

HIT-ASR: Hierarchical Transformer Routing for Adaptive ASR Expert Selection

Huseyin Karaca, A. Samil Namli, and Suleyman S. Kozat, *Senior Member, IEEE*

Abstract—Different pretrained automatic speech recognition (ASR) systems excel under different acoustic conditions, speakers, and domains, so that the system that best transcribes a given clip is not known in advance and varies from clip to clip. We study how to select, for each clip, one system from a pool of complete, frozen, and heterogeneous ASR systems before any of them decodes. We introduce HIT-ASR, a hierarchical transformer router that operates on the frame-level encoder states of every expert: a per-expert temporal encoder summarizes each expert’s frame sequence at that expert’s native frame rate, a cross-expert fusion step combines the summaries, and a WER-aware objective trains the router on the measured error of every expert on every clip. We evaluate HIT-ASR on four corpora (AMI-SDM, Earnings-22, People’s Speech, and AfriSpeech-200), each with its own trio of current open ASR systems. HIT-ASR attains the lowest word error rate (WER) on every corpus, 6.3–11.2% relative below that of the best single expert. Its improvements over the best single expert, over a pooled-feature router trained with the same WER-aware objective, and over two hypothesis-level ensembles are statistically significant on every corpus. At inference, all K expert encoders run once per clip, the router operates on their states, and only the selected expert is decoded. Measured end to end on raw audio, HIT-ASR requires less computation time per clip than the best single expert on three of the four corpora. The code, the data, and a reproducibility package are publicly available.

Index Terms—Speech recognition, automatic speech recognition (ASR), model selection, algorithm selection, adaptive routing, transformer, frame-level representations, system combination, ensemble of pretrained models.

I. INTRODUCTION

A. Background

Automatic speech recognition (ASR, also referred to as speech-to-text) has advanced significantly with deep sequence models [1], [2], [3]. The Transformer architecture [4] and its convolution-augmented variant, the Conformer [5], now achieve state-of-the-art accuracy on benchmarks such as LibriSpeech, outperforming recurrent and convolutional models [3], [6]. Self-supervised pretraining [7], [8], [9] further improves accuracy, and streaming architectures such as the RNN-Transducer [10] and its transformer-based extensions enable low-latency transcription [11], [12]. Large-scale weakly supervised training [13] and speech-aware large language models

[14], [15] have since multiplied the number of open systems available to practitioners. However, each of these systems is static in that it applies one model to all inputs. In realistic deployments, acoustic conditions such as noise, channel, and speaker can vary unpredictably over time, and no single ASR system is optimal in all scenarios [16], [17], [18]. Different models perform better in different acoustic regimes and thus form a complementary set of experts. Exploiting this diversity requires an adaptive mechanism that selects the most suitable model for each input instead of committing to a fixed system.

A key difficulty in this setting is predicting the performance of candidate models without executing them [19]. Observable features of the audio signal, such as acoustic statistics, temporal dynamics, and learned representations, provide useful cues for this purpose [7], [8], [16], [20]. Several related approaches to ASR confidence estimation and system selection compress an utterance into a sequence-level score or embedding before making a downstream decision [21], [22], [23]. Such pooling can obscure temporal structure that is relevant for routing, because speech acoustics and interaction conditions evolve over time [16], [24]. Within a single clip, the speaker, the channel, the background, and the spoken content are rarely homogeneous [16], [17], [20], [25]. For example, a single meeting utterance may contain both overlapping speech and clean segments [17], [18], [25], [26], and the speaking rate or emotion may change within an utterance. The expert that best transcribes such a clip may therefore depend strongly on these intra-clip dynamics rather than only on a single global summary [21], [24].

This observation motivates routers that read the frame-level temporal structure of a clip before deciding which expert transcribes it. In this paper, we introduce HIT-ASR, a hierarchical transformer router that operates on the frame-level encoder states of the candidate experts rather than on pooled features. The router selects one expert for the whole clip: the temporal structure within the clip informs this clip-level decision, and the selected expert then transcribes the clip. Because the decision is made before decoding, the encoders of all experts run once per clip, but only the decoder of the selected expert runs. We assess the router by the criterion of per-instance algorithm selection [27], [19]: it should never perform worse than the best single expert, which a practitioner would deploy without a router, and it should close as much as possible of the gap to the per-clip oracle.

B. Related Work

Modern end-to-end ASR systems often use attention-based encoder-decoder or transducer models [3]. The original Trans-

H. Karaca and S. S. Kozat are with the Department of Electrical and Electronics Engineering, Bilkent University, Ankara 06800, Türkiye (e-mails: huseyin.karaca@bilkent.edu.tr; kozat@ee.bilkent.edu.tr).

A. S. Namli is with the School of Computer Science, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: anamli3@gatech.edu).

This work was supported by The Scientific and Technological Research Council of Türkiye (TUBITAK) 1515 Frontier R&D Laboratories Support Program for Turk Telekom 6G R&D Lab under project number 5249902, and by the TUBITAK 2210-A program.

former [4] eliminated recurrence in sequence-to-sequence tasks and has been widely adapted for speech. In ASR, the Conformer architecture [5] augments self-attention with convolutional modules to better model local speech structure. Self-supervised pretraining methods, such as wav2vec 2.0 [7], learn useful speech representations from unlabeled audio and enable strong ASR performance even with limited labeled data; streaming extensions include the RNN-Transducer [10] and its attention-based variants. These models improve the accuracy and, in streaming settings, the latency of a single network. Our work is orthogonal to these approaches: rather than modifying the architecture of a single ASR model, HIT-ASR selects, for each clip, one of several complete pretrained systems, so that any improvement to an individual system carries over to the pool.

When several recognizers are available, their outputs can also be combined after decoding. ROVER aligns the hypotheses of several systems into a word transition network and votes within each slot by frequency of occurrence or by confidence [28], and confusion-network combination selects the word sequence with the minimum expected word error from the merged word posteriors of the systems [29], [30]. Confidence-based ensembles of end-to-end models select the output of the most confident model [31]. All of these methods run every system to completion before combining the outputs, so that their cost includes the full pipeline of every expert; HIT-ASR, in contrast, decides before decoding and runs a single decoder.

Selecting among complete systems is an instance of the algorithm selection problem [27], [19], in which a selector predicts, from features of an input, which of several algorithms performs best on it. ADASTT [32] performs system-level selection among pretrained speech-to-text systems with a gradient-boosted meta-learner over pooled utterance-level acoustic, prosodic, and embedding features. HIT-ASR is loosely related to ADASTT in that both route among complete pretrained systems; it differs in that it operates on the frame-level encoder states of the experts themselves, which makes the temporal structure within the clip available to the decision.

Mixture-of-experts (MoE) models route inside a single network [33], [34], [35], and sparse MoE layers scale model capacity at nearly constant cost per token [36], [37], [38], [39]. In ASR, SpeechMoE [40] inserts MoE layers with a learned router into an acoustic model, SpeechMoE2 [41] adds domain and accent embeddings to the routing, and the MoE Conformer of Hu et al. [42] routes among the expert layers of a streaming multilingual model while activating only part of its parameters. In these models, the experts are sub-networks trained jointly with the router on the target task, they share one representation space, and every frame or hidden state is dispatched inside the network. HIT-ASR instead addresses system-level routing: its experts are complete, independently pretrained ASR systems with different architectures, widths, frame rates, and decoders; the experts remain frozen; and the router makes one decision per clip from their encoder states before any of them decodes. The two approaches are complementary, since an MoE acoustic model can itself be an expert in the pool.

More broadly, conditional computation and dynamic neural

networks adapt the computation to the input [43], [44] by skipping recurrent updates or residual blocks [45], [46], by exiting early through intermediate classifiers [47], [48], [49], or by varying the depth of the computation across inputs or positions [50], [51]. These methods change the computation inside one network, whereas HIT-ASR selects among complete networks; they are orthogonal to system-level selection and can be applied inside each expert of the pool.

The novelty of HIT-ASR lies in the problem it addresses and in the design that this problem requires, not in its individual building blocks. HIT-ASR selects among complete, frozen ASR systems from their frame-level encoder states before any of them decodes, whereas previous system-level selectors decide from a pooled, utterance-level description of the clip, and MoE models route inside one jointly trained network. Its premise is that the encoder states of the experts carry information about which expert will fail on a clip, and that this information is local in time, so that averaging the states over the clip dilutes it. Two design choices follow from this setting. First, because the experts' states differ in width and frame rate, a per-expert temporal encoder summarizes each expert's sequence at its native frame rate before the experts are fused. Second, because no gradient reaches the experts, the router is supervised by the measured WER of every expert on every clip, so that a wrong choice is penalized in proportion to its cost.

C. Contributions

Our contributions are summarized as follows:

- We formulate the problem of selecting, for each clip and before decoding, one of several complete, frozen, and heterogeneous ASR systems, and we introduce HIT-ASR, a router that makes this decision from the frame-level encoder states of all experts.
- We propose a hierarchical router, in which a per-expert temporal encoder summarizes each expert's frame sequence at its native frame rate and a cross-expert fusion step with four interchangeable mechanisms combines the summaries, together with a WER-aware composite objective that supervises the router with the measured error of every expert on every clip.
- We evaluate HIT-ASR on four corpora, each with its own trio drawn from seven open ASR systems of five decoder families, against the best single expert, a pooled-feature router trained with the WER-aware objective, ADASTT, and two hypothesis-level ensembles, all tuned on the same hold-out and tested on the same folds with paired significance tests. HIT-ASR attains the lowest WER on every corpus and reduces the WER of the best single expert by 6.7, 6.3, 9.1, and 11.2% relative.
- We measure the inference cost end to end on raw audio and analyze the complexity and scalability of the router. Because all K encoders run once and only the selected expert decodes, HIT-ASR requires less computation time per clip than the best single expert on three of the four corpora.

- We release the code, the data, and a reproducibility package with three levels of reproduction at <https://github.com/huseyin-karaca/hit-asr>.

D. Organization

Section II formalizes the per-clip selection problem, Section III presents the router and its training objective, Section IV reports the experiments, Section V describes the reproducibility package, and Section VI concludes the paper. Throughout the paper, the term *expert* refers to a complete pretrained ASR system, comprising both its encoder and its decoder, and never to an encoder in isolation.

II. PROBLEM DESCRIPTION

We consider the problem of adaptive transcription at the level of individual audio clips. Let $\{x_n\}_{n=1}^N$ denote a collection of N audio clips, where each clip x_n is itself a temporal signal composed of a sequence of acoustic frames. The frame-level structure of a clip captures variations in background noise, speaker identity, accent, speaking rate, and recording quality. State-of-the-art ASR experts respond differently to these intra-clip variations [16], [17], [20], and the best-performing expert for a given clip is generally not known in advance.

Let $\mathcal{A} = \{A_1, A_2, \dots, A_K\}$ denote a finite set of K pretrained ASR experts, where $A_j \in \mathcal{A}$ represents the j -th expert. For each clip x_n , the transcription produced by the j -th expert is $Y_{n,j} = A_j(x_n)$. Each transcription $Y_{n,j}$ is compared with the ground-truth reference R_n by a transcription error measure, such as the standard word error rate (WER) [16]. The transcription loss incurred by using expert A_j on clip x_n is $\ell(Y_{n,j}, R_n)$, where $\ell(\cdot, \cdot)$ denotes the chosen error metric.

Because the reference of a new clip is unknown, the best transcription cannot be identified even after decoding, and running all K experts on every clip is expensive. The goal is therefore to select for each clip, before decoding, a single expert $j_n \in \{1, \dots, K\}$ that transcribes the whole clip, such that the resulting transcription $\hat{Y}_n = A_{j_n}(x_n)$ minimizes the cumulative transcription error over the dataset:

$$\sum_{n=1}^N \ell(Y_{n,j_n}, R_n).$$

Two reference points frame this goal. The best single expert, i.e., the expert with the lowest corpus WER on the training data, is the system a practitioner would deploy without a router; the per-clip oracle, $j_n^* = \arg \min_k \ell(Y_{n,k}, R_n)$, is the ceiling of any selector that chooses one expert per clip. Following the per-instance algorithm selection literature [27], [19], a router should never perform worse than the best single expert, and it should close as much as possible of the gap between the best single expert and the oracle.

A key challenge is that each clip is internally non-stationary: the speaker, channel, and background can shift across frames within the same clip, and these intra-clip dynamics influence which expert transcribes the clip most accurately. Selection mechanisms that compress a clip into a single pooled feature

vector discard this temporal structure and thereby lose information relevant to the choice of expert. We next introduce an algorithm that operates directly on frame-level encoder representations and uses a transformer-based router to model the temporal structure of each clip when selecting the expert that transcribes it.

III. HIT-ASR: HIERARCHICAL TRANSFORMER ROUTING FOR ADAPTIVE ASR EXPERT SELECTION

A. Algorithm Overview

HIT-ASR performs adaptive ASR model selection at the clip level by exploiting the frame-level temporal structure of each audio clip. The system operates in two phases, an offline training phase and an online inference phase, as illustrated in Fig. 1. The offline phase learns a routing model that predicts, from the frame-level encoder representations of a clip, which expert is most likely to transcribe that clip with the lowest error. The online phase uses the trained router to select an expert for each incoming clip and invokes only that expert to produce the final transcription.

In the offline training phase, the system is given a collection of labeled audio clips $\{(x_n, R_n)\}_{n=1}^N$. For each clip x_n and each expert $A_k \in \mathcal{A}$, the encoder of A_k is run once on x_n to obtain a frame-level hidden-state sequence

$$\mathbf{H}_n^{(k)} = (\mathbf{h}_{n,1}^{(k)}, \dots, \mathbf{h}_{n,T_n^{(k)}}^{(k)}), \quad \mathbf{h}_{n,t}^{(k)} \in \mathbb{R}^{D_k}, \quad (1)$$

where $T_n^{(k)}$ is the number of encoder frames produced by expert A_k for clip x_n and D_k is its hidden-state dimension. We use $\mathbf{H}_n^{(k)}$ rather than a pooled summary so that intra-clip acoustic dynamics are preserved. In parallel, the full ASR pipeline of each expert is run on the clip to produce its candidate transcription $Y_{n,k} = A_k(x_n)$ and the corresponding transcription error $\text{WER}(Y_{n,k}, R_n)$, where the WER is defined as $\text{WER}(Y_{n,k}, R_n) = \frac{S+I+D}{|R_n|}$, with S , I , and D denoting the numbers of substitutions, insertions, and deletions under standard Levenshtein alignment [16] and $|R_n|$ denoting the number of words in R_n . These errors serve as the supervision signal that indicates which expert performed best on each clip.

The router takes the full set of frame-level hidden-state sequences $\{\mathbf{H}_n^{(k)}\}_{k=1}^K$ as input and produces a probability vector $\mathbf{w}_n = (w_{n,1}, \dots, w_{n,K})$ over the experts. The router parameters are optimized so that $\mathbf{w}_n$ favors the experts with low error in the empirical per-clip error vector $(\text{WER}(Y_{n,k}, R_n))_{k=1}^K$, using the training objective described in Section III-D.

In the online inference phase, the system receives a new audio clip x_n . The encoder of each expert is run on x_n to obtain its frame-level hidden states $\mathbf{H}_n^{(k)}$, which are passed through the trained router g_θ to produce $\mathbf{w}_n$. The expert with the highest predicted probability,

$$j_n = \arg \max_{k \in \{1, \dots, K\}} w_{n,k},$$

is selected, and only its decoder is executed on the clip, reusing its encoder pass, to produce the final transcription $\hat{Y}_n = A_{j_n}(x_n)$. Because only the selected expert is decoded for each clip, the framework avoids running the decoders of

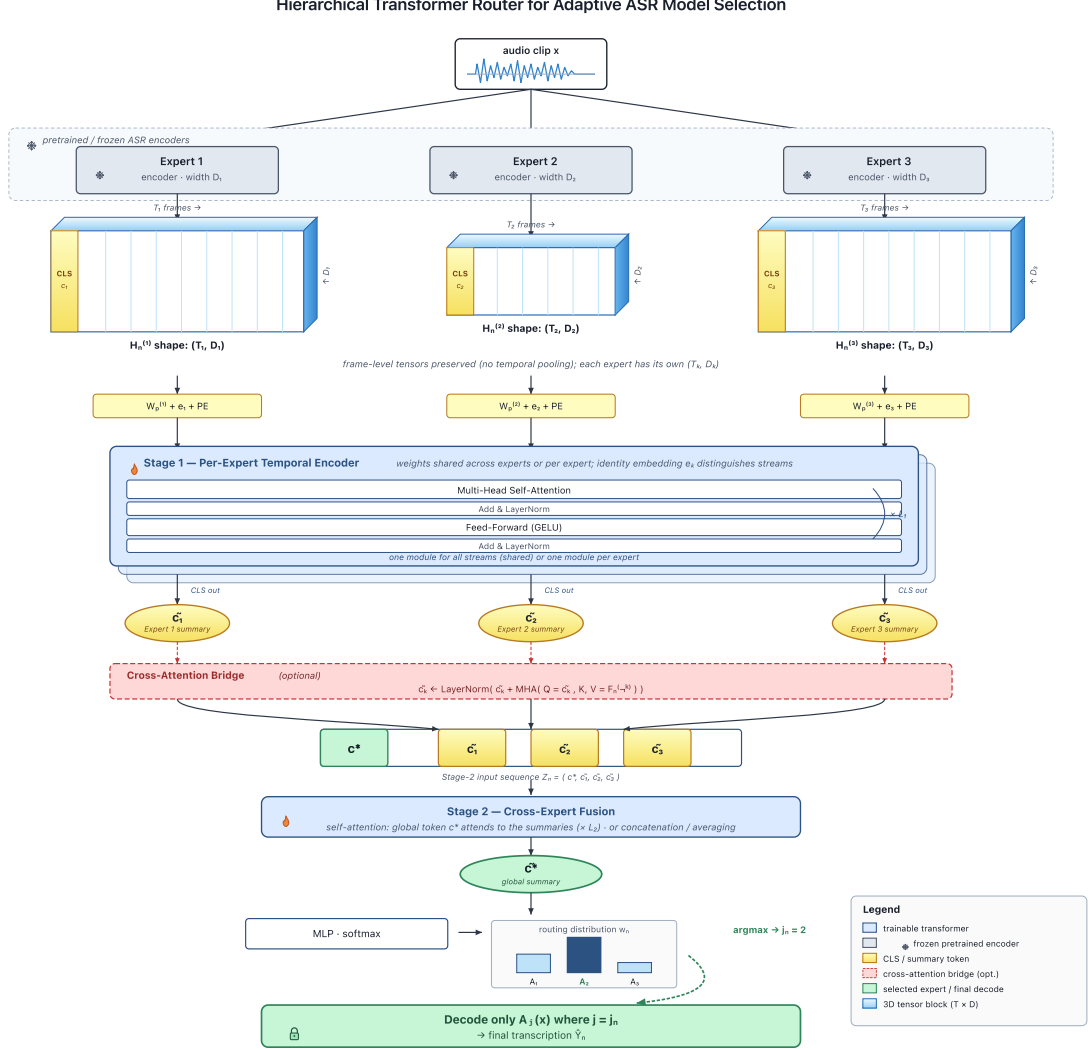


Fig. 1. Overview of the proposed adaptive ASR model selection framework, illustrated with $K = 3$ experts. Each frozen expert encoder produces a 2-D frame-level embedding tensor $\mathbf{H}_n^{(k)} \in \mathbb{R}^{T_n^{(k)} \times D_k}$ with model-specific frame count $T_n^{(k)}$ and hidden dimension D_k ; no temporal pooling is applied. Per-model linear projections, sinusoidal positional encodings, and learnable model-identity embeddings map the heterogeneous sequences to a shared dimension. A Stage 1 transformer encoder, either shared across the experts or separate for each expert, summarizes each expert’s frame sequence in a summary token. The summaries are fused by one of four mechanisms: a Stage 2 transformer with a global summary token, optionally preceded by a cross-attention bridge through which each summary attends to the frame-level outputs of the other experts; concatenation; or averaging. A small MLP followed by a softmax maps the fused representation to the routing distribution $\mathbf{w}_n$. At inference, only the argmax expert A_{j_n} is decoded.

the other experts, at the cost of K encoder forward passes and one router forward pass.

The following subsections describe the frame-level feature representation, the hierarchical router, and the training objective in detail.

B. Frame-Level Feature Representation

For each clip x_n and each expert A_k , we use the encoder of A_k as a feature extractor and keep its full frame-level hidden-state sequence $\mathbf{H}_n^{(k)}$ in (1), as illustrated by the per-expert encoder branches in Fig. 1. Because the encoders differ in architecture and frame rate, $T_n^{(k)}$ and D_k are model-specific. We do not pool these frames into a single vector. Instead, the temporal sequence is exposed directly to the router so that

intra-clip dynamics, such as shifts in noise level, speaker, or speaking rate, can influence the routing decision.

This choice contrasts with prior model-selection approaches that compress each clip into a single hand-crafted or pooled feature vector. Although such compact representations are inexpensive, they discard the temporal structure that often distinguishes the regimes in which different experts perform well. Keeping frame-level sequences allows the router to learn, for example, that an expert is preferable when a clip contains a stretch of overlapping speech or when its spectral characteristics drift over time.

The frame-level features are a by-product of the encoder forward pass that each expert performs as part of decoding, as shown in Fig. 1. They therefore require no acoustic feature extraction beyond what the experts already perform.

Algorithm 1 Online ASR Model Selection

Inputs:

- Audio clip x_n
- Set of base ASR encoders $\{E_1, \dots, E_K\}$ associated with the experts $\mathcal{A} = \{A_1, \dots, A_K\}$
- Trained routing function $g_\theta(\cdot)$

Output:

- Final transcription $\hat{Y}_n$ for clip x_n

Inference Steps:

- 1: Extract encoder representations: $\mathbf{H}_n^{(k)} \leftarrow E_k(x_n)$, $k = 1, \dots, K$
 - 2: Compute expert selection logits: $\mathbf{s}_n \leftarrow g_\theta(\{\mathbf{H}_n^{(k)}\}_{k=1}^K)$ ▷ See (5)
 - 3: Normalize logits via softmax to obtain routing probabilities $\mathbf{w}_n$ ▷ See (6)
 - 4: Select expert: $j_n \leftarrow \arg \max_{k \in \{1, \dots, K\}} w_{n,k}$
 - 5: Decode the selected expert: $\hat{Y}_n \leftarrow A_{j_n}(x_n)$
 - 6: Return $\hat{Y}_n$
-

C. Hierarchical Transformer Router

Given the per-expert frame-level sequences $\{\mathbf{H}_n^{(k)}\}_{k=1}^K$ obtained from the feature pipeline described in Section III-B, the routing model must map these heterogeneous sequences (with different $T_n^{(k)}$ and D_k across experts) to a single probability vector over the experts. We implement the router as a hierarchical transformer. The first stage produces a per-expert summary that captures the intra-clip temporal structure as seen through that expert's encoder. A fusion step then combines these summaries across experts to produce the final selection; this component of the router admits four mechanisms, which are described in Section III-C3.

1) *Per-Model Projection and Positional Encoding*: Because the encoder dimension D_k differs across experts, we first project each frame to a common router dimension d with a model-specific linear transformation, as shown in Fig. 1:

$$\mathbf{p}_{n,t}^{(k)} = W_p^{(k)} \mathbf{h}_{n,t}^{(k)} + \mathbf{b}_p^{(k)}, \quad W_p^{(k)} \in \mathbb{R}^{d \times D_k}. \quad (2)$$

A standard sinusoidal positional encoding [4] is added to each projected frame to preserve the frame order within the clip. To indicate which expert produced a given frame, a learnable model-identity embedding $\mathbf{e}_k \in \mathbb{R}^d$ is added to every frame of expert k . Before the projection, the frames of each expert can be normalized, either standardized per dimension or scaled by their root-mean-square value, with statistics computed on the training clips; whether and how the inputs are normalized is a hyperparameter of the router.

2) *Stage 1: Per-Expert Temporal Encoder*: For each expert k , we prepend a learnable summary token $\mathbf{c}_k \in \mathbb{R}^d$ to its projected frame sequence, forming

$$\mathbf{X}_n^{(k)} = (\mathbf{c}_k, \mathbf{p}_{n,1}^{(k)} + \mathbf{e}_k, \dots, \mathbf{p}_{n,T_n^{(k)}}^{(k)} + \mathbf{e}_k).$$

This sequence is processed by a transformer encoder [4], denoted by f_{stage1} ,

$$\mathbf{Y}_n^{(k)} = f_{\text{stage1}}(\mathbf{X}_n^{(k)}), \quad (3)$$

which performs masked multi-head self-attention and a position-wise feed-forward transformation in each layer.

Padding frames are excluded by a key-padding mask, and sequences longer than a maximum number of frames are truncated. The output corresponding to the summary token,

$$\tilde{\mathbf{c}}_n^{(k)} = [\mathbf{Y}_n^{(k)}]_0,$$

serves as the per-expert summary of clip x_n , pooled over time by attention rather than by averaging, and is passed from Stage 1 to the fusion step, as shown in Fig. 1.

The parameters of f_{stage1} can be shared across the K experts, in which case the model-identity embedding $\mathbf{e}_k$ carries the expert-specific information, or kept separate for each expert, which increases the capacity of Stage 1; the choice is a hyperparameter of the router.

3) *Cross-Expert Fusion*: The per-expert summaries $\{\tilde{\mathbf{c}}_n^{(k)}\}_{k=1}^K$ are combined into a single clip representation $\mathbf{z}_n$ by one of four fusion mechanisms; the choice of mechanism is a hyperparameter of the router, and the four are compared in Section IV-E.

Stage-2 self-attention. A learnable global summary token $\mathbf{c}^* \in \mathbb{R}^d$, which serves as the cross-expert aggregation token, is prepended to the summaries,

$$\mathbf{Z}_n = (\mathbf{c}^*, \tilde{\mathbf{c}}_n^{(1)}, \dots, \tilde{\mathbf{c}}_n^{(K)}),$$

and a second transformer encoder f_{stage2} processes $\mathbf{Z}_n$ to mix information across experts. The updated global token $\tilde{\mathbf{c}}_n^*$ is the fused representation, $\mathbf{z}_n = \tilde{\mathbf{c}}_n^*$.

Cross-attention bridge. Stage 1 produces the per-expert summary $\tilde{\mathbf{c}}_n^{(k)}$ from the frames of expert k alone. To inform each summary with frame-level evidence from the other experts, a cross-attention bridge can be inserted before Stage 2. For each expert k , let

$$\mathbf{F}_n^{(\neg k)} = [\mathbf{Y}_n^{(j)}]_{1:T_n^{(j)}} \mid j \neq k$$

denote the concatenated frame-level outputs of all other experts (with their summary positions removed). The bridge updates each summary by

$$\tilde{\mathbf{c}}_n^{(k)} \leftarrow \text{LN} \left(\tilde{\mathbf{c}}_n^{(k)} + \text{MHA}(\tilde{\mathbf{c}}_n^{(k)}, \mathbf{F}_n^{(\neg k)}, \mathbf{F}_n^{(\neg k)}) \right), \quad (4)$$

where MHA is multi-head attention with the summary as query and the other experts' frames as keys and values, and LN denotes layer normalization. The bridged summaries are then fused by Stage 2 as above.

Concatenation and averaging. The two simplest mechanisms use the summaries directly, either concatenated, $\mathbf{z}_n = [\tilde{\mathbf{c}}_n^{(1)}; \dots; \tilde{\mathbf{c}}_n^{(K)}] \in \mathbb{R}^{Kd}$, or averaged, $\mathbf{z}_n = \frac{1}{K} \sum_{k=1}^K \tilde{\mathbf{c}}_n^{(k)}$.

In all four cases, the fused representation is fed to a small MLP classifier, which produces the routing logits

$$\mathbf{s}_n = W_2 \text{GELU}(W_1 \mathbf{z}_n + \mathbf{b}_1) + \mathbf{b}_2 \in \mathbb{R}^K. \quad (5)$$

The corresponding routing probabilities are obtained by a softmax

$$w_{n,k} = \frac{\exp(s_{n,k})}{\sum_{m=1}^K \exp(s_{n,m})}, \quad k = 1, \dots, K. \quad (6)$$

The router selects the expert with the highest predicted probability, as summarized in Algorithm 1.

D. Training Objective

The router parameters θ are learned offline from a labeled dataset

$$\mathcal{D} = \{(x_n, R_n)\}_{n=1}^N$$

of audio clips and their reference transcriptions. Before training, each expert A_k is run on every clip to obtain the candidate transcription $Y_{n,k} = A_k(x_n)$ and the per-clip WER, and the encoder hidden states $\{\mathbf{H}_n^{(k)}\}$ are cached in the same pass. During router training, both the WER scores and the encoder features are treated as fixed inputs. The per-clip WER entering the objective is capped at 100%, so that a rare hallucination loop, whose WER is unbounded, cannot dominate the supervision of the router.

Given these fixed supervision signals and cached encoder representations, we optimize the router with a composite objective consisting of three loss terms. The first is a primary weighted-WER loss that directly minimizes the expected transcription error under the router's distribution:

$$\mathcal{L}_{\text{wer}}(\theta) = \frac{1}{N} \sum_{n=1}^N \sum_{k=1}^K w_{n,k} \text{WER}_{n,k}. \quad (7)$$

This loss directly aligns the router with task-level performance, but the gradient that a single sample provides is weak when all of its $\text{WER}_{n,k}$ values are similar in magnitude.

To strengthen the supervision early in training, we add an auxiliary hard cross-entropy (CE) term against the per-clip oracle expert, $j_n^* = \arg \min_k \text{WER}_{n,k}$, with label smoothing $\varepsilon \in [0, 1]$:

$$\mathcal{L}_{\text{hard}}(\theta) = -\frac{1}{N} \sum_{n=1}^N \sum_{k=1}^K \tilde{q}_{n,k} \log w_{n,k}, \quad (8)$$

where

$$\tilde{q}_{n,k} = \begin{cases} 1 - \varepsilon + \varepsilon/K, & \text{if } k = j_n^*, \\ \varepsilon/K, & \text{otherwise.} \end{cases}$$

This term provides a stronger gradient but ignores how much better the oracle expert is than the others.

To preserve the WER-magnitude information while keeping a strong gradient, we also use a soft CE term against a temperature-controlled target distribution derived from the WER vector,

$$q_{n,k}^{\text{soft}} = \frac{\exp(-\text{WER}_{n,k}/\tau)}{\sum_{m=1}^K \exp(-\text{WER}_{n,m}/\tau)}, \quad (9)$$

$$\mathcal{L}_{\text{soft}}(\theta) = -\frac{1}{N} \sum_{n=1}^N \sum_{k=1}^K q_{n,k}^{\text{soft}} \log w_{n,k}. \quad (10)$$

Small τ makes the target nearly hard, while larger τ makes it smoother and more sensitive to the relative WER differences between experts.

The overall training objective is a weighted sum

$$\min_{\theta} \mathcal{L}(\theta) = \lambda_{\text{wer}} \mathcal{L}_{\text{wer}}(\theta) + \lambda_{\text{hard}} \mathcal{L}_{\text{hard}}(\theta) + \lambda_{\text{soft}} \mathcal{L}_{\text{soft}}(\theta), \quad (11)$$

with non-negative coefficients $(\lambda_{\text{wer}}, \lambda_{\text{hard}}, \lambda_{\text{soft}})$ that determine which combination of the three signals is used. The objective is a hyperparameter of the router: the search of Section IV-B

chooses among four instances of (11), namely, hard CE alone, soft CE alone, the WER term with hard CE, and the full composite objective, with $\tau = 0.1$. The WER term alone is not among them, because a router trained with it collapses onto a single expert (Section IV-E).

Remark. Although WER is used as the primary error measure because of its widespread use in speech recognition research [52], [16], any non-negative per-clip error metric, such as character error rate (CER) or a domain-specific score, can be substituted into (7)–(10) without modifying the router architecture.

The complete offline training procedure is summarized in Algorithm 2.

Algorithm 2 Offline Training of the Hierarchical Router

Inputs

- $\{(x_n, R_n)\}_{n=1}^N$: Labeled dataset of audio clips and references
- $\mathcal{A} = \{A_1, \dots, A_K\}$ and their encoders $\{E_1, \dots, E_K\}$
- Loss coefficients $(\lambda_{\text{wer}}, \lambda_{\text{hard}}, \lambda_{\text{soft}})$, label smoothing ε , temperature τ

Output

Trained router g_{θ}

Training Steps

- 1: Split dataset into training and validation sets: $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}$
 - 2: **for** $n \leftarrow 1$ **to** N **do**
 - 3: **for** $k \leftarrow 1$ **to** K **do**
 - 4: Cache encoder features: $\mathbf{H}_n^{(k)} \leftarrow E_k(x_n)$
 - 5: Decode and score: $Y_{n,k} \leftarrow A_k(x_n)$; $\text{WER}_{n,k} \leftarrow \text{WER}(Y_{n,k}, R_n)$
 - 6: **end for**
 - 7: **end for**
 - 8: **for** each training step **do**
 - 9: Sample a minibatch $\mathcal{B} \subset \mathcal{D}_{\text{train}}$
 - 10: Compute routing probabilities $\mathbf{w}_n$ from $\{\mathbf{H}_n^{(k)}\}$ using (2)–(6)
 - 11: Compute soft target q_n^{soft} via (9)
 - 12: Compute composite loss $\mathcal{L}(\theta)$ on $\mathcal{B}$ via (11)
 - 13: Update θ with a gradient-based optimizer (e.g., AdamW)
 - 14: **end for**
 - 15: Perform model selection and early stopping using $\mathcal{L}_{\text{wer}}$ on $\mathcal{D}_{\text{val}}$
-

IV. EXPERIMENTS

This section first verifies on synthetic data that the router can exploit temporal structure when that structure carries the routing signal (Section IV-A). It then describes the evaluation on four real corpora (Section IV-B) and reports the main results (Section IV-C), the routing behavior (Section IV-D), the ablation of the design switches (Section IV-E), and the cost, complexity, and scalability of the router (Section IV-F), before discussing the scope of the method (Section IV-G).

A. Synthetic Feasibility Check

1) *Motivation:* The central design choice in this work is to keep the per-expert encoder outputs at frame-level resolution rather than collapsing them into a single pooled vector. A controlled test of this choice requires a setting in which the optimal expert for a clip is determined by the temporal structure of the clip itself rather than by any pooled summary of it. In the synthetic setting below, pooled summaries are insufficient by construction; the experiment is therefore a

feasibility check, which verifies that the router can exploit temporal structure when such structure carries the routing signal.

2) *Synthetic Data Generation*: We construct synthetic datasets of $N = 10,000$ clips, each with a fixed length of $T = 128$ frames, for pools of $K = 3, 5$, and 10 experts. Each clip x_n is divided into two segments separated by a regime switch at frame ρ_n , drawn uniformly between $T/4$ and $3T/4$. The first segment is drawn from an acoustic regime $r_n^{(1)}$ and the second from a different regime $r_n^{(2)} \neq r_n^{(1)}$, both from a set of $R = 4$ regimes. For each regime, we fix a center vector μ_r , and each frame is generated as

$$\mathbf{h}_{n,t}^{(k)} = W_k(\mu_{r_{n,t}} + \eta_{n,t}), \quad \eta_{n,t} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I),$$

where W_k is an expert-specific random projection that simulates the hidden-state space of expert A_k , with widths D_k of 1024 or 512, and $r_{n,t} = r_n^{(1)}$ for $t \leq \rho_n$ and $r_{n,t} = r_n^{(2)}$ otherwise.

We assign per-expert WER scores using a fixed table $\text{WER}_{r_1, r_2, k}$ that depends only on the ordered pair of regimes within the clip. For every ordered pair, one expert has a low WER and the others have a high one. The table is constructed so that no expert is uniformly best and the identity of the best expert depends on both segments of the clip and, in general, on their order, which a temporal average of the frames does not reveal. We further add small Gaussian noise to each $\text{WER}_{n,k}$ to make the optimization non-trivial.

3) *Models Compared*: We compare HIT-ASR with MLP-pool, a router that averages each expert’s frame sequence over time and feeds the concatenated averages to a multilayer perceptron (Section IV-B); with the best single expert on the training split; and with the oracle that selects the per-clip $\arg \min_k \text{WER}_{n,k}$. Both routers are trained with the same objective and optimizer on an 80/20 training/test split, with validation clips drawn from the training split, and every configuration is repeated with three seeds.

4) *Results*: Table I reports the test WER, the selection accuracy, and the oracle gap, defined as $\text{WER}_{\text{method}} - \text{WER}_{\text{oracle}}$. At every pool size, HIT-ASR identifies the oracle expert from the frame sequence on every test clip, so that its oracle gap is zero, whereas MLP-pool, which reads the same frames averaged over time, reaches 58.1, 51.3, and 50.4% selection accuracy at $K = 3, 5$, and 10, respectively. As the pool grows, each expert is best on a smaller share of the clips, and the WER of the best single expert rises from 22.2 to 29.1%.

B. Experimental Setup on Real Corpora

1) *Corpora*: We evaluate on four English corpora that differ in domain and recording conditions (Table II): AMI-SDM [53], meetings recorded with a single distant microphone; Earnings-22 [54], earnings calls with accented and spontaneous speech; People’s Speech [55], diverse, openly licensed recordings collected from the Internet; and AfriSpeech-200 [56], African-accented English in the clinical and general domains. A clip is an utterance-level segment of a recording; the mean clip length ranges from 2.6 to 10.3 s across the corpora. For every corpus, Table II lists the numbers of clips,

TABLE I
SYNTHETIC REGIME-SWITCH CHECK WITH $K = 3, 5$, AND 10 EXPERTS:
TEST WER (%), MEAN $\pm$ SD OVER THREE SEEDS), SELECTION ACCURACY
(%), AND ORACLE GAP ($\text{WER}_{\text{METHOD}} - \text{WER}_{\text{ORACLE}}$, IN WER POINTS).

K	System	WER	Sel. acc.	Oracle gap
3	Best single expert	22.24 $\pm$ 0.41	42.3	17.76
	MLP-pool	17.10 $\pm$ 0.79	58.1	12.61
	HIT-ASR	4.49 $\pm$ 0.27	100.0	0.00
	Oracle	4.49 $\pm$ 0.27	100.0	0.00
5	Best single expert	25.06 $\pm$ 1.15	34.0	20.20
	MLP-pool	19.14 $\pm$ 0.42	51.3	14.28
	HIT-ASR	4.86 $\pm$ 0.43	100.0	0.00
	Oracle	4.86 $\pm$ 0.43	100.0	0.00
10	Best single expert	29.10 $\pm$ 0.25	18.4	24.61
	MLP-pool	19.76 $\pm$ 0.48	50.4	15.28
	HIT-ASR	4.49 $\pm$ 0.33	100.0	0.00
	Oracle	4.49 $\pm$ 0.33	100.0	0.00

hours, and reference words, the size of the hyperparameter hold-out, the number of evaluation clips, and the training and test sizes of every fold. Hypotheses and references are normalized with the English text normalizer of Whisper [13] before the errors are counted.

2) *Experts*: Each corpus has its own trio of experts, drawn from seven open ASR systems of five decoder families (Table III): Cohere-Transcribe [57], Granite-Speech-4.1-2b [58], Parakeet-TDT-0.6b-v3 [59], Kyutai-STT-2.6b [60], Parakeet-CTC-1.1b [61], [62], Qwen3-ASR-1.7B [15], and Voxtral-Mini-3B [14]. Their encoder widths range from 512 to 1280, their frame rates range from 12.5 to 50 frames per second, and five of them decode with an autoregressive transformer decoder: an attention decoder, a decoder-only transformer, or a large language model. Each trio consists of complementary experts, for which the per-clip oracle leaves headroom over the best single expert (Table II). On Earnings-22 and AfriSpeech-200, the three experts lie within 1.1 and 1.0 WER points of one another, respectively, so that no static choice is clearly preferable. Every expert is run once over its corpus to cache its transcripts, per-clip errors, and encoder states.

3) *Baselines*: The *best single expert* is the expert with the lowest WER on the training half of each fold, and the *oracle* selects, for every clip, the expert with the lowest WER on that clip. *Random* selects an expert uniformly at random for every clip and is reported as a reference. *ROVER* with confidence-weighted voting [28] and confusion-network minimum-Bayes-risk combination (*CN-MBR*) [29], [30] combine the hypotheses of the three experts after all of them have decoded; their weighting, confidence sources, and null-word penalties are tuned on the hold-out, as is every other model. *MLP-pool* is a pooled-feature router: it averages each expert’s frame-level encoder states over time, concatenates the averages with the logarithm of the clip length, and maps them to a routing distribution with a multilayer perceptron. It is trained with the same code as HIT-ASR, with the same optimizer, schedule, early-stopping rule, weight averaging, and ensemble of three seeded fits, and with the full composite objective of Eq. (11); its hyperparameters are selected on the same hold-out under the same budget. The objective of HIT-ASR is selected on

TABLE II

CORPORA, EXPERT TRIOS, AND PARTITIONS. A DISJOINT 25% HOLD-OUT OF EACH CORPUS IS USED ONLY FOR HYPERPARAMETER, OBJECTIVE, AND ARCHITECTURE SELECTION; THE REMAINING CLIPS ARE EVALUATED BY 5×2 CROSS-VALIDATION (FIVE RANDOM HALVINGS, EACH HALF USED ONCE FOR TRAINING AND ONCE FOR TESTING). EXPERT WERS (%) ARE MEANS OVER THE TEN TEST FOLDS. BSM: BEST SINGLE EXPERT.

	AMI-SDM	Earnings-22	People's Speech	AfriSpeech-200
Domain	meetings, single distant microphone	earnings calls, accented and spontaneous	Internet-sourced, noisy and diverse	African-accented English, clinical and general
Experts (WER)	Cohere-Transcribe (18.42) Granite-Speech-4.1-2b (19.57) Parakeet-TDT-0.6b-v3 (26.23)	Cohere-Transcribe (13.88) Kyutai-STT-2.6b (12.79) Parakeet-TDT-0.6b-v3 (13.74)	Granite-Speech-4.1-2b (17.39) Parakeet-CTC-1.1b (18.39) Parakeet-TDT-0.6b-v3 (18.77)	Cohere-Transcribe (25.90) Qwen3-ASR-1.7B (25.23) Voxtral-Mini-3B (26.25)
Clips	41,641	12,088	14,299	9,544
Hours	29.7	25.9	28.2	27.4
Mean clip length (s)	2.6	7.7	7.1	10.3
Reference words	315,864	217,118	286,169	156,221
Hold-out clips (tuning)	10,410	3,022	3,575	2,386
Evaluation clips (5×2 cv)	31,231	9,066	10,724	7,158
Train / test clips per fold	15,615 / 15,616	4,533 / 4,533	5,362 / 5,362	3,579 / 3,579
Oracle headroom above the BSM (WER points)	6.7	3.8	4.6	8.3

TABLE III

THE SEVEN EXPERTS: HUGGING FACE CHECKPOINT, DECODER FAMILY, WIDTH d AND FRAME RATE OF THE ENCODER STATES THAT THE ROUTER READS, AND THE CORPORA ON WHICH EACH EXPERT IS USED.

Expert	Checkpoint	Decoder family	d	Hz	Used on
Cohere-Transcribe	CohereLabs/cohere-transcribe-03-2026	attention encoder-decoder (FastConformer, Transformer decoder)	1280	12.5	AMI-SDM, Earnings-22, AfriSpeech-200
Granite-Speech-4.1-2b	ibm-granite/granite-speech-4.1-2b	Conformer encoder, Granite LLM decoder	1024	50.0	AMI-SDM, People's Speech
Parakeet-TDT-0.6b-v3	nvidia/parakeet-tdt-0.6b-v3	token-and-duration transducer (FastConformer)	1024	12.5	AMI-SDM, Earnings-22, People's Speech
Kyutai-STT-2.6b	kyutai/stt-2.6b-en-trfs	streaming decoder-only transformer on the Mimi codec	512	18.8	Earnings-22
Parakeet-CTC-1.1b	nvidia/parakeet-ctc-1.1b	CTC (FastConformer)	1024	12.5	People's Speech
Qwen3-ASR-1.7B	Qwen/Qwen3-ASR-1.7B-hf	audio encoder, Qwen3 LLM decoder	1024	13.0	AfriSpeech-200
Voxtral-Mini-3B	mistralai/Voxtral-Mini-3B-2507	Whisper-style encoder, Mistral LLM decoder	1280	50.0	AfriSpeech-200

the hold-out from the four instances of the same composite loss (Section III-D). *ADASTT* [32] is reimplemented within the same code base as gradient-boosted trees [63] over the same pooled encoder states of the experts as MLP-pool, so that it runs on the same clips, experts, hold-out, and folds as HIT-ASR; its hyperparameters are selected on the same hold-out under the same budget.

4) *Protocol*: Every model is tuned on a hold-out of 25% of the clips of each corpus and evaluated by 5×2 cross-validation [64] on the remaining clips: the evaluation clips are split into two halves at random five times, and each half is used once for training and once for testing, so that every model is trained and tested on the same ten folds. Every result is the mean $\pm$ standard deviation, over these ten folds, of the corpus WER, i.e., the total number of word errors divided by the total number of reference words. Every design choice, for HIT-ASR and for every baseline alike, is made on the hold-out: the hyperparameters, the objective and its coefficients, the fusion mechanism, the sharing of Stage 1, and the input normalization. The search splits the hold-out into fitting and validation clips (30% validation), runs a random search over each model's search space (Table S1 of the supplementary material) with the same budget for every model, and keeps the configuration with the lowest validation WER. This configuration is then frozen, trained on the training half of every fold, and tested on its test half. The evaluation folds are used only for testing.

Significance test. Every comparison with HIT-ASR is tested with the paired 5×2 cv t test [64] on the corpus WER of the ten folds. Let $d_{i,j}$ be the WER difference between a system and HIT-ASR on fold j of repetition i , and let s_i^2 be the variance of the two differences of repetition i . The statistic $t = d_{1,1} / \sqrt{\frac{1}{5} \sum_{i=1}^5 s_i^2}$ follows a t distribution with five degrees of freedom under the null hypothesis, from which a two-sided p -value is computed.

Tables IV and V and the timing of Table VII are computed on these test folds, whereas the ablation of Table VI is computed on the validation clips of the hold-out and reports shares of the oracle gap rather than WERs.

5) *Training Details*: Both neural routers (HIT-ASR and MLP-pool) are trained with AdamW, a linear warm-up of 200 steps followed by cosine decay, gradient clipping at 1.0, and mixed precision. An exponential moving average of the weights over two epochs is used for validation and as the final model. Training stops early on the expected validation WER, i.e., $\mathcal{L}_{\text{wer}}$ on a validation subset drawn from the training half of the fold, after a minimum of ten epochs. Each model is an ensemble of three fits with different seeds, whose routing distributions are averaged. All experiments are run on NVIDIA RTX PRO 6000 Blackwell Server Edition GPUs.

C. Main Results

Table IV reports the corpus WER of every system together with the p -value of the paired test against HIT-ASR. HIT-ASR attains the lowest WER on every corpus. Relative to the best single expert, which is the system that would be deployed without a router, HIT-ASR reduces the WER by 6.7, 6.3, 9.1, and 11.2% relative on AMI-SDM, Earnings-22, People's Speech, and AfriSpeech-200, respectively, and the improvement is statistically significant on every corpus.

The comparison with MLP-pool isolates the contribution of the temporal structure: MLP-pool reads the same encoder states of the same experts, averaged over time, and shares the training procedure, ensemble, hold-out, and budget of HIT-ASR. MLP-pool itself clearly outperforms the best single expert, and HIT-ASR improves significantly on MLP-pool on all four corpora, with fewer parameters on three of them (24.5, 11.9, and 2.0M versus 45.7, 15.0, and 44.1M on AMI-SDM, Earnings-22, and People's Speech, respectively). HIT-ASR also improves on *ADASTT*, the tree-based selector over

TABLE IV

CORPUS WER (%), MEAN $\pm$ SD OVER THE TEN TEST FOLDS OF THE 5×2 CROSS-VALIDATION) AND THE p -VALUE OF THE PAIRED 5×2 CV t TEST AGAINST HIT-ASR ON THE SAME FOLDS (SECTION IV-B). THE BEST SINGLE EXPERT IS SELECTED ON THE TRAINING HALF OF EACH FOLD; THE ORACLE SELECTS, FOR EVERY CLIP, THE EXPERT WITH THE LOWEST WER. ALL SELECTORS AND ENSEMBLES ARE TUNED ON THE DISJOINT HOLD-OUT. BOLD: LOWEST WER AMONG THE NON-ORACLE SYSTEMS.

System	AMI-SDM		Earnings-22		People's Speech		AfriSpeech-200	
	WER	p	WER	p	WER	p	WER	p
Best single expert	18.42 $\pm$ 0.22	0.008	12.79 $\pm$ 0.24	0.003	17.39 $\pm$ 0.26	0.001	25.82 $\pm$ 1.03	0.047
Random	21.42 $\pm$ 0.48	<0.001	13.51 $\pm$ 0.32	0.004	18.32 $\pm$ 0.89	0.067	25.69 $\pm$ 0.94	0.012
ROVER (conf.-weighted)	20.03 $\pm$ 0.37	<0.001	13.60 $\pm$ 0.36	<0.001	18.15 $\pm$ 0.23	<0.001	31.17 $\pm$ 1.27	0.002
CN-MBR	23.99 $\pm$ 0.44	<0.001	14.06 $\pm$ 0.40	0.003	19.12 $\pm$ 1.17	0.036	31.71 $\pm$ 1.36	0.002
MLP-pool	17.44 $\pm$ 0.21	0.034	12.14 $\pm$ 0.36	0.040	16.29 $\pm$ 0.32	0.033	23.71 $\pm$ 0.79	0.045
ADASTT	18.23 $\pm$ 0.22	0.010	12.82 $\pm$ 0.33	0.016	16.78 $\pm$ 0.23	0.008	24.84 $\pm$ 1.36	0.154
HIT-ASR	17.19 $\pm$ 0.30	—	11.98 $\pm$ 0.27	—	15.80 $\pm$ 0.19	—	22.93 $\pm$ 0.59	—
Oracle	11.68 $\pm$ 0.13	—	9.00 $\pm$ 0.18	—	12.74 $\pm$ 0.13	—	17.51 $\pm$ 0.38	—

TABLE V

ROUTING BEHAVIOR ON THE TEN TEST FOLDS: SHARE OF CLIPS (%) ROUTED TO EACH EXPERT OF THE TRIO, AND SELECTION ACCURACY (%), MEAN $\pm$ SD OVER THE FOLDS), I.E., THE SHARE OF CLIPS ROUTED TO AN EXPERT WITH THE LOWEST WER ON THAT CLIP.

Router	E1	E2	E3	Sel. acc.
<i>AMI-SDM</i> : E1–E3 = Cohere / Granite / Parakeet-TDT				
MLP-pool	74.8	21.0	4.3	78.4 $\pm$ 0.5
ADASTT	96.1	3.8	0.1	74.9 $\pm$ 0.4
HIT-ASR	39.7	54.1	6.2	79.7 $\pm$ 0.6
Oracle	73.6	20.8	5.6	100.0
<i>Earnings-22</i> : E1–E3 = Cohere / Kyutai / Parakeet-TDT				
MLP-pool	66.7	18.2	15.1	73.1 $\pm$ 0.6
ADASTT	89.8	2.6	7.6	68.9 $\pm$ 0.8
HIT-ASR	73.5	12.0	14.6	73.2 $\pm$ 0.7
Oracle	64.1	20.9	15.0	100.0
<i>People's Speech</i> : E1–E3 = Granite / Parakeet-CTC / Parakeet-TDT				
MLP-pool	69.8	15.1	15.1	70.5 $\pm$ 0.7
ADASTT	91.7	6.6	1.7	69.2 $\pm$ 0.7
HIT-ASR	45.5	22.1	32.4	71.6 $\pm$ 0.4
Oracle	67.3	18.3	14.3	100.0
<i>AfriSpeech-200</i> : E1–E3 = Cohere / Qwen3-ASR / Voxtral				
MLP-pool	63.1	24.4	12.6	62.9 $\pm$ 0.6
ADASTT	87.0	10.0	3.0	61.6 $\pm$ 0.5
HIT-ASR	52.8	27.6	19.6	63.4 $\pm$ 0.9
Oracle	60.6	24.9	14.4	100.0

the same pooled states, on all four corpora, by 5.7, 6.5, 5.8, and 7.7% relative. With these trios, the hypothesis-level ensembles, which decode every expert before combining, have a higher WER than the best single expert, and HIT-ASR improves significantly on both on every corpus while decoding a single expert per clip.

D. Routing Behavior

Table V reports how each router distributes the clips over the experts, together with its selection accuracy (the share of clips routed to an expert with the lowest WER on that clip). Neither pooled router collapses onto one expert: MLP-pool sends between 63 and 75% of the clips to its most frequent choice, and ADASTT concentrates more strongly on one expert. HIT-ASR reaches the highest selection accuracy of all routers on every corpus (79.7, 73.2, 71.6, and 63.4%).

Because many clips are transcribed equally well by several experts, a router can depart from the oracle's distribution while still choosing a best expert. HIT-ASR does so on AMI-SDM, where it routes most clips to Granite-Speech-4.1-2b, whereas the oracle's most frequent choice is Cohere-Transcribe.

E. Ablation of the Design Switches

The design switches of HIT-ASR (the fusion mechanism, the sharing of Stage 1, the objective, and the input normalization) are hyperparameters of the search. We estimate their effect from an exploration of the router's search space on the hold-outs of AMI-SDM, Earnings-22, and People's Speech, in which 60 configurations per corpus are drawn at random with every switch free, and each configuration is fitted once on the fitting clips of the hold-out. For every level of every switch, Table VI reports the share of configurations that collapse onto the best single expert, the median and the best share of the oracle gap closed on the validation clips of the hold-out, and the median parameter count.

Over the configurations that do not collapse, the four fusion mechanisms close a similar share of the oracle gap: the median share closed by the cross-attention bridge is 13–21, 29–33, and 21–38% on AMI-SDM, Earnings-22, and People's Speech, respectively (range over three seeds), versus –3–17, 27–33, and 34–44% for the other mechanisms. The mechanisms differ in reliability: the bridge, the largest mechanism, collapses in 36–57% of its configurations with labeled sets of this size, versus 17–50% for the others, and the search selects a simpler mechanism on every corpus. The fusion mechanism is therefore best treated as a hyperparameter: with a few thousand labeled clips, the simpler mechanisms are the more reliable choice, and the bridge remains an option of the framework.

The capacity of Stage 1 matters: per-expert Stage 1 weights double the size of the router (3.3 versus 1.7M parameters at the median on AMI-SDM) and raise the median share of the gap closed on all three corpora (7 versus 5%, 28 versus 22%, and 35 versus 23%), while the best configurations of the two levels are close. The search selects per-expert weights on three of the four corpora.

The WER term alone leads to collapse. Its gradient is weak early in training, when the routing distribution is close to

TABLE VI

DESIGN SWITCHES OF HIT-ASR, ESTIMATED FROM THE EXPLORATION OF ITS HYPERPARAMETER SPACE ON THE HOLD-OUTS OF AMI-SDM, EARNINGS-22, AND PEOPLE’S SPEECH; NO EVALUATION CLIP IS USED. FOR EVERY LEVEL OF A SWITCH, OVER ALL OTHER HYPERPARAMETERS: NUMBER OF CONFIGURATIONS (n), SHARE OF CONFIGURATIONS THAT COLLAPSE ONTO THE BEST SINGLE EXPERT (COLL., CLOSING AT MOST 1% OF ITS GAP TO THE ORACLE), MEDIAN AND BEST SHARE OF THAT GAP CLOSED (%) ON THE VALIDATION CLIPS OF THE HOLD-OUT, AND MEDIAN NUMBER OF PARAMETERS (PAR., M).

Switch	Level	AMI-SDM				Earnings-22				People’s Speech				Par. M
		n	coll.	med.	best	n	coll.	med.	best	n	coll.	med.	best	
Fusion	cross-attention bridge	14	50	5	22	14	57	0	34	14	36	14	44	3.2
	concatenation	6	50	1	10	6	17	27	41	6	17	27	43	2.8
	averaging	18	22	8	28	18	28	24	42	18	17	29	59	2.7
	self-attention	22	36	5	25	22	36	27	41	22	27	34	53	1.8
Stage 1	per-expert	27	26	7	26	27	37	28	42	27	26	35	59	3.3
	shared	33	45	5	28	33	36	22	41	33	24	23	55	1.6
Objective	WER + hard + soft CE	7	43	11	16	7	14	31	35	7	14	26	43	3.2
	WER + soft CE	6	33	11	25	6	50	2	35	6	33	42	52	2.4
	hard CE	9	56	0	5	9	22	22	37	9	11	22	37	1.9
	soft CE	10	20	7	14	10	0	30	42	10	0	42	59	1.8
	WER + hard CE	8	0	9	14	8	12	31	41	8	12	41	52	2.2
	WER + soft CE ($\tau=1$)	8	12	19	21	8	38	5	30	8	0	23	54	1.6
	WER only	12	75	−21	28	12	100	0	0	12	83	0	25	3.4
Input	raw	27	37	6	25	27	44	5	41	27	26	21	54	2.1
	scaled	17	47	5	21	17	29	29	42	17	29	28	55	2.4
	standardized	16	25	6	28	16	31	27	35	16	19	36	59	2.8

uniform and the experts’ WERs differ little on most clips, so that the router converges to the expert that is best on average: 75–100% of the WER-only configurations collapse onto the best single expert, versus 0–20% with soft CE, and the median WER-only configuration routes 100% of the AMI-SDM clips to a single expert. This objective is therefore not among those searched in the main experiments. Soft CE alone and the WER term with hard CE are the objectives that collapse least often, and the search selects one of them on every corpus. Normalizing the inputs raises the median share of the gap closed on Earnings-22 and People’s Speech, while the best configurations of the three input levels are close.

F. Cost, Complexity, and Scalability

Table VII reports the cost of every pipeline measured end to end on raw audio, one clip at a time on one RTX PRO 6000 Blackwell Server Edition GPU: the encoder and decoder times of every expert; HIT-ASR with all encoders, the router, and the selected expert’s decoder; and the decoding of all experts that every hypothesis-level ensemble requires. For the five experts with an autoregressive transformer decoder, the decoder takes 46–99% of the pipeline’s time. HIT-ASR incurs the cost of every encoder and of its router (5.6 to 8.2 ms per clip), but of only one decoder. It needs 40 to 90% less time than decoding all experts, and it is faster than the best single expert on Earnings-22, People’s Speech, and AfriSpeech-200 (0.11, 0.73, and 0.49 times its time, respectively), because it routes many clips away from an expensive autoregressive decoder. On AMI-SDM, whose best single expert is one of the fastest pipelines, HIT-ASR takes 2.52 times as long.

The complexity of each stage follows from the architecture. The encoder cost is linear in K and is incurred on every clip, whereas the decoder cost is that of one expert, independent of K . Stage 1 applies self-attention within each expert’s sequence

of at most T frames, at a cost of $O(KT^2d)$ for router width d . The fusion operates on K summary vectors, except for the cross-attention bridge, which attends from each summary to the frames of the other experts at a cost of $O(K^2Td)$. On AMI-SDM, the router has 24.5M parameters over its three seeded fits and needs 0.72 GFLOPs and 7.7 ms per clip, versus 51.3 ms for the three encoders.

The supervision of the router requires one labeling pass, in which every expert transcribes every labeled clip once and its encoder states are cached. Estimated from the experts’ measured throughput, this pass takes 0.4, 2.1, 0.2, and 0.5 GPU-hours on AMI-SDM, Earnings-22, People’s Speech, and AfriSpeech-200, respectively. The pass is parallel across experts and clips and is amortized over all router training runs; training the three-seed ensemble on the cached states then takes 2 to 30 min per fold. The router is trained for a fixed pool: adding an expert requires retraining the router, but only the new expert has to be run over the labeled clips, and no expert is fine-tuned.

The cost of the router grows moderately with the pool size, because only the per-expert projections and Stage 1 grow with K , while the fusion operates on K summary vectors. From $K = 3$ to $K = 10$, one fit of the router selected for People’s Speech grows from 0.7 to 1.6M parameters and from 1.7 to 5.1 ms per clip, and on the synthetic task, HIT-ASR selects the oracle expert at $K = 3, 5$, and 10 (Table I). The cost that grows linearly with K is that of the experts’ encoders, which run on every clip, together with the memory that holds the experts (8.0 to 17.6 GB of weights for the trios studied here). For large pools, natural remedies are to prune the pool to complementary experts or to use a cascade that runs additional encoders only when the first ones are uncertain.

TABLE VII

MEASURED COST PER CLIP ON RAW AUDIO, ONE CLIP AT A TIME ON ONE RTX PRO 6000 BLACKWELL SERVER EDITION GPU, ON CLIPS OF A TEST FOLD: ENCODER (ENC.), ROUTER, DECODER (DEC.), AND TOTAL TIME (MS), REAL-TIME FACTOR (RTF), TIME RELATIVE TO THE BEST SINGLE EXPERT ($\times$ BEST), FLOATING-POINT OPERATIONS OF THE ROUTER PER CLIP (GFLOPs), AND GPU MEMORY HELD BY THE EXPERTS' WEIGHTS (GB). HIT-ASR RUNS EVERY ENCODER, THE ROUTER, AND THE DECODER OF THE SELECTED EXPERT, WHICH REUSES ITS ENCODER PASS; "DECODE ALL + ROVER" RUNS EVERY EXPERT'S PIPELINE BEFORE COMBINING. FOR A SINGLE EXPERT, ENC. AND DEC. TIME THE SPLIT PATH THAT HIT-ASR USES (THE ENCODER PASS WHOSE STATES THE ROUTER READS, THEN DECODING FROM THAT PASS), WHEREAS TOTAL TIMES THE EXPERT'S OWN UNMODIFIED PIPELINE IN A SEPARATE RUN, SO THAT ENC. AND DEC. NEED NOT ADD UP TO TOTAL.

Corpus	System	Enc.	Router	Dec.	Total	RTF	$\times$ best	GFLOPs	Weights
AMI-SDM	Cohere-Transcribe (best single)	25.9	–	21.8	42.2	0.020	1.00	–	4.1
	Granite-Speech-4.1-2b	16.7	–	87.0	96.9	0.047	2.29	–	4.6
	Parakeet-TDT-0.6b-v3	11.1	–	17.0	37.4	0.018	0.89	–	1.3
	HIT-ASR	51.3	7.7	47.4	106	0.051	2.52	0.72	10.0
	Decode all + ROVER	–	–	178	178	0.086	4.21	–	10.0
Earnings-22	Cohere-Transcribe	27.5	–	46.2	67.5	0.010	0.03	–	4.1
	Kyutai-STT-2.6b (best single)	15.5	–	1,964	1,965	0.281	1.00	–	5.6
	Parakeet-TDT-0.6b-v3	12.6	–	36.5	57.5	0.008	0.03	–	1.3
	HIT-ASR	49.8	6.2	161	217	0.031	0.11	1.05	11.0
	Decode all + ROVER	–	–	2,106	2,106	0.301	1.07	–	11.0
People's Speech	Granite-Speech-4.1-2b (best single)	18.0	–	205	215	0.035	1.00	–	4.6
	Parakeet-CTC-1.1b	25.2	–	0.3	22.5	0.004	0.10	–	2.1
	Parakeet-TDT-0.6b-v3	13.0	–	36.1	57.2	0.009	0.27	–	1.3
	HIT-ASR	51.1	5.6	99.8	157	0.025	0.73	3.01	8.0
	Decode all + ROVER	–	–	298	299	0.048	1.39	–	8.0
AfriSpeech-200	Cohere-Transcribe	30.5	–	61.9	82.3	0.006	0.30	–	4.1
	Qwen3-ASR-1.7B (best single)	12.0	–	264	271	0.021	1.00	–	4.1
	Voxtral-Mini-3B	24.7	–	281	302	0.024	1.11	–	9.4
	HIT-ASR	65.5	8.2	59.3	133	0.010	0.49	5.82	17.6
	Decode all + ROVER	–	–	658	659	0.052	2.43	–	17.6

G. Discussion and Limitations

The synthetic check shows that the router can exploit temporal structure when that structure carries the routing signal. The four corpora show that it also does so under realistic conditions: on all of them, HIT-ASR improves significantly on MLP-pool, a router that reads the same encoder states averaged over time.

HIT-ASR selects one expert for the whole clip: it uses the temporal structure within the clip to inform a clip-level decision and does not switch experts within a clip, so that the clip-level oracle is the ceiling of the method. On AMI-SDM, Earnings-22, and People's Speech, a clip is itself an utterance-level segment of a longer recording, so that the selected expert can change from one segment of a recording to the next. Routing within a clip is a natural extension, in which a router chooses an expert for every segment with a cost for each switch; the clip-level router is the special case with an infinite switching cost. Switching within a clip is not free, however. First, the experts recognize each word in the context of the surrounding segment, and recognition benefits from longer context [65]. Second, joining the hypotheses of different experts at every cut requires word-level timestamps, which attention-based decoders such as Whisper do not provide natively and which must be recovered with an external forced aligner [66]. Third, even a single model that transcribes long audio in consecutive windows needs dedicated heuristics at the window boundaries [13]. In this study, the baselines that combine experts within a clip, word by word, are ROVER and CN-MBR, and with the experts of these trios, neither improves on the best single expert (Table IV).

V. REPRODUCIBILITY

The code, the data, and a reproducibility package are publicly available. The repository <https://github.com/huseyin-karaca/hit-asr> contains the router, every baseline, and one notebook per experiment of this paper, each of which runs on Google Colab, together with documentation and a test suite. The dataset <https://huggingface.co/datasets/huseyin-karaca/hit-asr> holds the experts' transcripts and per-clip errors, their frame-level encoder states, the hyperparameter searches, and the results of every fold; Docker images for CPU and GPU run the same notebooks on any machine. The experiments can be reproduced at three levels: level 1 recomputes every table of this paper from the stored searches and folds on a CPU in minutes, level 2 retrains every model from the published encoder states, and level 3 rebuilds the experts' transcripts and encoder states from the raw audio and retrains every model on them. Run from a fresh download of the public package, level 1 reproduces every table exactly, and in our level-2 runs on the GPU type of the published records, every baseline was reproduced to the last digit. Neural routers retrained on a GPU are not guaranteed to be bit-identical to the published runs, because GPU training is not deterministic. The dataset card documents how the transcripts of each expert were produced.

VI. CONCLUSION

We introduced HIT-ASR, a router that selects, for each clip, one of several complete and frozen ASR systems from the frame-level encoder states of all of them, before any of them decodes. A per-expert temporal encoder summarizes each expert's frames at that expert's native frame rate, a fusion

step combines the summaries, and a WER-aware objective supervises the router with the measured error of every expert on every clip. On four corpora with trios of current open ASR systems, HIT-ASR attains the lowest WER on every corpus, 6.3 to 11.2% relative below that of the best single expert. Its improvements over the best single expert, over a pooled-feature router trained with the WER-aware objective, and over two hypothesis-level ensembles are statistically significant on every corpus, and it requires less computation time per clip than the best single expert on three of the four corpora. Several directions remain open: segment-level routing with a cost for each switch, trained on segment-level labels obtained by aligning the references to the segments; routers that generalize to unseen experts without retraining, for instance from an expert-agnostic description of each expert’s encoder states; pruning the pool to complementary experts; and a cascade that runs additional encoders only when the first ones are uncertain.

REFERENCES

- [1] G. Hinton *et al.*, “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *IEEE Signal Processing Magazine (SPM)*, vol. 29, no. 6, pp. 82–97, 2012, <https://doi.org/10/gc8z3r>.
- [2] G. E. Dahl, D. Yu, L. Deng, and A. Acero, “Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition,” *IEEE Transactions on Audio, Speech, and Language Processing (TASL)*, vol. 20, no. 1, pp. 30–42, 2012, <https://doi.org/10/bjdkzf>.
- [3] J. Deng *et al.*, “Confidence score based speaker adaptation of Conformer speech recognition systems,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 31, pp. 1175–1190, 2023, <https://doi.org/10/hcb4nc>.
- [4] A. Vaswani *et al.*, “Attention is all you need,” in *Conference on Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5998–6008, <https://papers.nips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845a-a-Abstract.html>.
- [5] A. Gulati *et al.*, “Conformer: Convolution-augmented transformer for speech recognition,” in *Annual Conference of the International Speech Communication Association (Interspeech)*, 2020, pp. 5036–5040, <https://doi.org/10/gjp97z>.
- [6] Y. Zhang *et al.*, “BigSSL: Exploring the frontier of large-scale semi-supervised learning for automatic speech recognition,” *IEEE Journal of Selected Topics in Signal Processing (JSTSP)*, vol. 16, no. 6, pp. 1519–1532, 2022, <https://doi.org/10/grzs8c>.
- [7] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, “wav2vec 2.0: A framework for self-supervised learning of speech representations,” in *Conference on Neural Information Processing Systems (NeurIPS)*, 2020, <https://proceedings.neurips.cc/paper/2020/hash/92d1e1eb1cd6f9fba3227870bb6d7f07-Abstract.html>.
- [8] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, “HuBERT: Self-supervised speech representation learning by masked prediction of hidden units,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 29, pp. 3451–3460, 2021, <https://doi.org/10/grgs2>.
- [9] S. Chen *et al.*, “WavLM: Large-scale self-supervised pre-training for full stack speech processing,” *IEEE Journal of Selected Topics in Signal Processing (JSTSP)*, vol. 16, no. 6, pp. 1505–1518, 2022, <https://doi.org/10/gsqzxc>.
- [10] A. Graves, “Sequence transduction with recurrent neural networks,” 2012, <https://doi.org/10/qvvn>.
- [11] X. Gong *et al.*, “Advanced long-content speech recognition with factorized neural transducer,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 32, pp. 1803–1815, 2024, <https://doi.org/10/hcb4nb>.
- [12] H. Lee, J. W. Yoon, S. Kim, and N. S. Kim, “Towards maximum likelihood training for transducer-based streaming speech recognition,” *IEEE Signal Processing Letters (SPL)*, vol. 32, pp. 26–30, 2025, <https://doi.org/10/g92p3f>.
- [13] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, “Robust speech recognition via large-scale weak supervision,” in *International Conference on Machine Learning (ICML)*, 2023, pp. 28 492–28 518, <https://proceedings.mlr.press/v202/radford23a.html>.
- [14] A. H. Liu *et al.*, “Voxtral,” 2025, <https://doi.org/10/hck8mw>.
- [15] X. Shi *et al.*, “Qwen3-ASR technical report,” 2026, <https://doi.org/10/hck8mt>.
- [16] J. Li, L. Deng, Y. Gong, and R. Haeb-Umbach, “An overview of noise-robust automatic speech recognition,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 22, no. 4, pp. 745–777, 2014, <https://doi.org/10/gg8q3b>.
- [17] S. Cornell *et al.*, “The CHiME-7 DASR challenge: Distant meeting transcription with multiple devices in diverse scenarios,” in *International Workshop on Speech Processing in Everyday Environments (CHiME)*, 2023, pp. 1–6, <https://doi.org/10/hcb4m7>.
- [18] S. Cornell *et al.*, “The CHiME-8 DASR challenge for generalizable and array agnostic distant automatic speech recognition and diarization,” in *International Workshop on Speech Processing in Everyday Environments (CHiME)*, 2024, pp. 1–6, <https://doi.org/10/hcb4m6>.
- [19] P. Kerschke, H. H. Hoos, F. Neumann, and H. Trautmann, “Automated algorithm selection: Survey and perspectives,” *Evolutionary Computation*, vol. 27, no. 1, pp. 3–45, 2019, <https://doi.org/10/gp3qvx>.
- [20] P. Bell, J. Fainberg, O. Klejch, J. Li, S. Renals, and P. Swietojanski, “Adaptation algorithms for neural network-based speech recognition: An overview,” *IEEE Open Journal of Signal Processing (OJSP)*, vol. 2, pp. 33–66, 2021, <https://doi.org/10/hcb4pb>.
- [21] A. Kumar, S. Singh, D. Gowda, A. Garg, S. Singh, and C. Kim, “Utterance confidence measure for end-to-end speech recognition with applications to distributed speech recognition scenarios,” in *Annual Conference of the International Speech Communication Association (Interspeech)*, 2020, pp. 4357–4361, <https://doi.org/10/hcb4pc>.
- [22] P. Swarup, R. Maas, S. Garimella, S. H. Mallidi, and B. Hoffmeister, “Improving ASR confidence scores for Alexa using acoustic and hypothesis embeddings,” in *Annual Conference of the International Speech Communication Association (Interspeech)*, 2019, pp. 2175–2179, <https://doi.org/10/ggwb6b>.
- [23] Q. Li, Y. Zhang, B. Li, L. Cao, and P. C. Woodland, “Residual energy-based models for end-to-end speech recognition,” in *Annual Conference of the International Speech Communication Association (Interspeech)*, 2021, pp. 4069–4073, <https://doi.org/10/hcb4pg>.
- [24] Y. Li, Z. Zhao, O. Klejch, P. Bell, and C. Lai, “ASR and emotional speech: A word-level investigation of the mutual impact of speech and emotion recognition,” in *Annual Conference of the International Speech Communication Association (Interspeech)*, 2023, pp. 1449–1453, <https://doi.org/10/hbsppj>.
- [25] A. Vinnikov *et al.*, “NOTSOFAR-1 challenge: New datasets, baseline, and tasks for distant meeting transcription,” in *Annual Conference of the International Speech Communication Association (Interspeech)*, 2024, pp. 5003–5007, <https://doi.org/10/hcb4pf>.
- [26] T. Yoshioka, H. Erdogan, Z. Chen, X. Xiao, and F. Alleva, “Recognizing overlapped speech in meetings: A multichannel separation approach using neural networks,” in *Annual Conference of the International Speech Communication Association (Interspeech)*, 2018, pp. 3038–3042, <https://doi.org/10/gfb6x6>.
- [27] J. R. Rice, “The algorithm selection problem,” in *Advances in Computers*, vol. 15, 1976, pp. 65–118, <https://doi.org/10/cqnhc2>.
- [28] J. G. Fiscus, “A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER),” in *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 1997, pp. 347–354, <https://doi.org/10/cg5c35>.
- [29] L. Mangu, E. Brill, and A. Stolcke, “Finding consensus in speech recognition: word error minimization and other applications of confusion networks,” *Computer Speech & Language*, vol. 14, no. 4, pp. 373–400, 2000, <https://doi.org/10/fbdvbw>.
- [30] G. Evermann and P. C. Woodland, “Posterior probability decoding, confidence estimation and system combination,” in *NIST Speech Transcription Workshop*, 2000, <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.27.1408>.
- [31] I. Gitman, V. Lavrukhin, A. Laptev, and B. Ginsburg, “Confidence-based ensembles of end-to-end speech recognition models,” in *Annual Conference of the International Speech Communication Association (Interspeech)*, 2023, <https://doi.org/10/hcjf9n>.
- [32] A. S. Namli, O. F. Celik, E. Yapagci, and S. S. Kozat, “ADASTT: Adaptive speech-to-text algorithm selection via meta-learning,” *IEEE Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 34, pp. 1787–1801, 2026, <https://doi.org/10/hcb4nn>.
- [33] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, “Adaptive mixtures of local experts,” *Neural Computation*, vol. 3, no. 1, pp. 79–87, 1991, <https://doi.org/10/cnsnqg>.

- [34] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the EM algorithm," *Neural Computation*, vol. 6, no. 2, pp. 181–214, 1994, <https://doi.org/10/ftjqmt>.
- [35] S. E. Yuksel, J. N. Wilson, and P. D. Gader, "Twenty years of mixture of experts," *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, vol. 23, no. 8, pp. 1177–1193, 2012, <https://doi.org/10/f384ds>.
- [36] N. Shazeer et al., "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," in *International Conference on Learning Representations (ICLR)*, 2017, <https://arxiv.org/abs/1701.06538>.
- [37] D. Lepikhin et al., "GShard: Scaling giant models with conditional computation and automatic sharding," in *International Conference on Learning Representations (ICLR)*, 2021, <https://openreview.net/forum?id=qrwe7XHTmYb>.
- [38] W. Fedus, B. Zoph, and N. Shazeer, "Switch Transformers: Scaling to trillion parameter models with simple and efficient sparsity," *Journal of Machine Learning Research (JMLR)*, vol. 23, no. 120, pp. 1–39, 2022, <https://jmlr.org/papers/v23/21-0998.html>.
- [39] C. Riquelme et al., "Scaling vision with sparse mixture of experts," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2021, pp. 8583–8595, <https://proceedings.neurips.cc/paper/2021/hash/48237d9f2dea8c74c2a72126cf63d933-Abstract.html>.
- [40] Z. You, S. Feng, D. Su, and D. Yu, "SpeechMoE: Scaling to large acoustic models with dynamic routing mixture of experts," in *Annual Conference of the International Speech Communication Association (Interspeech)*, 2021, pp. 2077–2081, <https://doi.org/10/hcb4pm>.
- [41] Z. You, S. Feng, D. Su, and D. Yu, "SpeechMoE2: Mixture-of-experts model with improved routing," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 7217–7221, <https://doi.org/10/hcb4nd>.
- [42] K. Hu, B. Li, T. N. Sainath, Y. Zhang, and F. Beaufays, "Mixture-of-expert Conformer for streaming multilingual ASR," in *Annual Conference of the International Speech Communication Association (Interspeech)*, 2023, pp. 3327–3331, <https://doi.org/10/hcb4pp>.
- [43] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, and Y. Wang, "Dynamic neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 44, no. 11, pp. 7436–7456, 2022, <https://doi.org/10/gm4tsr>.
- [44] S. Scardapane, A. Baiocchi, A. Devoto, V. Marsocci, P. Minervini, and J. Pomponi, "Conditional computation in neural networks: Principles and research trends," *Intelligenza Artificiale*, vol. 18, no. 1, pp. 175–190, 2024, <https://doi.org/10/hcb4m8>.
- [45] V. Campos, B. Jou, X. Giró-i-Nieto, J. Torres, and S.-F. Chang, "Skip RNN: Learning to skip state updates in recurrent neural networks," in *International Conference on Learning Representations (ICLR)*, 2018, <https://openreview.net/forum?id=HkwVAXyCW>.
- [46] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez, "SkipNet: Learning dynamic routing in convolutional networks," in *European Conference on Computer Vision (ECCV)*, 2018, pp. 420–436, <https://doi.org/10/ggwcgt>.
- [47] S. Teerapittayanon, B. McDanel, and H. T. Kung, "BranchyNet: Fast inference via early exiting from deep neural networks," in *International Conference on Pattern Recognition (ICPR)*, 2016, pp. 2464–2469, <https://doi.org/10/ggv9jr>.
- [48] J. Xin, R. Tang, J. Lee, Y. Yu, and J. Lin, "DeeBERT: Dynamic early exiting for accelerating BERT inference," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020, pp. 2246–2251, <https://doi.org/10/grk46t>.
- [49] S. Laskaridis, A. Kouris, and N. D. Lane, "Adaptive inference through early-exit networks: Design, challenges and directions," in *International Workshop on Embedded and Mobile Deep Learning*, 2021, pp. 1–6, <https://doi.org/10/hcb4nf>.
- [50] A. Graves, "Adaptive computation time for recurrent neural networks," 2016, <http://arxiv.org/abs/1603.08983>.
- [51] M. Elbayad, J. Gu, E. Grave, and M. Auli, "Depth-adaptive transformer," in *International Conference on Learning Representations (ICLR)*, 2020, <https://openreview.net/forum?id=SJg7KhVKPH>.
- [52] D. Yu and L. Deng, *Automatic speech recognition: A deep learning approach*. Springer London, 2015, <https://doi.org/10/qvbk>.
- [53] J. Carletta et al., "The AMI meeting corpus: A pre-announcement," in *International Workshop on Machine Learning for Multimodal Interaction (MLMI)*, 2006, pp. 28–39, <https://doi.org/10/bf23qf>.
- [54] M. Del Rio, P. Ha, Q. McNamara, C. Miller, and S. Chandra, "Earnings-22: A practical benchmark for accents in the wild," 2022, <https://doi.org/10/hcjgbc>.
- [55] D. Galvez et al., "The People's Speech: A large-scale diverse English speech recognition dataset for commercial usage," in *Conference on Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2021, <https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/202cb962ac59075b964b07152d234b70-Abstract-round1.html>.
- [56] T. Olatunji et al., "AfriSpeech-200: Pan-African accented speech dataset for clinical and general domain ASR," *Transactions of the Association for Computational Linguistics (TACL)*, vol. 11, pp. 1669–1685, 2023, <https://doi.org/10/hck8mr>.
- [57] Cohere Labs, "Cohere-Transcribe-03-2026 model card," 2026, <https://huggingface.co/CohereLabs/cohere-transcribe-03-2026>.
- [58] IBM Granite Team, "Granite-Speech-4.1-2b model card," 2026, <https://huggingface.co/ibm-granite/granite-speech-4.1-2b>.
- [59] M. Sekoyan et al., "Canary-1B-v2 & Parakeet-TDT-0.6B-v3: Efficient and high-performance models for multilingual ASR and AST," 2025, <https://doi.org/10/hck8mv>.
- [60] N. Zeghidour et al., "Streaming sequence-to-sequence learning with delayed streams modeling," 2025, <https://doi.org/10/hck8ms>.
- [61] NVIDIA, "Parakeet-CTC-1.1b model card," 2023, <https://huggingface.co/nvidia/parakeet-ctc-1.1b>.
- [62] D. Rekish et al., "Fast Conformer with linearly scalable attention for efficient speech recognition," in *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2023, pp. 1–8, <https://doi.org/10/lg7ns99>.
- [63] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016, pp. 785–794, <https://doi.org/10/gdp84q>.
- [64] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural Computation*, vol. 10, no. 7, pp. 1895–1923, 1998, <https://doi.org/10/fqc9w5>.
- [65] T. Hori, N. Moritz, C. Hori, and J. Le Roux, "Transformer-based long-context end-to-end speech recognition," in *Annual Conference of the International Speech Communication Association (Interspeech)*, 2020, pp. 5011–5015, <https://doi.org/10/hck8mq>.
- [66] M. Bain, J. Huh, T. Han, and A. Zisserman, "WhisperX: Time-accurate speech transcription of long-form audio," in *Annual Conference of the International Speech Communication Association (Interspeech)*, 2023, pp. 4489–4493, <https://doi.org/10/gt7nbf>.