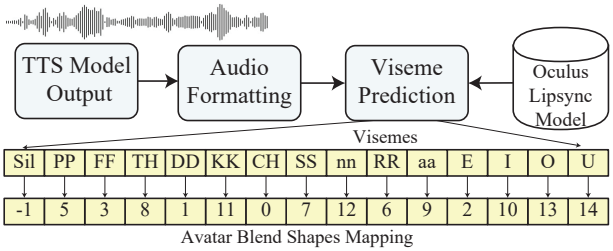

Fig. 12: OVR LipSync integration with a digital avatar blend shapes. Unity handles audio using its specific AudioClip format, and the generated audio in WAV is converted to match the Unity engine's format for viseme prediction.

The Oculus Lipsync system utilizes 15 distinct visemes, covering a wide range of lip movements and expressions, making it language-agnostic and highly versatile. The automatic detection and mapping routines we developed ensure a seamless transition from speech synthesis to corresponding visual representation, vital for creating a cohesive and interactive digital avatar experience.

IV. Experimental Setup and Results

We evaluated the performance of our tinyDigiClones with a particular focus on assessing feasibility for real-time deployment on edge devices. The primary hardware used in our tests included the NVIDIA Jetson Orin Nano, a compact yet powerful edge computing device known for its neural network acceleration capabilities, and a range of consumer-grade NVIDIA GPUs. These GPUs represent the typical consumer hardware that potential users may possess, providing a relevant benchmark for our system's performance outside of specialized server environments. The experimental setup is highlighted in Figure 13.

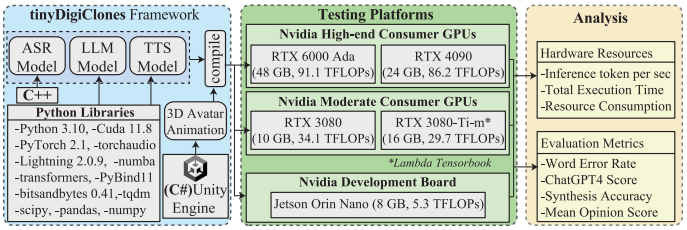

Fig. 13: Experimental evaluation setup deployed on Python and C++, tested on various platforms, with evaluation based on metrics and resource usage.

The device's specifications are conducive to running advanced deep learning models while operating within the typical resource constraints of edge devices. For comparison, the consumer-grade GPUs from NVIDIA's product lineup vary in terms of processing power, memory, and price points, thereby offering a comprehensive perspective on how our system scales across different hardware tiers. Figure 14 shows the LLM performance across different devices.

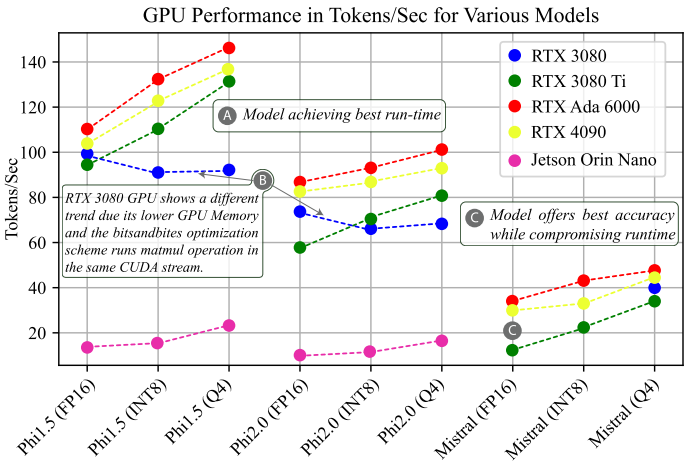

Fig. 14: GPU performance for selected LLMs across various GPU devices, where tokens/sec serve as a measure of runtime efficiency: higher token generation rates correspond to reduced runtime. (B) highlights an observation where performance deviates from the estimated trend.

In our analysis, we assessed the system's performance using several key metrics, including latency, model inference time, resource consumption, and the quality of the generated output. The tinyDigiClones framework estimates the model configuration most suitable for a given GPU device and user requirements. In the case of real-time inference mode, models with lower parameters and quantizations are employed. For accuracy mode, the best model that can be deployed on the specified hardware is selected based on heuristics. Table 4 lists the different models deployed using the framework on given devices along with their runtime metrics and resource usage.

Figure 15 shows the evaluation metrics for our framework. Due to the unavailability of a standard benchmark for testing this framework, we conducted our analysis using a custom-mod in ChatGPT-4, utilizing the GPT builder, named LLM-eval. This model is specifically designed to assess the coherence and accuracy of the LLM's output, with the

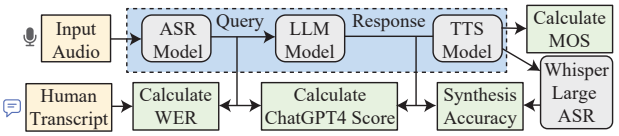

Fig. 15: The tinyDigiClones framework is analyzed using an end-to-end approach, where a model's output is evaluated against a ground truth. In the case of LLM, this evaluation is conducted using ChatGPT-4.

consideration that the model will be generating brief chat responses. LLM-eval enables ChatGPT-4 to analyze the input query in comparison to the generated output and assign a score based on the quality of the LLM generation. The validity of this score is grounded in GPT-4's performance and the benchmark scores claimed in their technical report [6]. We conducted this experiment using 100 single-turn questions, scoring the responses generated by the LLM for each model and then calculating the mean score listed in Table 4.

Furthermore, for evaluating the TTS model, we utilized Whisper Large to generate transcripts of the speech produced. We acknowledge that Whisper Large, like any large-scale ASR system, inherently introduces transcription errors. Therefore, we consider the combined WER score as indicative of synthesis accuracy and assume that any errors exceeding the baseline WER of Whisper can be attributed to the TTS system. Our evaluation estimated a Synthesis Accuracy of 91.4%. Moreover, the Mean Opinion Score (MOS) was evaluated using human feedback, and for our trained TTS model it was evaluated to be 4.36.

V. Conclusion

Our research has demonstrated a comprehensive workflow for generating real-time, multi-modal digital clones on edge devices. By integrating state-of-the-art technologies such as efficient speech-to-text conversion, large language model integration and speech synthesis, we have established an efficient framework for creating interactive and personalized digital experiences. This workflow's optimization for edge deployment highlights its potential for wide-ranging applications from virtual assistance to augmented reality.

Acknowledgment

This work was supported in part by the NYUAD Center for Artificial Intelligence and Robotics (CAIR), funded by Tamkeen under the NYUAD Research Institute Award CG010.

References

[1] P. Klaus and J. L. Zaichkowsky, "The convenience of shopping via voice ai: Introducing aidm," *Journal of Retailing and Consumer Services*, vol. 65, p. 102490, 2022.

[2] S. Malodia, N. Islam, P. Kaur, and A. Dhir, "Why do people use artificial intelligence (ai)-enabled voice assistants?" *IEEE Transactions on Engineering Management*, pp. 1–15, 2021.

[3] M. Shumanov and L. Johnson, "Making conversations with chatbots more personalized," *Computers in Human Behavior*, vol. 117, 2021.

[4] Nvidia. (2024, Apr) Nvidia omniverse. [Online]. Available: https://developer.nvidia.com/omniverse/ace

[5] Meta. (2024, Apr) Facebook metaverse. [Online]. Available: https://about.meta.com/metaverse/

TABLE IV
SUMMARY OF PERFORMANCE, METRICS, AND RESOURCE EVALUATION ACROSS VARIOUS DEVICES IN ACCURACY AND REAL-TIME MODES USING SELECTED MODEL CONFIGURATIONS BY THE TINYDIGICLONES FRAMEWORK.

Model	GPU Memory	TFLOPs	Mode	Model Configuration	LLM Runtime	ASR Runtime	TTS Runtime	Resource Consumed	ChatGP4 Score
RTX 6000 Ada	48 GB	91.1	Acc	Mistral (FP16) + Wp-L	1.64 s	0.8624 s	0.0278 s	25.43 GB	4.70
			RT	Phi2.0 (FP16) + Wp-M	0.56 s	0.0881 s	0.0263 s	12.98 GB	4.26
RTX 4090	24 GB	86.2	Acc	Mistral (FP16) + Wp-M	1.57 s	0.0960 s	0.0296 s	23.24 GB	4.70
			RT	Phi2.0 (FP16) + Wp-M	0.65 s	0.0929 s	0.0284 s	12.93 GB	4.26
RTX 3080-Ti	16 GB	29.7	Acc	Mistral (Q8) + Wp-S	2.24 s	0.1070 s	0.0183 s	12.32 GB	4.51
			RT	Phi2.0 (FP16) + Wp-T	0.82 s	0.0362 s	0.0175 s	6.28 GB	4.26
RTX 3080	10 GB	34.1	Acc	Mistral (Q4) + Wp-S	1.25 s	0.0388 s	0.0265 s	8.47 GB	4.39
			RT	Phi1.5 (FP16) + Wp-B	0.63 s	0.0385 s	0.0253 s	5.10 GB	3.28
Orin Nano	8 GB	5.3	Acc	Phi2.0 (FP16) + Wp-T	6.25 s	0.2354 s	0.1572 s	7.74 GB	4.26
			RT	Phi1.5 (Q4) + Wp-T	2.37 s	0.2291 s	0.1491 s	3.61 GB	2.97

[6] J. Achiam, S. Adler, and S. Agarwal, "Gpt-4 technical report," *arXiv:2303.08774*, 2023.

[7] A. Chowdhery, S. Narang, J. Devlin *et al.*, "Palm: Scaling language modeling with pathways," *Journal of Machine Learning Research*, vol. 24, no. 240, pp. 1–113, 2023.

[8] E. Almazrouei, H. Alobeidli, A. Alshamsi *et al.*, "The falcon series of open language models," *arXiv:2311.16867*, 2023.

[9] L. Gao, J. Tow, S. Biderman *et al.*, "A framework for few-shot language model evaluation," *Zenodo Version v0. 0.1. Sept*, p. 8, 2021.

[10] Y. Leviathan and Y. Matias, "Google duplex: An ai system for accomplishing real-world tasks over the phone," 2018.

[11] H. Li, Y. Chen, J. Luo *et al.*, "Privacy in large language models: Attacks, defenses and future directions," *arXiv:2310.10383*, 2023.

[12] X. Wu, R. Duan, and J. Ni, "Unveiling security, privacy, and ethical concerns of chatgpt," *Journal of Information and Intelligence*, 2023.

[13] U. Iqbal, T. Kohno, and F. Roesner, "Llm platform security: Applying a systematic evaluation framework to openai's chatgpt plugins," *arXiv:2309.10254*, 2023.

[14] W. X. Zhao, K. Zhou, J. Li *et al.*, "A survey of large language models," *arXiv:2303.18223*, 2023.

[15] W.-C. Huang, C.-H. Wu, S.-B. Luo *et al.*, "Speech recognition by simply fine-tuning bert," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021.

[16] Y. Fu, L. Cheng, S. Lv *et al.*, "Aishell-4: An open source dataset for speech enhancement, separation, recognition and speaker diarization in conference scenario," *arXiv:2104.03603*, 2021.

[17] A. Radford, J. W. Kim, T. Xu *et al.*, "Robust speech recognition via large-scale weak supervision," in *International Conference on Machine Learning*. PMLR, 2023, pp. 28 492–28 518.

[18] X. Wu, "Deep sparse conformer for speech recognition," *arXiv:2209.00260*, 2022.

[19] Y.-A. Chung, Y. Zhang, W. Han *et al.*, "W2v-bert: Combining contrastive learning and masked language modeling for self-supervised speech pre-training," in *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2021, pp. 244–250.

[20] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai *et al.*, "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3451–3460, 2021.

[21] H. Touvron, T. Lavril, G. Izacard *et al.*, "Llama: Open and efficient foundation language models," *arXiv:2302.13971*, 2023.

[22] H. Touvron, L. Martin, K. Stone *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv:2307.09288*, 2023.

[23] T. Le Scao, A. Fan, C. Akiki *et al.*, "Bloom: A 176b-parameter open-access multilingual language model," *BigScience Workshop*, 2023.

[24] G. Penedo, Q. Malartic, D. Hesslow *et al.*, "The refinedweb dataset for falcon llm: Outperforming curated corpora with web data only," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[25] G.-J. Hwang and C.-Y. Chang, "A review of opportunities and challenges of chatbots in education," *Interactive Learning Environments*, vol. 31, no. 7, pp. 4099–4112, 2023.

[26] L. Kohnke, B. L. Moorhouse, and D. Zou, "Chatgpt for language teaching and learning," *RELC Journal*, p. 00336882231162868, 2023.

[27] P. Smutny and P. Schreiberova, "Chatbots for learning: A review of educational chatbots for the facebook messenger," *Computers & Education*, vol. 151, p. 103862, 2020.

[28] A. Q. Jiang, A. Sablayrolles, A. Mensch *et al.*, "Mistral 7b," *arXiv:2310.06825*, 2023.

[29] Z. Li, Z. Yang *et al.*, "Reinforcement learning with human feedback: Learning dynamic choices via pessimism," *arXiv:2305.18438*, 2023.

[30] R.-Y. Sun, "Optimization for deep learning: An overview," *Journal of the Operations Research Society of China*, vol. 8, pp. 1–46, 06 2020.

[31] C. Wang, S. Chen, Y. Wu, Z. Zhang, L. Zhou, S. Liu, Z. Chen, Y. Liu, H. Wang, J. Li *et al.*, "Neural codec language models are zero-shot text to speech synthesizers," *arXiv:2301.02111*, 2023.

[32] M. Le, A. Vyas, B. Shi *et al.*, "Voicebox: Textguided multilingual universal speech generation at scale.(2023)," *arXiv:2306.15687*.

[33] E. Kharitonov, D. Vincent, Z. Borsos *et al.*, "Speak, read and prompt: High-fidelity text-to-speech with minimal supervision," *Transactions of the Association for Computational Linguistics*, vol. 11, 2023.

[34] J. Kim, S. Kim, J. Kong, and S. Yoon, "Glow-tts: A generative flow for text-to-speech via monotonic alignment search," *Advances in Neural Information Processing Systems*, vol. 33, pp. 8067–8077, 2020.

[35] X. Tan, J. Chen, H. Liu *et al.*, "Naturalspeech: End-to-end text-to-speech synthesis with human-level quality," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[36] K. Ito and L. Johnson, "The lj speech dataset," 2017. [Online]. Available: https://keithito.com/LJ-Speech-Dataset

[37] F. S. Oliveira, E. Casanova, A. C. Junior *et al.*, "Evaluation of speech representations for mos prediction," in *International Conference on Text, Speech, and Dialogue*. Springer, 2023, pp. 270–282.

[38] J. W. Kim, R. Heise, G. Klein *et al.* (2024, Apr) Whisper-openai. [Online]. Available: https://github.com/openai/whisper

[39] S. Gandhi. (2024, Apr) Whisper-jax. [Online]. Available: https://github.com/sanchit-gandhi/whisper-jax

[40] G. Gerganov. (2024, Apr) Whisper-cpp. [Online]. Available: https://github.com/ggerganov/whisper.cpp

[41] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An asr corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 5206–5210.

[42] C. Zhang, D. Song, Z. Ye, and Y. Gao, "Towards the law of capacity gap in distilling language models," *arXiv:2311.07052*, 2023.

[43] D. Hendrycks, C. Burns, S. Basart *et al.*, "Measuring massive multitask language understanding," 2021.

[44] D. Dua, Y. Wang, P. Dasigi *et al.*, "Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs," *arXiv:1903.00161*, 2019.

[45] Y. Huang, Y. Bai, Z. Zhu *et al.*, "C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[46] K. Cobbe, V. Kosaraju, M. Bavarian *et al.*, "Training verifiers to solve math word problems," *arXiv:2110.14168*, 2021.

[47] Q. Ye, H. Y. Fu, X. Ren, and R. Jia, "How predictable are large language model capabilities? a case study on big-bench," *arXiv:2305.14947*, 2023.

[48] M. Chen, J. Tworek, H. Jun *et al.*, "Evaluating large language models trained on code," *arXiv:2107.03374*, 2021.

[49] Y. Li, S. Bubeck, R. Eldan *et al.*, "Textbooks are all you need ii: **phi-1.5** technical report," *arXiv:2309.05463*, 2023.

[50] J. Kong, J. Kim, and J. Bae, "Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis," *Advances in neural information processing systems*, vol. 33, pp. 17 022–17 033, 2020.